Université de Franche-Comté **UFR Sciences, Techniques et Gestion de l'Industrie**
LIFC **École doctorale SPIM**
Numéro d'ordre:

Les problèmes de tournées de véhicules en planification industrielle : classification et comparaison d'opérateurs évolutionnaires

THÈSE

sera présentée et soutenue publiquement le 30 août 2010

pour l'obtention du

Grade de Docteur de l'Université de Franche-Comté

École Doctorale SPIM - Spécialité Informatique

par

Mais HAJ RACHID

Composition du jury

Président :	Marie-Claude PORTMANN	Professeur de l'école des Mines de Nancy
Rapporteurs :	Bernard GRABOT	Professeur à l'Ecole Nationale des Ingénieurs de Tarbes
	Christian PRINS	Professeur à l'Université de Technologie de Troyes
Directeur de thèse :	François SPIES	Professeur à l'Université de Franche-Comté
Encadrants:	Christelle BLOCH	Maître de Conférences à l'Université de Franche-Comté
	Pascal CHATONNAY	Maître de Conférences à l'Université de Franche-Comté
	Wahiba RAMDANE-CHERIF	Maître de Conférences à l'école des Mines à Nancy

Laboratoire d'Informatique de l'Université de Franche-Comté – **EA 4269**

Remerciements

Je désire avant tout exprimer ma profonde gratitude au gouvernement Syrien et à l'Université d'Alep pour le soutien financier qu'ils m'ont accordé tout au long de cette recherche.

Je remercie Monsieur Christian Prins, professeur à l'Université de Technologie de Troyes, et Monsieur Bernard Grabot, professeur à l'Ecole Nationale des Ingénieurs de Tarbes, d'avoir accepté d'être rapporteurs de ce mémoire. Je tiens aussi à remercier Mademoiselle Marie-Claude Portmann, professeur à l'école des Mines de Nancy, qui m'a fait l'honneur d'avoir accepté d'examiner ce travail. C'est un très grand plaisir pour moi d'avoir pu associer ces trois noms à ce travail.

Cette thèse n'aurait pu avoir le jour sans l'aide, les compétences, l'enthousiasme, la patience et la générosité de ma co-encadrante, Madame Christelle Bloch, maître de conférences à l'Université de Franche-Comté. Elle m'a offert une ambiance de travail motivante et encourageante, je ne saurais jamais la remercier assez pour son caractère accueillant. Merci Christelle d'avoir toujours répondu avec une gentillesse et une grande compréhension à mes nombreuses questions. Merci Christelle d'avoir passé des nuits de travail avec moi, quelques fois jusqu'à six heures du matin!!!

Je tiens aussi à remercier ma co-encadrante Madame Wahiba Ramdane-Cherif, maître de conférences à l'école des Mines de Nancy, de m'avoir donné l'opportunité de réaliser ce travail et de m'avoir orienté tout au long de cette thèse. Son expérience dans le domaine de tournées de véhicules, sa disponibilité, son aide et ses suggestions m'ont permis de mener à bon terme ce travail.

Mes plus sincères remerciements vont également à Monsieur Pascal Chatonnay, maître de conférences à l'Université de Franche-Comté, pour ses remarques toujours constructives, pour sa gentillesse et pour ses conseils très utiles.

Je voudrais aussi remercier Monsieur François Spies, professeur à l'Université de Franche-Comté, qui a accepté de diriger ce travail et Monsieur Jean-Christophe Lapayre, professeur à l'Université de Franche-Comté, directeur du laboratoire LIFC, qui m'a accueilli dans son établissement.

Les expérimentations réalisées dans cette recherche n'auraient pu être obtenues sans l'aide de certaines personnes : Maan Haj Rachid, Mouhannad Alattar, Laurent Hippolyte et Wahabou Abdou. Un grand merci à toutes ces personnes ainsi qu'à ma collègue Kahina Boutoustous pour ses conseils et son aide.

Je tiens aussi à remercier le GDR RO, d'avoir soutenu une part des travaux de cette thèse à travers le projet Vero & Cleo (Vehicle Routing : Classification and Evolutionary Operators). Ce projet a regroupé plusieurs chercheurs de différents laboratoires :

Mademoiselle Marie-Claude Portmann, Professeur à l'Ecole Des Mines De Nancy, Madame Wahiba Ramdane-Cherif, maître de conférences à l'Ecole Des Mines De Nancy, Madame Christelle Bloch, maître de conférences au Laboratoire Informatique de l'Université de Franche-Comté (LIFC) à Montbéliard, Madame Marie-Ange Manier, maître de conférences à l'Université de Technologie de Belfort-Montbéliard (UTBM), Monsieur Pascal Chatonnay, maître de conférences au Laboratoire Informatique de l'Université de Franche-Comté (LIFC) à Montbéliard, Monsieur Christophe Lenté, maître de conférences à l'Université de Tours et Messieurs Mathieu Rouleau et Patrice Leclaire, doctorants à l'Université de Tours et à l'IUT Montluçon – Moulins –Vichy. Je les remercie pour leurs collaborations, qui ont permis de valider et d'enrichir la classification des problèmes de tournées proposée dans cette thèse.

Je remercie également Madame Laetitia Trussardi, l'ancienne secrétaire de mon laboratoire pour les nombreux services rendus, toujours avec le sourire, tout au long de ces années d'études.

Un grand merci profondément sincère à mes parents et ma famille pour l'amour inconditionnel et les encouragements soutenus qui m'ont donné la force de continuer malgré la grande distance.

Je réserve le mot de la fin à mon mari, Mohammad Bassem, qui a toujours été à mes côtés pour partager mes frustrations avant mes joies. Je n'y serais pas arrivé sans lui. Je le remercie de tout mon cœur pour sa patience pendant mes absences lors des promenades, des sorties familiales, des jours de fêtes, …et pendant mes nuits studieuses.

Dédicace

A ma tendre mère et mon trop cher père

A mon bien-aimé mari Mohammad Bassem

A Ahmad Samir et Ibrahim, mes enfants et joies de ma vie

A mes frères merveilleux

A ma famille

A chaque personne sincère dans la vie

.... Je dédierais ce travail.

Table des matières

Liste des tableaux

Table des figures

Introduction

Initialement, les termes « planification industrielle » (*industrial planning*) désignaient diverses problématiques rencontrées dans le secteur de la production industrielle de biens. Toutes avaient un but commun : planifier dans le temps diverses actions pour optimiser cette production, autrement dit pour maximiser sa rentabilité. Cette question prise dans sa globalité comporte de nombreux leviers d'amélioration, de la construction d'unités de productions à l'ordonnancement détaillé de tâches dans un atelier de fabrication. Elle est donc trop complexe pour être modélisée de manière analytique et résolue par des approches exactes.

Par conséquent, la communauté scientifique s'est employée à décomposer ce problème en sous-problèmes pouvant être abordés à l'aide des approches de recherche opérationnelle et des moyens disponibles. Un découpage communément admis repose sur trois niveaux de décision : stratégique, tactique et opérationnel. Ces niveaux se distinguent notamment par l'horizon temporel considéré, la précision des estimations possibles, le niveau d'agrégation des données. En particulier, le niveau stratégique porte sur des investissements lourds, à long terme, sur la base de l'estimation d'une enveloppe budgétaire globale. Il peut s'agir par exemple de décider de l'implantation d'unités de production en différents lieux géographiques en fonction des parts de marché escomptées sur un territoire. A l'autre extrême, le niveau opérationnel détaille des données de fabrication précises (durées des opérations, coûts, délais...), sur une ou plusieurs ressources de production, pour en planifier l'exécution unité de temps par unité de temps.

Le terme « ordonnancement » (*scheduling*) est alors plus communément employé que celui de « planification ». Ordonnancer le passage de (n) travaux sur une machine pour minimiser la date de fin de production, en tenant compte des coûts de changement d'outillage induits par les changements de série, par exemple, relève du niveau opérationnel. Ce schéma général a par la suite subi de nombreuses évolutions et modifications.

Premièrement, les chercheurs se sont intéressés également à la planification de la production de services, tels que le transport. Le transport des biens entre les moyens de production était généralement négligé dans la définition des problèmes de planification de la production. Les chercheurs se sont donc intéressés à son optimisation, que ce soit au niveau stratégique, tactique ou opérationnel. Ainsi, on planifiait soit des tâches de transport soit des tâches de production, mais généralement de manière indépendante et séparée. Le terme « logistique » était employé le plus souvent pour regrouper les problèmes de planification du transport, quel que soit le niveau de décision considéré, du transport de marchandises entre usines au transport des pièces en atelier de fabrication, en incluant bien sûr toute la partie approvisionnement en matière première

et la distribution jusqu'au client final. La même communauté scientifique s'est également intéressée à la planification du transport de personnes, car les problèmes à modéliser et résoudre étaient sensiblement similaires à ceux rencontrés pour le transport de biens. Et une grande partie des problèmes considérés a donc été regroupée sous l'appellation « problèmes de tournées » (*routing problems*). Le plus simple et le plus célèbre d'entre eux sans doute est le problème du voyageur de commerce (*Traveling Salesman Problem* ou *TSP*). Il consiste à planifier les déplacements d'un individu rendant visite à des clients répartis sur une zone géographique donnée, de manière à minimiser la distance qu'il parcourt. Le TSP est utilisé pour modéliser naturellement de nombreux problèmes de tournées de véhicules, que ce soit pour collecter/distribuer des biens ou transporter des personnes. Une variante intégrant la notion de capacité du(es) véhicule(s) a également été définie : le *Vehicle Routing Problem* (*VRP*) ou *Capacitated Vehicle Routing Problem* (*CVRP*). Mais le TSP et le VRP sont très largement utilisés aussi pour modéliser des problèmes de planification/ordonnancement de la production, en particulier en ordonnancement d'atelier (*shop scheduling*).

Puis, la notion de logistique a évolué. De plus en plus, les chercheurs se sont attachés à planifier les tâches de transport et de production de manière intégrée plutôt que séparée. La chaine logistique incluant les deux aspects, production et transport, est à présent considérée dans sa globalité, sous l'appellation « *supply-chain* », et le terme logistique est de plus en plus utilisé à présent pour représenter l'ensemble de la planification industrielle.

Parallèlement, les moyens de résolution eux aussi ont fortement évolué. D'une part par l'apparition de méthodes approchées, en particulier les métaheuristiques, qui permettent d'aborder des problèmes plus complexes au détriment de la garantie de l'optimalité de la solution. D'autre part par les progrès techniques aboutissant à la mise sur le marché de moyens de calcul plus puissants, utilisés individuellement ou en réseau. Ces avancées, ont permis aux chercheurs de moins se limiter dans la définition des problèmes à modéliser et à résoudre. De ce fait, ces derniers incluent à présents d'autres préoccupations, tels que l'intégration d'aspects humains ou liés au développement durable. La conjugaison de toutes ces évolutions a fait des problèmes de tournées de véhicules l'un des thèmes de recherche les plus prisés en planification industrielle.

De ce fait de très nombreux travaux leur sont consacrés et une multitude de variantes de problèmes de plus en plus fidèles aux applications réelles, et donc de plus en plus complexes, ont été définies à partir des problèmes de base que représentaient le TSP et le VRP. La littérature dédiée à ce domaine est très vaste et il est très difficile, en particulier pour de jeunes doctorants, de mener une étude bibliographique sur le sujet. L'une des difficultés majeures provient du manque de standardisation dans la dénomination des variantes étudiées par les auteurs. De nombreux sigles sont utilisés pour désigner ces variantes de problème en fonction de leurs hypothèses, contraintes et objectifs. Mais comme il n'y a aucune notation de référence, certains auteurs utilisent le même sigle pour des problèmes différents ou au contraire un sigle identique pour des variantes qui diffèrent sur certains aspects.

Cette thèse vise à combler ce manque de standardisation. Elle propose une notation des problèmes de tournées de véhicules qui puisse être utilisée comme référence pour

que l'identification des variantes de problème soit plus homogène. La notation proposée repose sur un principe similaire à celles développées précédemment en ordonnancement d'atelier. Chaque variante de problème est désignée à l'aide de paramètres qui décrivent les hypothèses, les contraintes et les objectifs qu'elle contient. Cela permet d'identifier plus rapidement quels sont les travaux de la littérature qui portent effectivement sur des problèmes identiques et dont les résultats peuvent donc objectivement être comparés. Dans un second temps, son utilisation peut également permettre une classification des articles scientifiques du domaine, en utilisant un sous-ensemble pertinent des paramètres de notation. Un tel classement met en évidence les variantes les plus étudiées, mais également les pistes qui restent encore quasiment inexplorées. Il permet aussi, lorsque l'on analyse l'état de l'art ainsi classé sous un autre angle, de chercher des relations entre les variantes de problème et les types de méthodes de résolution. Enfin, pour un type de méthode donné, il est également possible d'étudier les relations entre certaines caractéristiques de la variante à résoudre et l'efficacité des opérateurs utilisés dans l'approche de résolution. A terme, l'identification de tels liens pourraient permettre d'établir des règles pour guider la conception des algorithmes de résolution en fonction des caractéristiques du problème posé.

Naturellement, il aurait été illusoire, au cours d'une seule thèse, de mener à bien tous ces travaux pour l'ensemble des articles de la littérature et l'ensemble des approches de résolution existantes.

Ce mémoire présente la notation, puis illustre ces différentes formes d'utilisation pour une partie de l'état de l'art des problèmes de tournées. Pour l'étude détaillée des relations entre caractéristiques des variantes et opérateurs de résolution, la méthode considérée est l'algorithmique évolutionnaire et la variante étudiée est le VRP avec contrainte de capacité.

Le mémoire est divisé en quatre chapitres. Le premier chapitre situe le VRP dans le contexte général de la planification industrielle. Il montre quelles sont les utilisations multiples de ce modèle tant pour la production de biens que de services. Cela permet de mettre en évidence les enjeux importants liés à sa résolution. Puis le chapitre donne une définition plus formelle du VRP. Il décrit ensuite différentes variantes de plus en plus complexes qui sont considérées dans la littérature. Il donne pour chacune d'entre elles le (ou les) sigle(s) traditionnellement utilisés pour les identifier. Enfin, il fait un inventaire rapide des divers types de méthodes de résolution généralement utilisées pour l'aborder.

Le second chapitre présente quelques notations existantes pour les problèmes de planification/ordonnancement, en particulier pour les problèmes de tournées de véhicules. Il les confronte à la variété des variantes présentées au chapitre un pour montrer leurs avantages et leurs limites. Puis il expose la notation proposée, et illustre ses applications pour le classement d'une partie de l'état de l'art. Deux angles de vue sont utilisés pour cela. La première application ne concerne pas les méthodes de résolution, elle met en regard les sigles utilisés dans la littérature et l'identification utilisant la nouvelle notation. La seconde illustre une utilisation plus détaillée. Elle lie aux variantes identifiées les opérateurs de résolution utilisés dans des approches de résolution évolutionnaires. Dans les deux cas, un tableau de synthèse est tout d'abord présenté puis chaque référence citée est ensuite commentée.

Les troisième et quatrième chapitre illustrent le dernier niveau d'application. Le but à terme est de proposer des règles pour guider la conception d'algorithmes de résolution

évolutionnaires, en fonction des caractéristiques de la variante de problème à résoudre. La première étape consiste à illustrer à titre d'exemple la comparaison des performances d'opérateurs génétiques classiques (croisements et mutations) par expérimentation. 3 types de codage et 2 types de sélection ont également été retenus, pour estimer la sensibilité des croisements et mutations au choix du codage (souvent réalisé avant le choix des opérateurs) et à l'importance de la pression sélective de l'algorithme. Cette étude repose sur la résolution de plusieurs benchmarks de la littérature pour une variante de problème de tournées avec contrainte de capacité.

Le chapitre trois rappelle succinctement les principes d'un algorithme évolutionnaire simple, et présente plusieurs types d'opérateurs existant. Il détaille l'algorithme évolutionnaire utilisé pour les expérimentations, en particulier les codages et les sélections utilisés et les opérateurs comparés.

Le chapitre 4 explique la démarche et les conditions d'expérimentation, puis analyse et commente les résultats obtenus.

Le mémoire s'achève par une conclusion présentant un bilan des objectifs atteints et des limites des travaux réalisés avant d'en déduire les principales perspectives.

Chapitre 1

Le problème de tournées de véhicules : contexte, définition, variantes et méthodes de résolution

Sommaire

Dans un premier temps, nous avons cherché à situer les problèmes de tournées ou *Vehicle Routing Problems* (VRP) dans le contexte général de la planification industrielle. Pour cela, la section 1.1 dresse un historique montrant l'évolution du management industriel. Une évolution qui a permis de placer le transport au cœur des préoccupations des entreprises et de donner à la logistique plus d'ampleur. La section 1.2 donne un aperçu des applications des modèles de tournées au domaine de la gestion de production. Une définition plus formelle du VRP est présentée en section 1.3. La section 1.4 enchaîne sur la description des variantes du VRP les plus étudiées dans la littérature. Enfin la section 1.5 fait un inventaire des méthodes de résolution généralement utilisées pour aborder le VRP et ses variantes.

1.1 Introduction : historique et contexte en planification industrielle

L'organisation d'un système de distribution occupe une part importante du management industriel. Ce dernier recouvre un très vaste ensemble de domaines (Baglin *et al*., 2001) :

- la conception de produits vendus par l'entreprise,
- la conception des processus de production qui permettent de fabriquer ces produits
- la gestion des flux physiques à tous les niveaux et des stocks,
- les technologies mises en œuvre dans les produits et dans les processus
- la politique d'achat des matières premières et des composants ainsi que de toutes les prestations et services,
- la politique de qualité à tous les niveaux,
- ainsi que le management des ressources humaines mobilisées dans le domaine industriel comme dans l'étude proposée par (Grabot et Letouzey, 2000).

Le management industriel a évolué au fil des siècles :

Au 14ème siècle, ont commencé à se poser les problèmes de gestion de projet. Par exemple- pour construire des édifices de grande taille, il fallait standardiser pour mettre en place des processus.

Le début du 20ème siècle a été caractérisé par l'apparition des méthodes de rationalisation du travail : vers 1950, la naissance de la recherche opérationnelle a permis le recours à une gestion scientifique de la production.

Au milieu des années 1960, cette gestion scientifique de la production a connu un développement intéressant avec l'arrivée de l'informatique et les progiciels. Il est devenu possible de gérer une production plus complexe. Ceci a permis le développement du concept de la gestion intégrée de la production et des progiciels de gestion de production assistée par ordinateur (GPAO), qui prennent en compte tous les aspects de la logistique des achats à la livraison des produits.

Dans les années 1980, les bouleversements des marchés et les exigences de performance financière, combinés aux progrès technologiques (TIC, nouveaux procédés...) ont forcé les entreprises à proposer des produits de bonne qualité à bas prix. Afin d'améliorer les rendements et les temps de cycle de production par rapport à la concurrence, les entreprises ont eu recours au management par les méthodes de « juste à temps » (JIT : *Just-in-time*), qui permettent de limiter les stocks de composants en organisant et en ordonnançant précisément l'approvisionnement avec les fournisseurs. Il est devenu alors nécessaire d'avoir une totale disponibilité des équipements, d'où la nécessité de recevoir les marchandises commandées à temps. Ceci a impliqué une redéfinition des relations que l'on entretient avec les fournisseurs, passant d'une relation parfois conflictuelle à un mode de coopération (ou partenariat) dans lequel fournisseurs et entreprises clientes collaborent pour le succès de leur activité commune en partageant les risques. C'est ainsi que la fonction distribution a été ajoutée à la fonction approvisionnement pour former la « logistique intégrée », connue aussi sous le nom de gestion de la chaîne logistique ou *Supply Chain Management* (SCM) (Tan, 2001).

De 1990 à nos jours, le SCM s'étend à tous les fournisseurs (sur plusieurs rangs) et à toutes les entités de la distribution (entrepôts, grossistes, détaillants,…). Désormais, le SCM se centre davantage sur une famille de produits finis. L'idée de cette intégration est de répandre les bonnes pratiques de gestion à tous les maillons de la chaîne afin d'améliorer globalement la performance (Erschler et Grabot, 2001). (Rota, 1998) définit la chaîne logistique d'un produit comme l'ensemble des entreprises qui interviennent dans le processus de fabrication, de distribution, et de vente du produit, du premier des fournisseurs au client ultime. Cette notion de chaîne logistique est ainsi très étendue et très complexe (voir annexe A) car les fournisseurs ont eux-mêmes leurs propres fournisseurs et les clients sont souvent fournisseurs d'autres clients.

La chaîne logistique est ainsi constituée de deux processus de base:

- Processus de planification de la production et de gestion des stocks : la planification de la production désigne la gestion du processus de production (planification de la matière première, planification du processus de transformation et de manutention des matières). La gestion du stock désigne la gestion des politiques de stockage et des procédures de stock de matière première, des en-cours et des produits finis
- Processus de distribution : ce dernier désigne la manière dont les produits sont acheminés. Ceci pourrait être fait soit à travers les centres de distribution, soit directement des stocks de produits finis vers les clients.

Les performances des deux processus cités ci-dessus sont interdépendantes. Par exemple, il ne sert à rien d'avoir d'excellentes performances en production si des fournisseurs non fiables obligent à conserver d'importants stocks de matières premières, ou si par exemple le système de distribution ne permet pas de livrer les produits dans des délais très courts. La collecte ou la distribution (de marchandises ou de personnes) constitue en particulier l'un des maillons de la chaîne logistique les plus étudiés vu les enjeux environnementaux et économiques associés. Ces aspects sont devenus encore plus cruciaux aujourd'hui, par la nécessité du développement durable admise par le plus grand nombre et par les prix des carburants qui semblent augmenter de manière inéluctable et irrémédiable. Cela s'est traduit par une forte activité de recherche dans le domaine du transport ces dernières années. Les chercheurs s'emploient à concevoir des modèles et approches de résolution de mieux en mieux adaptés aux problèmes réels. L'ensemble des problèmes d'optimisation correspondants est regroupé dans la littérature sous l'expression générique de « problèmes de tournées » (*Routing Problems*). Cette appellation unique recouvre cependant des problèmes qui peuvent présenter des caractéristiques très différentes.

Dans les problèmes de tournées, il s'agit de déterminer, pour un ensemble donné de « clients », par quels véhicules de la flotte ils seront visités et à quel moment dans la tournée ils le seront. Deux grandes classes émergent en fonction de la position des clients : s'ils se trouvent sur des points précis d'un réseau, le problème étudié est un problème de tournées sur nœuds. En revanche, si les ("clients") sont localisés sur des liens du réseau, il s'agit d'un problème de tournées sur arcs. La deuxième classe apparaît quand les "clients" se trouvent distribués le long des rues ou lorsque la rue elle-même nécessite un service.

Le problème de base de la classe des problèmes de tournées est le problème de voyageur de commerce (*Traveling Salesman Problem* ou TSP) (voir section 1.3). De nombreuses problématiques, dans divers secteurs d'application, peuvent être avantageusement modélisées et résolues en s'appuyant sur des modélisations et résolutions du problème de voyageur de commerce (TSP) ou de l'une de ses variantes.

1.2 Application des modèles de tournées en planification industrielle

Les chercheurs n'ont cessé d'enrichir les modèles de tournées avec des contraintes complexes afin de répondre mieux aux besoins des applications industrielles. Ceci a donné naissance à plusieurs variantes (voir section 1.4) pour modéliser naturellement des applications en transport de biens comme la livraison de marchandise, la livraison de carburant à des stations service, l'alimentation des distributeurs automatiques en boissons et en billets, la distribution d'aliments pour bétail à des éleveurs, la collecte de lait en environnement rural, la collecte de déchets industriels ; mais aussi en transport de « service » comme l'organisation des tournées pour les soins à domicile, pour la maintenance et pour la livraison à domicile ; et enfin le transport de personnes comme l'organisation des services de taxi à la demande (Chevrier *et al*., 2008), le covoiturage, le transport scolaire.

L'application des modèles de tournées n'a pas été restreinte au domaine de la distribution et du transport. Ils ont été aussi très largement utilisés pour modéliser des problèmes de production très variés, comme le montrent les exemples d'applications ci-dessous. Les problèmes de tournées servent pour

- La conception des systèmes de production comme l'optimisation de l'agencement et de l'organisation des ateliers de production. Ainsi (Cheng *et al.*, 2007) ont proposé une organisation cellulaire de l'atelier de production en se basant sur des modèles de tournées. Le principe de l'organisation cellulaire consiste à fabriquer un ensemble de produits similaires (familles de produits) au moyen d'un ensemble de machines partitionné en sous-ensembles. Par exemple, il est possible de grouper les produits en espérant profiter de leurs similarités fortes : mêmes composantes, mêmes opérations, etc. Les machines peuvent être groupées en fonction des gammes opératoires dans l'optique de limiter les mouvements intercellulaires et de réduire les manutentions.
- La conception d'algorithmes pour automatiser le travail des robots ou des machines notamment dans le domaine de l'industrie électronique (Chan et Mercier, 1989) (Kumar et Zhonghui, 2003). Un récent état de l'art (Ayob et Kendall, 2009) a pointé l'importance et l'efficacité des modèles de tournées dans l'optimisation des temps de production pour le placement, le montage et l'assemblage de composants sur des cartes de circuits électroniques.
- L'optimisation des temps de production, en minimisant les temps de découpes. En effet, dans de nombreuses industries, se pose, à un moment ou un autre du processus de fabrication, le problème de la découpe de pièces (de formes rectangulaires, régulières ou non régulières) dans un ou plusieurs supports. (Bart *et al.*, 2007) se sont intéressés à l'optimisation du temps de découpe de tôles métalliques où les tracés sont déjà prédéfinis. Ils ont modélisé le mouvement de la tête de découpe par des modèles de tournées.
- L'optimisation du transport de produits dans les ateliers de production et les entrepôts logistiques. Dans les ateliers de production, le transport est couramment réalisé à l'aide de véhicules autoguidés (AGV, *Automated guided vehicle system*). Ce sont des véhicules équipés d'un dispositif de guidage automatique, qui suivent un trajet défini et qui s'arrêtent à chaque station d'usinage ou d'assemblage pour charger ou décharger des pièces. Intuitivement, le routage des AGVs peut être considéré comme une variante du problème de tournées de véhicules (Bodin et Golden, 1981) (Bodin *et al.,* 1983). Plusieurs applications sont concernées (Ling *et al*., 2002) comme l'approvisionnement des composants et de matières premières, l'évacuation des produits finis ou semi-finis, l'évacuation des déchets, l'approvisionnement des articles de conditionnement (emballage, piles de palettes,…). Dans les entrepôts logistiques, le transport des produits est assuré par des employés utilisant du matériel de manutention. Là aussi, les modèles de tournées ont trouvé un large cadre d'application pour l'optimisation du temps de travail des opérateurs, ainsi que dans l'optimisation des tournées des préparateurs de commandes. Par exemple, (Theys *et al.*, 2010) ont comparé dans leur étude deux approches pour l'optimisation des tournées de préparateurs de commandes dans un entrepôt logistique : une approche classique spécifique à l'entrepôt, et une approche

utilisant les modèles de tournées. Les résultats obtenus montrent des gains significatifs avec l'utilisation de ces derniers.

- L'optimisation du temps improductif induit par les changements d'outils dans le processus de production, qui dépend fortement des tâches à exécuter. A titre d'exemple, (Aguilera *et al.*, 1996) ont présenté une application de fabrication de plusieurs modèles de bouteilles en verre. Les auteurs ont développé une méthode de résolution en deux étapes, la première permettant d'affecter les bouteilles aux machines et la deuxième consiste à trouver le meilleur ordre de passage des bouteilles par machine en utilisant le modèle des tournées. De même, (Jeong *et al.*, 1997) et (Liu *et al.*, 2008) utilisent le modèle de tournées a été utilisé pour définir la séquence de production optimale de produits polymères, afin de minimiser le montant des rebus générés lors du changement de séquence de production. Enfin, (Monkman *et al.*, 2008) considèrent un cas d'application dans l'industrie électronique, où il s'agit d'assembler à la commande des composants pour fournir des produits les plus personnalisés possible. Ce problème nécessite également une réduction des temps de production et de changement de série.

Ainsi, pendant très longtemps les chercheurs ont utilisé les modèles de tournées pour résoudre des problèmes variés soit pour la production de biens, soit pour le transport de ces biens ou de personnes, mais de manière séparée. Actuellement, de plus en plus, les chercheurs s'attachent à planifier les tâches de transport et de production de manière intégrée. En effet, avec le développement des chaînes logistiques, l'optimisation de la production est devenue plus compliquée et plus difficile. Il est alors nécessaire d'avoir un plan de production optimal pour chaque membre de la chaîne logistique considéré individuellement, mais aussi pour la chaîne entière. La production est souvent répartie sur plusieurs sites et le transport de composants de fabrication entre sites devient un enjeu à part entière dans la maîtrise du processus de production. Ainsi, bien que la production reste la préoccupation centrale, la prise en compte de la logistique inter-sites offre la possibilité d'améliorer les plans de production sur l'ensemble des sites. Si les entreprises cherchent un plan optimal pour chacune des entités, sans tenir compte des attributs de tout le système, elles font une optimisation locale. Cette approche donne des plans réalisables mais pas optimaux pour la chaîne logistique entière. C'est pourquoi les modèles de tournées se sont enrichis avec de nouvelles contraintes liées à cette optimisation dite « globale » de la chaine logistique (Chandra et Fisher, 1994) (Hwang, 2005) (Suerie, 2005) (Boudia *et al*., 2007) (Boudia et Prins, 2009).

La diversité de champs d'application dont a témoigné cette section s'est traduite dans la littérature scientifique par la définition d'une multitude de variantes de problèmes de tournées, adaptées à chacun des cas traités. Ces variantes sont définies en ajoutant des hypothèses, contraintes et objectifs à une version de problèmes de tournées dite « de base ». La section 1.3 présente la définition des problèmes de tournées telle qu'elle a été posée à l'origine, et décrit le problème de base à partir duquel les différentes variantes décrites dans la section 1.4 ont été créées.

1.3 TSP et VRP : définitions et généralités

Le problème du voyageur de commerce (*Traveling Salesman Problem ou TSP*) (Dantzig *et al*., 1954) est sans doute le plus simple, le plus connu et le plus étudié des problèmes de tournées. Il consiste à définir la tournée d'un vendeur visitant un ensemble de villes et retournant à la ville de départ (généralement nommée dépôt). Une solution fixe l'ordre dans lequel les villes seront visitées et la date exacte à laquelle le voyageur de commerce desservira chacune d'entre elles. Le problème est généralement modélisé sous forme d'un graphe dans lequel les nœuds représentent les villes à visiter et les arcs représentent les routes liant les villes deux à deux. Ainsi, le vendeur doit visiter une et une seule fois chaque nœud (la tournée correspond à un cycle hamiltonien) dans un graphe non orienté complet et valué. L'objectif du TSP est de minimiser la distance totale parcourue.

De nombreux autres problèmes de tournées de véhicules ont été définis par extension de ce problème de base, en rajoutant différentes hypothèses, contraintes et/ou objectifs. Un de ces problèmes est le problème de tournées de véhicules (*Vehicle Routing Problem, VRP*). Ce dernier a été formulé en 1959 par (Dantzig et Ramser, 1959). C'est une généralisation du TSP qui peut être décrite comme un problème de conception d'itinéraires (routes) de plusieurs véhicules, à moindre coût, d'un dépôt à un ensemble de points géographiquement dispersés (villes, magasins, entrepôts, écoles, clients, machines de fabrication dans un atelier, etc.). La figure (1.1) présente un exemple de solution pour un problème comportant 1 dépôt (carré central) et neuf clients à visiter (nœuds C_1 à C_9). Cette solution propose 3 tournées de : (C_8, C_2, C_1), (C_4, C_6, C_9), (C_5, C_3, C_7).

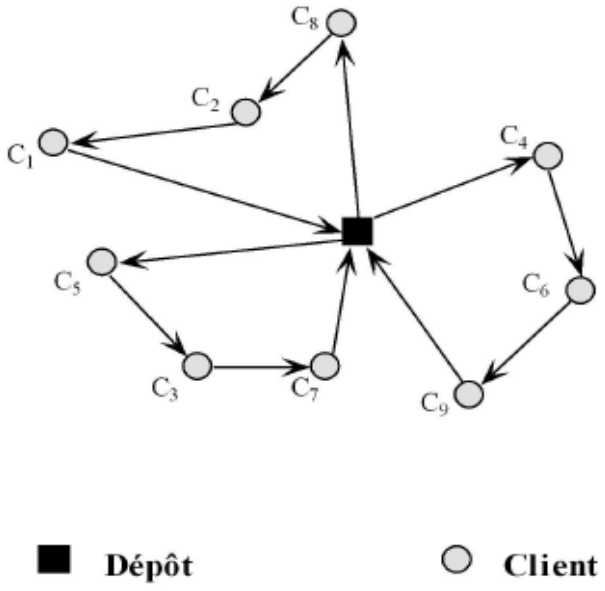


Fig. 1.1 – Exemple de solution du VRP

Les contraintes qu'une solution doit respecter sont les suivantes :

- Un client ne peut être servi que par un et un seul véhicule.

- Chaque véhicule effectue une seule tournée.
- Tous les clients doivent être desservis.

L'objectif est de trouver des itinéraires de visite de ces sites qui minimisent le coût total (par exemple la distance totale parcourue par les véhicules). Alors, c'est une généralisation du TSP, qui s'en distingue selon les auteurs par le nombre de véhicules, et/ou la contrainte de capacité et/ou la contrainte d'autonomie. La contrainte de capacité spécifie que la somme des demandes des clients regroupées en une tournée ne doit pas excéder la capacité du véhicule, sachant que la demande de chaque client est supposée inférieure à la capacité du véhicule, et que tous les véhicules sont ici de même capacité (flotte homogène). La contrainte d'autonomie indique que la durée maximale entre le départ d'un véhicule et son retour au dépôt est limitée par une borne maximale. (Sörensen, 2003) a identifié 4 types de VRP, mais ce choix reste discutable :

- le « VRP de base » est distingué du TSP uniquement par le fait que plusieurs véhicules soient disponibles. Mais ce cas est souvent désigné dans la littérature par le sigle m-TSP, ou par la lettre M en majuscule (MVRP) par d'autres auteurs, par exemple (Libralao *et al*., 2005);
- dans le *Capacitated Distance-Constrained VRP ou CDVRP*, une contrainte de capacité des véhicules et une contrainte d'autonomie des véhicules séparent le VRP du TSP (voir par exemple (Achuthan *et al*., 1995) (Renaud *et al.*, 1996) (Laporte *et al*., 2000) (Cordeau e*t al.*, 2005) ;
- seule la contrainte de capacité des véhicules distingue le *Capacitated VRP (ou CVRP)* du TSP. Ce sigle est sans doute l'un des sigles les plus utilisés (Housroum *et al*., 2005) (Lacomme et Prins, 2006). Il est pourtant troublant. Car il sous-entend que la contrainte de capacité n'apparaît pas dans le VRP de base, alors que de nombreux membres de la communauté scientifique considèrent que cette contrainte est justement celle qui distingue le VRP « de base » du TSP. Pour (Cordeau *et al*., 2005), il est nécessaire d'utiliser ce sigle CVRP pour préciser l'absence de contrainte d'autonomie. En effet, pour ces auteurs, le VRP « de base » contient les deux contraintes (autonomie et capacité).
- enfin, le *Distance Constrained VRP*, à l'inverse, distingue le VRP du TSP uniquement par la contrainte d'autonomie, comme par exemple sur le site de VRP[1].

Certains auteurs, comme par exemple (Osman, 1993) et (Kilby *et al*., 2000), contredisent cette classification de (Sörensen, 2003) en utilisant divers types de contraintes susceptibles de limiter la tournée (capacité, temps total, un nombre de clients maximum...) pour distinguer VRP et TSP.

Ainsi, la définition du sigle VRP lui-même est déjà source de difficulté et de confusion. Il paraît difficile, voire même délicat, en particulier pour un jeune chercheur, d'adopter arbitrairement l'une de ces définitions. De plus, les multiples applications de ce modèle présentées dans la section précédente ont généré, durant des années de recherches sur le VRP, diverses variantes de ce problème. La littérature qui leur est dédiée est immense. La section suivante tente de donner un aperçu des diverses évolutions qui ont été observées dans la littérature et des variantes complexes auxquelles elles ont donné naissance.

[1]http://neo.lcc.uma.es/radi-aeb/WebVRP

1.4 Variantes du problème VRP

L'apparition de nouvelles variantes du VRP est due essentiellement aux activités des chercheurs qui tentent de mettre à profit les ressources puissantes disponibles pour être de plus en plus fidèles aux problèmes rencontrés sur le terrain en planification industrielle.

En particulier, ils s'intéressent non plus seulement aux problèmes sur nœuds (dans lesquels les clients à desservir sont sur les nœuds) mais également aux problèmes sur arcs. Dans ce dernier cas, la demande à satisfaire est située sur les arcs plutôt que sur les nœuds. C'est le cas par exemple pour la collecte des ordures ménagères, dans laquelle les marchandises à collecter se trouvent le long des rues parcourues par le véhicule. Initialement, les problèmes considérés étaient symétriques, tant pour les problèmes sur nœuds que sur les problèmes sur arcs. Cela signifie que toutes les arêtes liant les nœuds peuvent être parcourues dans les deux sens et que la durée de trajet est indépendante du sens de parcours. Mais à présent, il existe des variantes plus complexes dans lesquelles le problème est asymétrique (le temps de trajet dépend du sens de parcours). Il est possible également que le problème comporte d'autres contraintes d'accessibilité comme le fait que certaines routes soient à sens unique (Alonso *et al*., 2006).

Mais ceci ne représente que l'une des évolutions observées dans la définition des problèmes de tournées étudiés. Il en existe quantité d'autres dans la littérature scientifique. Les paragraphes ci-dessous présentent les principaux problèmes dérivés du VRP qui en résultent, ainsi que les sigles habituellement utilisés pour les désigner :

Le *Capacitated Vehicle Routing Problem (CVRP)* consiste à affecter chaque client à une tournée effectuée par un seul véhicule de capacité finie. Ce véhicule commence et termine sa tournée au dépôt (Ralphs, 2003). Mais comme VRP, il est l'une des variantes pour lesquelles il existe des définitions contradictoires. Pour certains auteurs (Cordeau *et al.*, 2005) (Christiansen, 2007) elle se caractérise par la présence de la contrainte de capacité et l'absence de contrainte d'autonomie. Pour d'autres (Chen *et al.,* 2006) avec contraintes de capacité et d'autonomie.

Le *Vehicle Routing Problem with Time Windows (VRPTW),* ou VRP avec fenêtres de temps, quant à lui, spécifie que chaque client a une fenêtre de temps. Celle-ci est un intervalle de temps au cours duquel son service (par exemple le chargement ou le déchargement de marchandises) doit être accompli. Un véhicule peut arriver plus tôt, avant le début de la fenêtre de temps, mais il doit attendre jusqu'à ce que le service soit possible. Dans ce cas le temps d'attente total peut être pris en compte dans le modèle et peut représenter un objectif à minimiser (Dell'Amico *et al*., 2006). S'il arrive plus tard, le service ne peut pas être rendu et le client correspondant ne sera jamais satisfait. Dans ce cas, le problème est dit "à contraintes dures" (*hard time window constraints*). Le nombre de véhicules peut être fixé d'avance, comme dans (Louis *et al.,* 1999) (Zhu, 2000) ou être considéré comme l'une des variables du problème (Pereira *et al.,* 2002). L'objectif du problème est alors de minimiser le nombre de véhicules et la distance totale parcourue pour servir les clients sans violer les contraintes de fenêtres de temps. Dans les modèles dits à contraintes molles (*soft time windows constraints*), les véhicules peuvent servir le client en dehors de sa fenêtre de temps mais au prix d'une certaine forme de pénalité à minimiser. Bien que cette définition du VRP à fenêtre de temps soit

la plus répandue, il est possible de rencontrer dans certaines variantes du problème des fenêtres de temps relatives aux horaires d'ouverture du dépôt ou aux horaires de travail du personnel (Ombuki *et al.,* 2006).

Le *Vehicle Routing Problem with Backhauls (VRPB),* comporte deux types de clients, soit receveurs (*linehauls*), soit livreurs (*backhauls*). Tous les produits livrés sont pris d'un dépôt et tous les produits prélevés sont retournés au dépôt. Ce modèle général résulte en trois catégories de VRPB, (Parragh *et al.,* 2006a) :

- Le *VRP with Clustered Backhauls (VRPCB)*, dans lequel toutes les livraisons sont effectuées avant le premier ramassage, comme dans (Brandao, 2006) par exemple.
- Le *VRP with Mixed Linehauls and Backhauls (VRPMB)*, dans lequel les ramassages et les livraisons peuvent être mêlés dans une même tournée, comme dans (Crispim et Brandao, 2005).
- Le *VRP with Simultaneous Delivery and Pickup (VRPSDP)*, dans lequel chaque client peut être receveur et livreur en même temps (Hoff et Løkketangen, 2006).

Le *Vehicle Routing Problem with Pick-up and Delivery (VRPPD)* ou Problème de Ramassage et de Livraison est presque identique au VRPB à l'exception du fait que les produits livrés peuvent être pris, soit au dépôt, soit chez les livreurs et non uniquement au dépôt. Ce modèle recouvre deux catégories de problèmes, (Parragh *et al.*, 2006b) :

- Dans la première (*unpaired pickup and delivery points*), les demandes des clients sont indépendantes, ce qui signifie que chaque unité ramassée (au dépôt ou chez un livreur) peut être employée pour livrer n'importe quel client receveur. Généralement, ce problème est désigné sous le nom de *Pick-up and Delivery VRP (PDVRP)* et il est mono produit, c'est-à-dire que tous les produits transportés sont du même type, comme par exemple dans (Chalasani et Motwani, 1999).
- Dans la seconde catégorie (*paired pickup and delivery points*), les demandes des clients sont dépendantes (liées): chaque transport doit lier une origine et une destination précises. Deux types de ce modèle coexistent : le *Pickup and Delivery Problem (PDP)* et le *Dial-A-Ride Problem (DARP)*. PDP porte sur le transport des marchandises (Ambrosini *et al.*, 2004), tandis que DARP porte sur le transport de personnes (Aldaihani et Dessouky, 2003).

Le *Stochastic Vehicle Routing Problem (SVRP)*, est un VRP dans lequel au moins un élément du problème est aléatoire. (Gendreau *et al*., 1996) présente un état de l'art consacré au SVRP. Les trois cas les plus connus de SVRP (Cordeau *et al.*, 2005) sont :

- Le *VRP with Stochastic Customers (VRPSC)*, où P_i représente la possibilité qu'un client C_i émette une demande;
- Le *VRP with Stochastic Demands (VRPSD)*, où la demande ξ_i d'un client C_i est une variable aléatoire ;
- Le *VRP with Stochastic Travel Times (VRPSTT)*, où le temps de service s_i d'un client C_i et le temps de trajet t_{ij} d'un arc (i, j) sont des variables aléatoires.

Le *Periodic Vehicle Routing Problem (PVRP),* ou Problème de Tournées de Véhicules Multi-Périodique (*PTVMP*) considère un horizon de planification à M périodes. Chaque client doit être visité k fois au cours de l'horizon ($1 \leq k \leq M$), et les demandes quotidiennes sont fixes (Lacomme *et al*., 2005) (Francis et Smilowitz, 2006).

Le *Multi-Depot Vehicle Routing Problem (MDVRP)*, comporte plusieurs dépôts dans lesquels sont localisés les véhicules. Chaque tournée d'un véhicule doit commencer et finir au même dépôt. De plus, chaque client doit être visité exactement une fois par l'un des véhicules situés dans les dépôts indiqués par ce client (Mingozzi, 2005).

Le *Dynamic Vehicle Routing Problem (DVRP),* peut être défini comme un VRP où l'information nécessaire à la planification des tournées :

- n'est pas connue entièrement quand le processus de planification commence,
- peut changer après que les tournées initiales aient été construites.

Autrement dit, certaines données du problème dépendent explicitement du temps, comme l'apparition d'un nouveau client ou la fin de service d'un client (Bianchi, 2000).

Le *Vehicle Routing Problem with Split Delivery (VRPSD ou SDVRP)* remet en cause deux contraintes du problème classique :

- chaque client peut être visité plus d'une fois si cela est nécessaire. Autrement dit, la demande d'un client peut être divisée (split) sur plusieurs tournées.
- la demande de chaque client peut alors être plus grande que la capacité des véhicules (Archetti *et al.*, 2002).

Le *Multi-Compartment Vehicle Routing Problem (MC-VRP),* est un problème où chaque client a une demande de différents produits qui doivent être transportés en compartiments indépendants sur un même véhicule, en raison de contraintes d'incompatibilité. Dans ce cas, la division (*split*) est tolérée par demande mais interdite par produit (Mendoza *et al*., 2009) (El Fallahi *et al*., 2006).

Le *Vehicle Routing Problem with Multiple Trips (VRPMT),* se rencontre principalement lorsque la flotte de véhicules est petite. Dans ce cas, certains véhicules peuvent exécuter plusieurs tournées au cours d'une même période, en respectant une durée maximum, qui correspond à la durée de travail de chaque véhicule par période (Alonso *et al.*, 2006).

Le *Selective Vehicle Routing Problem (SVRP),* se rencontre lorsque la flotte de véhicules n'est pas suffisante pour satisfaire toutes les demandes. Par conséquent, certains des clients ne sont pas servis du tout. L'objectif de ce problème est de déterminer en même temps que la planification des tournées, le sous-ensemble de clients qui sera effectivement visité (Gueguen , 1999) (Hayari *et al.*, 2003).

Le *Heterogeneous Fleet Vehicle Routing Problem (HVRP)*, considère que les véhicules ont des caractéristiques différentes, par exemple en termes de vitesse, de coût ou de capacité (Gendreau *et al*., 1999).

Le *Open Vehicle Routing Problem (OVRP)* , est un VRP à la différente près que les parcours des véhicules sont des chemins ouverts parce que les véhicules ne sont pas obligés de retourner au dépôt ou dans le cas où ils le sont, ils rebroussent chemin en revisitant les clients qui leurs sont affectés dans l'ordre inverse (Fu *et al*., 2003) (Eglese *et al*., 2005).

Le *Node, Edge, Arc, Routing Problem (NEARP)* est en fait la généralisation à plusieurs véhicules et à un graphe mixte d'un problème très important sur le plan

théorique, le *GRP* ou *General Routing Problem*. Le GRP consiste à construire une tournée de coût minimum visitant un sous-ensemble de noeuds et d'arêtes dans un graphe non orienté (Prins et Bouchenoua, 2004).

Bien sûr de nombreux articles de la littérature étudient des problèmes combinant plusieurs des variantes présentées ci-dessus, comme (Liu et Shen, 1999) par exemple qui considère une flotte hétérogène et des fenêtres temporelles. Etant donné le nombre important de combinaisons possibles, et donc de variantes différentes, les chercheurs ont développé de nombreuses approches de résolution pour les aborder, notamment des approches spécifiques permettant de faire le moins d'hypothèses simplificatrices possibles lors de la modélisation de problèmes rencontrés dans la réalité. La section 1.5 présente les diverses catégories d'approches de résolution qui figurent dans la littérature.

1.5 Les méthodes de résolution du VRP

Le VRP appartient au noyau de planification industrielle mais aussi à la grande famille des problèmes combinatoires NP-difficiles (NP-hard problem). Ceci signifie qu'il n'y a pas de méthode exacte pouvant résoudre les problèmes qui incluent un grand nombre de clients (> 100 clients) avec une durée de temps de calcul raisonnable (Zhu, 1999). Dans ce cas, il est nécessaire d'utiliser des méthodes approchées. Ainsi, les méthodes de résolution consacrées aux VRP et à ses variantes sont principalement classées en deux catégories : les méthodes exactes et les méthodes approchées. Ces dernières sont aussi divisées en deux sous groupes : les heuristiques et les métaheuristiques. La figure (1.2) présente la plupart de ces méthodes.
Pour plus de détails sur le VRP, le lecteur pourra également se reporter aux nombreux états de l'art qui lui sont consacrés comme (Laporte, 1992) (Fisher, 1995) (Laporte et Osman, 1995) (Gendreau *et al*., 1997) (Potvin et Thangiah, 1999) (Laporte *et al*., 2000) (Gendreau *et al*., 2002) (Toth et Vigo, 2002a) (Gendreau *et al*., 2003) (Bräysy *et al*., 2004a) (Bräysy et Gendreau, 2005b) (Gendreau *et al*., 2008) (Potvin, 2009).

1.5.1 Les méthodes exactes

Les méthodes exactes reposent sur l'utilisation d'algorithmes qui mènent de façon sûre vers la solution optimale. Néanmoins plusieurs méthodes ont été développées. Elles permettent généralement de résoudre efficacement des problèmes allant jusqu'à 50 clients, parfois plus. Par exemple, en 2003 une méthode résolvant un problème contenant 100 clients a été proposée dans (Ralphs, 2003). D'après (Laporte et Nobert , 1987), les méthodes exactes pour le VRP sont généralement de trois catégories selon qu'elles reposent sur un Branch and Bound, un Branch and Cut ou sur la programmation dynamique. Les sections suivantes présentent ces trois types de méthodes.

1.5.1.1 La procédure de séparation et d'évaluation (Branch and Bound)

Cette méthode appartient à la classe des méthodes de recherche arborescente. Elle a été proposée pour la première fois par (Land et Doig, 1960). Principalement, Elle

consiste en la construction d'un arbre de recherche représentant l'espace des solutions et élague des branches inutiles de cet arbre qui sont des branches contenant des solutions

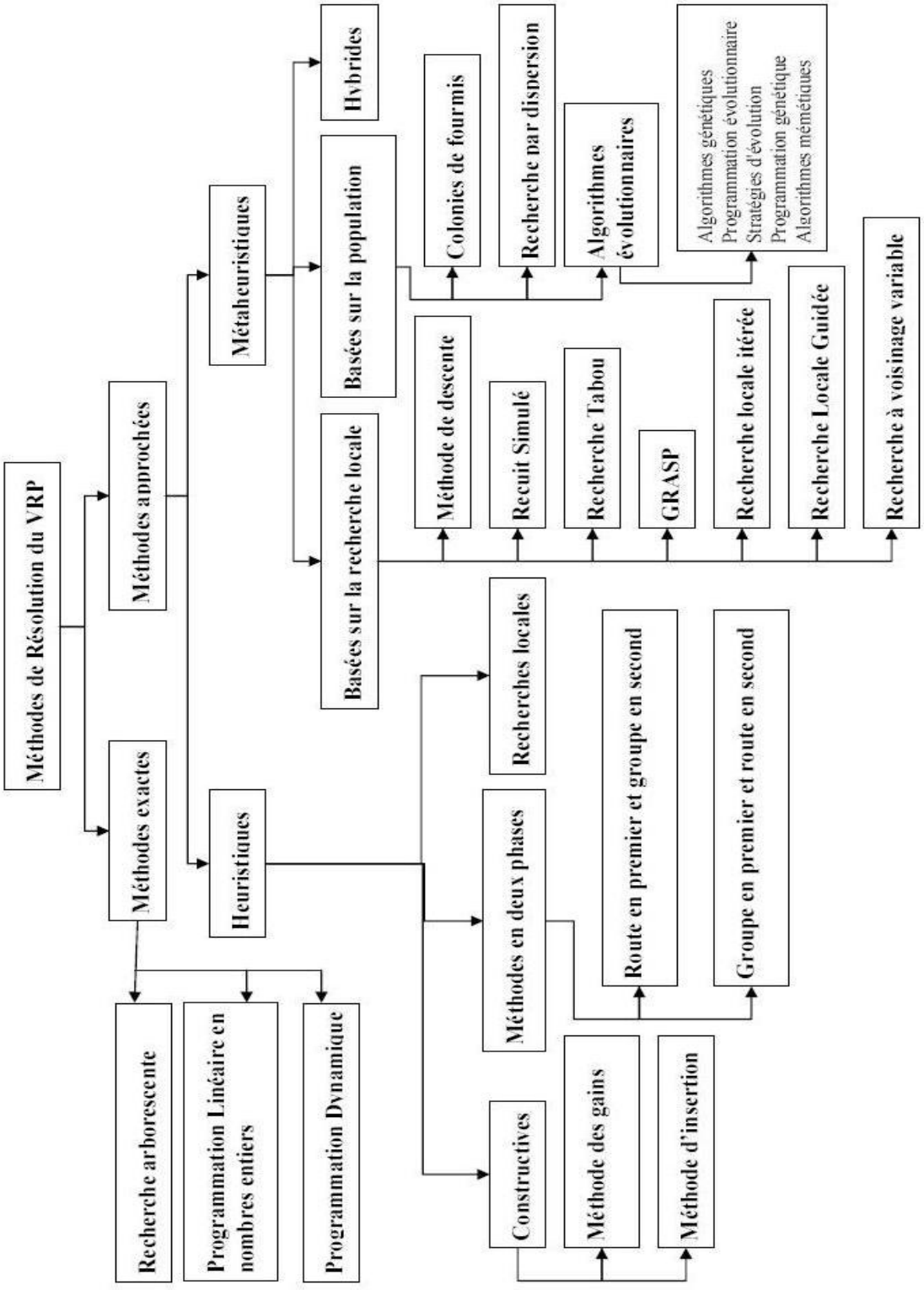


Fig. 1.2 – Les méthodes de résolution du VRP

non intéressantes ou non réalisables. Elle trouve les solutions exactes au VRP en construisant progressivement les routes arc par arc, mais en général la résolution nécessite un temps lié à la taille du problème. Elle donne de bons résultats pour des problèmes de petites tailles, mais dans le cas contraire, elle risque de générer des branches très étendues. L'exploration se fait avec des évaluations des branches et des comparaisons avec une borne ou une valeur seuil du critère à optimiser. La difficulté de ces méthodes est l'obtention de cette valeur de bonne qualité pour améliorer la phase d'élagage. Quelques méthodes comme la relaxation lagrangienne (Miller, 1995) donnent de meilleures bornes que les techniques de relaxation classiques (contrainte de capacité, contrainte de connectivité) (Laporte *et al*,. 1986).

1.5.1.2 Programmation linéaire en nombres entiers : Branch and Cut

Branch and Cut est une généralisation de Branch and Bound. Elle est utilisée lorsque le nombre de contraintes est trop élevé. Le terme de "Branch and Cut" a été introduit par (Padberg et Rinaldi, 1987) pour un algorithme de résolution du problème du voyageur de commerce (TSP). L'algorithme construit une arborescence nommée l'arbre du Branch and Cut. Les sous-problèmes qui forment l'arbre sont appelés des nœuds. Il existe trois types de nœuds dans l'arbre du Branch and Cut. Le nœud courant qui est en train d'être traité, les nœuds actifs qui sont dans la liste d'attente des problèmes et les nœuds inactifs qui ont été élagués au cours du déroulement de l'algorithme. Le principe est de partir d'une solution admissible entière du problème, et d'aller vers une autre solution admissible entière jusqu'à l'optimum (Bouzgarrou, 1998).

Un solveur de programmation linéaire est utilisé pour tenter de trouver une solution optimale entière qui respecte les contraintes du problème de Programmation Linéaire en Nombres Entiers (PLNE). Si celui-ci échoue, une phase de décomposition (i.e. Branch) du problème en 2 sous problèmes est nécessaire et la phase de coupes (Cut) est relancée sur les sous problèmes générés (Toth et Vigo, 2002b).

1.5.1.3 Programmation Dynamique

La programmation dynamique est un paradigme de conception qu'il est possible de voir comme une amélioration ou une adaptation de la méthode « diviser et régner ». Ce concept a été introduit par Bellman, dans les années 50, pour résoudre typiquement des problèmes d'optimisation. La programmation dynamique repose sur le principe d'optimalité « *toute politique optimale est composée de sous-politiques optimales* ». Elle a été appliquée pour la première fois au VRP par (Eilon *et al*., 1971). La programmation dynamique peut être appliquée dès lors qu'une solution optimale peut être représentée comme une combinaison de solutions optimales de sous-problèmes. Alors, la programmation dynamique résout chaque sous-problème une seule fois et stocke les solutions de tous les sous-problèmes rencontrés dans une matrice, pour éviter d'avoir à les recalculer par la suite. Enfin, elle cherche une relation de récurrence entre les sous-problèmes de sorte à résoudre le problème principal.

Cette méthode a été utilisée pour des problèmes de VRP de très petites tailles comme dans (Rego *et al*., 1994) pour la résolution de problèmes allant de 10 à 25 clients.

1.5.2 Les méthodes approchées

Les méthodes exactes rencontrent généralement des difficultés face à des applications de grande taille, et peuvent nécessiter un temps de calcul important même sur des instances de petite taille. C'est pour cette raison que l'utilisation des méthodes approchées s'est avérée d'une grande utilité. Ces méthodes permettent d'obtenir en temps raisonnable des solutions de bonne qualité pour des problèmes de grande taille mais sans garantie d'optimalité. On peut les subdiviser en deux classes : les heuristiques et les métaheuristiques.

1.5.2.1 Les heuristiques

Comme pour tous les problèmes NP-difficile, la recherche d'heuristiques performantes est l'une des principales préoccupations des spécialistes du domaine. Le principe d'une méthode heuristique est de trouver, en un temps raisonnable *une* solution de bonne qualité. Dans ce qui suit, nous décrirons les trois grandes familles d'heuristiques développées pour les problèmes de tournées : méthodes constructives, méthodes de recherches locales et méthodes en deux phases.

a. Méthodes Constructives

Les méthodes constructives construisent une seule solution progressivement, par une suite de choix partiels et définitifs et sans retours en arrière, c'est-à-dire qu'elles ne contiennent pas une phase d'amélioration. Parmi ces méthodes, nous pourrons citer la méthode des gains et la méthode d'insertion.

Méthode des gains (Clark and Wright)

La méthode de Clark et Wright a été proposée pour la première fois dans (Clarke et Wright, 1964) pour résoudre le CVRP avec un nombre variable de véhicules. Principalement, la méthode consiste à déterminer à chaque itération un couplage entre les sous-ensembles de la partition courante des clients (à gauche de la figure 1.3).

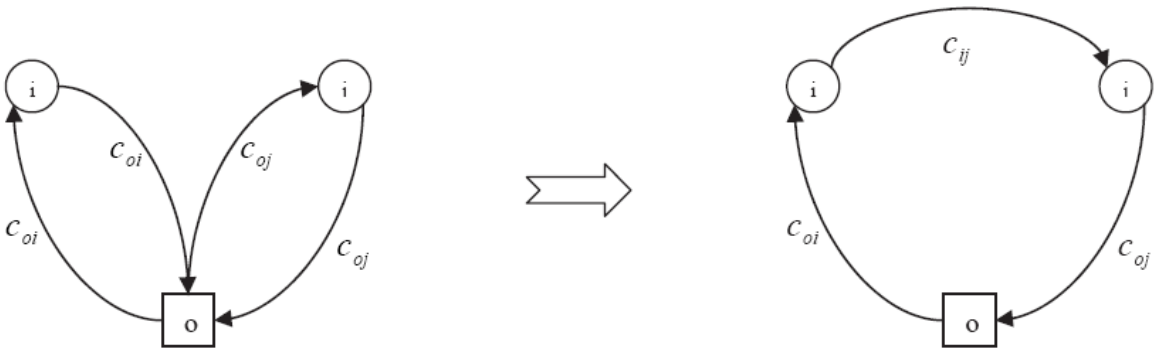


Fig. 1.3 – Méthode des gains

La méthode commence par créer une solution triviale dont laquelle chaque client est affecté à une tournée, et chaque tournée ne contient qu'un seul client. Puis les tournées

sont fusionnées deux à deux en fonction du plus grand gain (en terme de diminution du coût total des tournées). L'algorithme s'arrête, lorsqu'il n'y a plus de fusions améliorantes. Cette méthode a été utilisée par plusieurs chercheurs. Nous pourrons citer, par exemple (Gaskell, 1967) (Yellow, 1970) (Golden *et al*., 1977) (Nelson *et al*., 1985) (Paessens, 1988).

Méthode d'insertion

Les algorithmes d'insertion procèdent en deux phases ; la première phase sélectionne le nœud à insérer et la deuxième phase exécute l'opération d'insertion (Solomon, 87) (Toth et Vigo, 2002a). L'heuristique d'insertion la plus connue dans la littérature est *l'heuristique du plus proche voisin.* Cette heuristique sélectionne, à chaque itération, le plus proche nœud de la tournée en construction, puis l'insère à un endroit du cycle qui minimise l'augmentation du coût de la tournée. Ce processus est répété jusqu'à insertion de tous les nœuds. Une autre heuristique d'insertion est similaire au plus proche voisin sauf que le critère d'insertion est basé sur le choix, à chaque itération, des nœuds les plus éloignés de la tournée en construction. Cette heuristique est appelée *l'heuristique de la plus lointaine insertion*. Une dernière heuristique est *l'heuristique de la meilleure insertion*. Elle consiste à calculer pour chaque nœud non encore visité par la tournée en construction, le coût de la meilleure position d'insertion dans la tournée, et à exécuter l'insertion du nœud qui procure le meilleur gain.

b. Méthodes de recherches locales

Ces méthodes sont appelées aussi recherches de voisinage (neighborhood search) ou méthodes d'amélioration itérative. Pour expliquer ces méthodes, il faut avoir au préalable défini une notion de voisinage d'une solution :

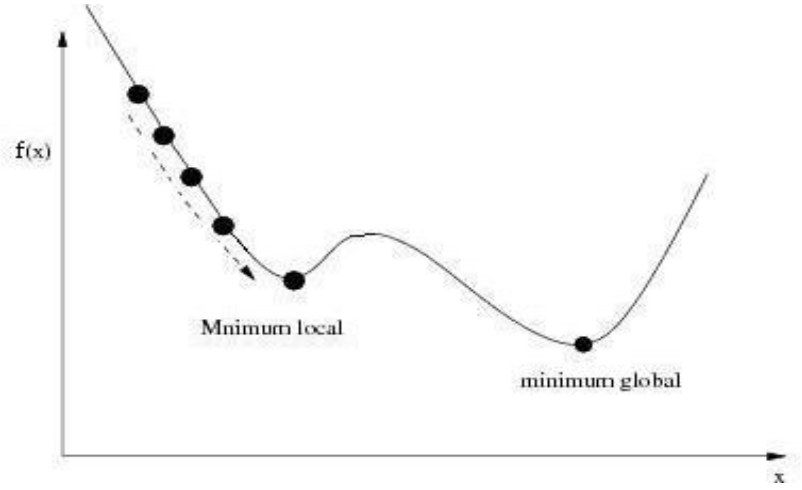


Fig. 1.4 – Illustration du principe des méthodes de voisinage.

Soit X un ensemble de solutions à un problème d'optimisation, figure (1.4). Soit f une fonction qui mesure la valeur f(x) de toute solution x dans X. Une structure de voisinage

(ou tout simplement un voisinage) est une fonction N qui associe un sous-ensemble de X à toute solution $x \in X$. Une solution $x' \in N(x)$ est dite voisine de x. Une solution $x \in X$ est un minimum local relativement à la structure de voisinage N si $f(x) \leq f(x')$ pour tout $x' \in N(x)$.Une solution $x \in X$ est un minimum global si $f(x) \leq f(x')$ pour tout $x' \in X$.

Dans (Siarry *et al.*, 2003), une technique de voisinage, nommée *chaîne d'éjections* a été proposée pour les problèmes de tournées. Elle consiste à déplacer un client d'une tournée à une autre, c'est-à-dire, procéder au transfert d'un client d'une première tournée vers une deuxième, cette dernière est modifiée en transférant un client vers une troisième tournée ; et ainsi de suite jusqu' à que le transfert arrive à une dernière tournée, qui elle même transfère un client vers la première tournée. Ce processus est illustré dans la figure (1.5).

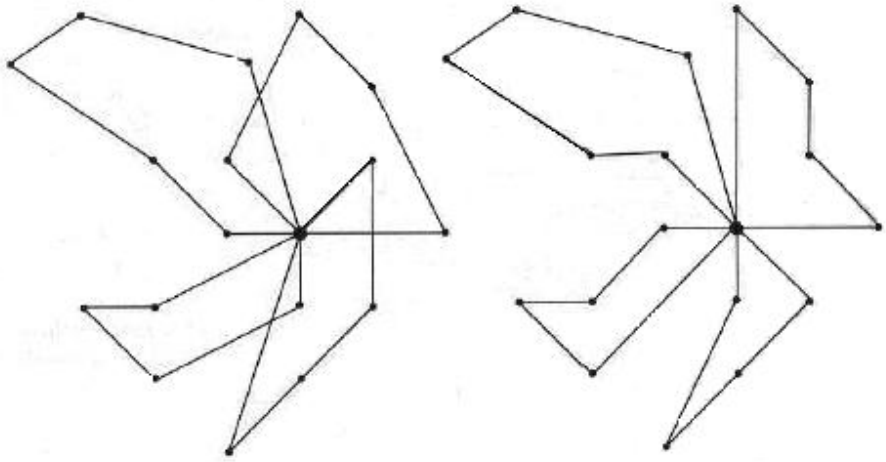


Fig. 1.5 – Illustration de la technique « chaîne d'éjections » (Siarry *et al.*, 2003)

Un autre type de voisinage, le plus connu pour les problèmes de tournées, est le voisinage de type λ – Opt (λ = 2, 3, 4..n) proposé par (Lin, 1965). Il consiste à supprimer λ arcs pour remettre les chaînes associées dans la meilleure combinaison possible. La figure (1.6) donne des exemples de 2-opt et 3-opt.

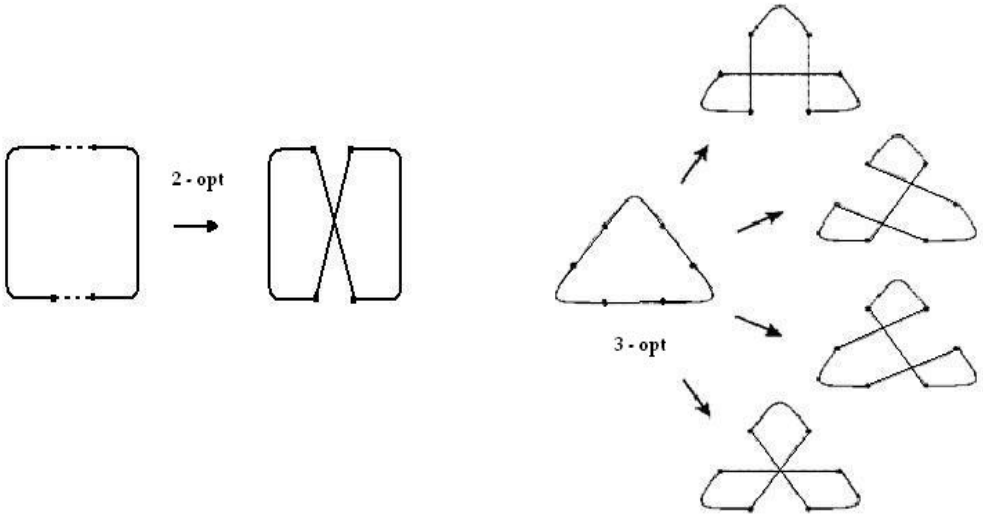


Fig. 1.6 – Exemples d'application des voisinages 2-opt et 3-opt

c. Méthodes en deux phases

Les méthodes en deux phases sont basées sur une décomposition du problème en une partition en sous-groupes de l'ensemble des clients et la détermination de la tournée associée à chaque sous-groupe. Deux classes d'algorithmes peuvent être considérées selon l'ordre dans lequel les deux phases sont effectuées : méthode de « route-first et cluster-second » et méthode de « cluster-first et route-second ».

La méthode « route-first et cluster-second »

Cette méthode a été initialement proposée par (Beasley, 1983). Le principe de cette heuristique est de construire une grande tournée pour le TSP, qui peut être vue comme un « tour géant » effectué par un véhicule de capacité infinie. Ce tour géant est ensuite découpé en tournées réalisables pour le VRP (Golden *et al.*, 1982).

La méthode de « cluster-first et route-second »

Cette méthode est une des heuristiques les plus connues. Elle est basée sur une approche géométrique qui permet de former des groupes de clients, puis des tournées à l'intérieur de ces groupes (Le Bouthillier, 2000). Plusieurs méthodes de ce type ont été proposées dans la littérature. Une de ces méthodes est *l'algorithme de balayage* (sweep) due à (Gillett et Miller, 1974), qui lui ont donné son nom bien que son principe peut remonter à (Wren, 1971) et (Wren et Holliday, 1972) pour le CVRP. Afin d'appliquer cet algorithme, il est préférable de représenter la localisation des clients par leurs coordonnées polaires (angle, longueur de rayon) dont l'origine est le dépôt central, voir la figure (1.7).

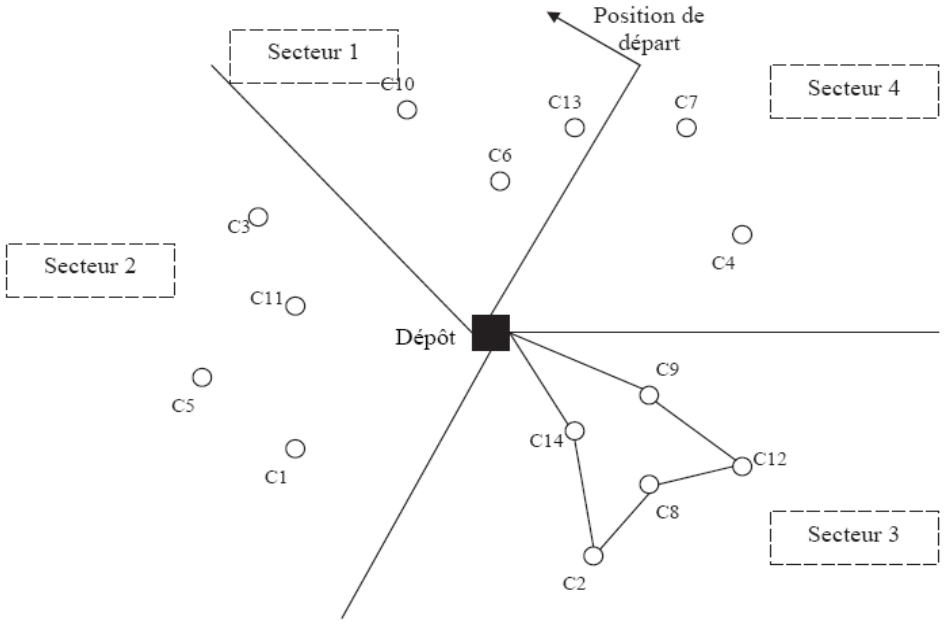


Fig. 1.7 – Illustration de l'algorithme de balayage (Gillett et Miller, 1974)

Dans la première phase, des « clusters » faisables sont créés en balayant un rayon centré autour du dépôt et en incluant des clients petit à petit dans un « cluster » tant que la capacité du véhicule n'est pas dépassée. Le processus est répété jusqu'à ce que toute la surface soit balayée et que tous les clients soient affectés. La deuxième phase consiste à obtenir l'ordre de parcours des clients, « une route », à l'intérieur de chaque « cluster ».

L'algorithme de balayage peut être étendu en générant un ensemble de « cluster » ou de groupes faisables, appelés *les pétales*, et puis en identifiant les ensembles conduisant à la meilleure solution en résolvant un problème de partition d'ensemble (*set partitioning problem*). Les premiers modèles de cette heuristique, appelé l'algorithme de 1-pétale, ont été proposés par (Foster et Ryan, 1976) et (Ryan *et al*., 1993). Ces modèles ont été étendus par (Renaud *et al*., 1996). Ces auteurs ont développé un algorithme appelé 2-pétales, où la recherche de meilleur ensemble de partition se fait sur des blocs de deux tournées. Cet algorithme de 2-pétales s'est montré comme le meilleur algorithme de la classe de méthodes de pétales.

Une autre heuristique de type « cluster-first et route-second » peut être trouvée dans (Fisher et Jaikumar, 1981). Elle consiste à créer des groupes en utilisant une méthode géométrique basée sur la division de la surface plane en K cônes selon les demandes de clients.

1.5.2.2 Les métaheuristiques

On appelle métaheuristiques des méthodes conçues pour échapper aux minima locaux. Une telle méthode a été proposée pour la première fois par (Glover, 1986). Le mot *métaheuristique* est dérivé de la composition de deux mots grecs : (*heuristique*) qui vient du verbe « heuriskein » et qui signifie 'trouver', et (*meta*) qui est un suffixe signifiant 'au-delà', ou 'dans un niveau supérieur' (Blum et Roli, 2008). Avant l'utilisation de ce nom, les métaheuristiques ont été souvent appelées *heuristiques modernes* (Reeves, 1993). Le but des métaheuristiques est similaire à celui des heuristiques : obtenir des solutions de bonne qualité en un temps raisonnable. Cependant, contrairement à une heuristique, le schéma général des métaheuristiques est totalement indépendant du problème à traiter. En plus, les métaheuristiques peuvent contenir des mécanismes qui permettent d'éviter le blocage dans un minimum local. Elles permettent ainsi d'explorer l'espace de recherche efficacement afin de donner de très bonnes solutions, mais aussi sans garantie d'optimalité.

Dans la littérature, plusieurs définitions ont été proposées pour le terme métaheuristique. La plus connue est une définition sur le site de Metaheuristics Network[2]: "Une métaheuristique est un ensemble de concepts qui peuvent être utilisés pour définir des méthodes heuristiques qui peuvent être appliquées à une large classe de problèmes différents. En d'autres mots, une métaheuristique peut être vu comme un cadre algorithmique général qui peut être appliqué sur des différents problèmes d'optimisation avec assez peu de modifications afin de les faire adaptées à un problème ".

Certaines métaheuristiques utilisent les concepts additionnels de diversification et d'intensification. Par diversification, on entend généralement une exploration assez large de l'espace de recherche, alors que le terme intensification vient plutôt mettre l'accent sur l'exploitation de l'information accumulée durant la recherche. Il est important de bien doser l'usage de ces deux ingrédients afin que l'exploration puisse

[2] http://www.metaheuristics.net

rapidement identifier des régions de l'espace de recherche qui contiennent des solutions de bonne qualité, sans perdre trop de temps à exploiter des régions moins prometteuses. Il existe de nombreux états de l'art sur l'application des métaheuristiques aux problèmes de tournées (Gendreau *et al*., 1997) (Potvin et Thangiah, 1999) (Laporte *et al*., 2000) (Gendreau *et al*., 2002) (Gendreau *et al*., 2003) (Bräysy *et al*., 2004a) (Bräysy *et al*., 2004b) (Bräysy et Gendreau, 2005b) (Gendreau *et al*., 2008). Les métaheuristiques peuvent être subdivisées en trois classes : métaheuristiques basées sur la recherche locale, métaheuristiques basées sur la population et métaheuristiques hybrides. Ces trois classes sont détaillées ci-dessous.

a. Métaheuristiques basées sur la recherche locale

Le but principal des métaheuristiques basées sur la recherche locale est que l'opérateur de voisinage échappe aux optimums locaux, figure (1.8).

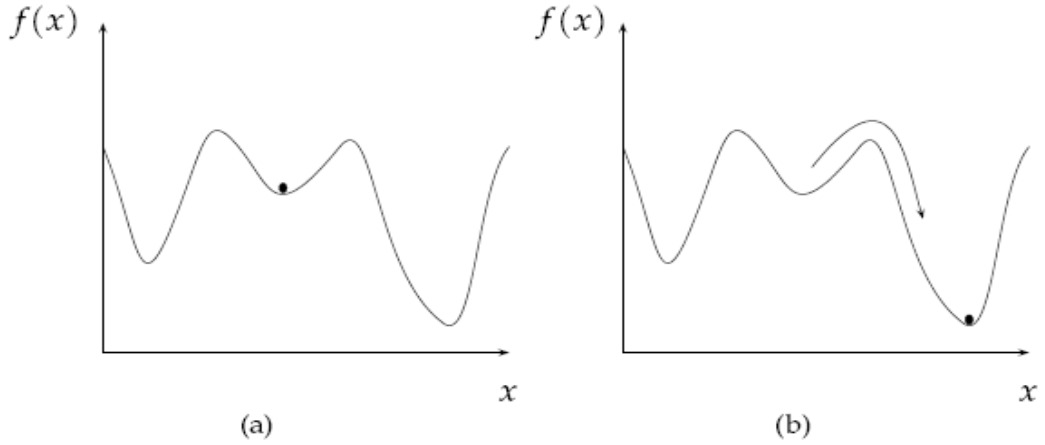


Fig. 1.8 – Illustration de la sortie d'un optimum local (Sörensen, 2003)

Ces métaheuristiques font appelle à la notion de voisinage qui est primordiale et le choix d'un opérateur de voisinage est un compromis entre efficacité et qualité. En d'autres termes, si l'opérateur de voisinage opère sur un espace large, on a de fortes chances de pouvoir échapper aux optimums locaux et de trouver l'optimum global en visitant une grande partie de l'espace des solutions ; mais cet opérateur peut être coûteux en temps. Par contre, si le voisinage est très restreint, l'opérateur de voisinage est alors moins coûteux en temps mais la probabilité d'être piégé dans un minimum local est élevée.

Dans la suite, nous présentons un ensemble de métaheuristiques fondées sur la notion de voisinage:

Méthode de descente

La méthode de recherche locale la plus élémentaire est la méthode de descente. Elle commence par une solution s, ensuite, à chaque itération et de manière aléatoire, elle construit une solution s' de N(s) (ou N est la fonction de voisinage). Soit f la mesure de la valeur de toute solution s. Si f(s') < f(s) alors s' devient la nouvelle solution courante s, et les

itérations se poursuivent. Par contre si f (s') > f(s) alors on construit de manière aléatoire une nouvelle solution s' de N(s) et on recommence cette opération tant qu'on n'a pas réussi à faire strictement mieux que f(s) et tant que le nombre de voisins de s visités n'a pas dépassé la taille du voisinage. L'avantage de cette méthode est qu'elle est stochastique, ce qui permet en l'appliquant plusieurs fois d'obtenir plusieurs minimums locaux dont certains pourront être différents et de retenir le meilleur.

Recuit Simulé (Simulated Annealing)

Les origines du recuit simulé remontent aux expériences réalisées (Metropolis *et al*., 1953) dans les années 50 dans la thermodynamique. Il est issu d'une analogie entre le phénomène physique de refroidissement lent d'un corps en fusion (recuit), qui le conduit à un état solide de basse énergie. Il faut abaisser lentement la température, en marquant des paliers suffisamment longs pour que le corps atteigne l'équilibre thermodynamique à chaque palier de température. Pour les matériaux, cette énergie minimale se manifeste par l'obtention d'une structure régulière (stable) comme dans les cristaux ou l'acier. (Kirkpatrick *et al*., 1983) et (Cerny , 1985) ont développé cette méthode pour les problèmes d'optimisation combinatoire. Des exemples d'utilisation de cette méthode pour résoudre le problème de VRP peuvent être trouvés dans (Rego *et al*., 1994)(Tan *et al.*, 2000) ou (Bent et Hentenryck, 2001). Avant d'expliquer la procédure du recuit simulé, on définit quelques paramètres utilisés. Soit s la solution courante, N(s) le voisinage de s, T une température qui agit comme un simple paramètre de contrôle, et soit r un nombre réel aléatoire dans [0,1]. La procédure générale du recuit simulé consiste en les étapes suivantes :

1. Choisir une solution courante $s \in S$ ainsi qu'une température initiale T;
2. Tant qu'aucun critère d'arrêt n'est pas satisfait faire;
3. Choisir aléatoirement un voisin $s' \in N(s)$;
4. Générer r aléatoirement;
5. Mettre à jour s selon la différence des valeurs de la fonction objectif $\Delta f = f(s') - f(s)$:

Si $\Delta f \leq 0$, alors s' est choisie comme solution courante (les solutions meilleures sont toujours acceptées).

Si $\Delta f > 0$ et si $e^{-\Delta f/t} \geq r$, alors s'est la nouvelle solution courante. Sinon, la transformation est rejetée et on conserve s pour l'itération suivante.

6. Mettre à jour T;

(La décroissance de la température se passe lentement au fur et à mesure jusqu'à atteindre une valeur proche de 0, ce qui signifie que la méthode n'acceptera plus de détériorer une solution).

7. Fin du tant que

La performance du recuit simulé dépend largement du choix de ses paramètres de contrôle, dont le réglage reste lui aussi très empirique. Les principaux paramètres de contrôle sont les suivants :

- la valeur initiale de la température : le rôle de la température T au cours du processus de recuit simulé est très important. En général, on choisit une température initiale suffisamment élevée qui donne une plus grande liberté pour

l'exploration de l'espace de recherche. Alors, on peut considérer une grande augmentation de la température comme un processus de diversification.

- la fonction de décroissance de la température : la performance du recuit simulé dépend largement du schéma de refroidissement utilisé. En fait, une forte décroissance de température risque de piéger l'algorithme dans un minimum local, alors qu'une faible décroissance au début du processus entraîne une convergence très lente de l'algorithme. Cette étape du processus de recuit simulé correspond à un processus d'intensification. Différents schémas de refroidissement ont été proposés. (Hao *et al*., 1999) classent ces schémas en trois catégories :
  - Réduction par paliers : chaque température est maintenue égale pendant un certain nombre d'itérations, et décroît ainsi progressivement.
  - Réduction continue : la température est modifiée à chaque itération (Lundy et Mees, 1986).
  - Réduction non-monotone: la température décroît à chaque itération avec des augmentations occasionnelles (Connolly, 1990).
- le critère de changement de palier de température,
- les critères d'arrêt : on peut choisir une limite liée au temps de calcul, ou à la température T qui doit atteindre la valeur zéro. En plus, si on améliore cette méthode qui est initialement sans mémoire en rajoutant une mémoire à long terme qui stocke la meilleure solution rencontrée, on peut stopper la recherche dès qu'un certain nombre d'itérations a été effectué sans amélioration de cette solution.

Recherche Tabou

Cette méthode a été élaborée par Glover vers1986. C'est une procédure itérative basée sur la notion de voisinage et sur l'enregistrement de statistiques sur les solutions visitées.

Dans une première phase, en partant d'une solution s, on examine complètement le voisinage N(s) de cette solution et on choisit la meilleure solution s', même si cela entraîne une augmentation de la fonction objectif que l'on veut minimiser [ou f(s')>f(s)]. Lorsqu'on atteint un minimum local s par rapport au voisinage N, la recherche tabou va donc se déplacer vers une solution s' plus mauvaise que s.

L'inconvénient est alors de revenir à s immédiatement si s $\in$ N(s') puisque s est meilleure que s'. En d'autres termes, on tourne en rond sur un ensemble de solutions. Plusieurs noms ont été donnés à ce phénomène, comme bouclage, cyclage ou blocage.

Pour remédier à ce problème, la méthode Tabou utilise une petite mémoire pour mémoriser les dernières solutions visitées et interdire tout déplacement vers une solution déjà explorée.

Ces solutions sont déclarées des solutions taboues, d'où le nom de la méthode. Elles sont stockées dans une liste de taille (longueur) donnée, appelée liste tabou. A chaque itération, la solution la plus ancienne de la liste tabou est remplacée par le dernier mouvement (ou le mouvement inverse) de la nouvelle solution qui n'est acceptée que si

elle n'appartient pas à liste tabou. Cette technique permet d'éviter le phénomène de bouclage, durant la visite d'un nombre de solutions au moins égal à la taille de la liste tabou.

Elle dirige l'exploration de la méthode vers des régions du domaine de solutions non encore visitées. Une taille de liste tabou trop petite risque de conduire au bouclage, alors qu'une grande taille peut interdire des mouvements intéressants qui nous auraient conduits vers de nouvelles solutions. Généralement, les règles pour déterminer la taille de la liste tabou sont divisées en deux : 1) statiques qui choisissent une valeur fixe de la taille tout au long de la recherche et 2) dynamiques qui font varier la valeur de la taille au cours de l'algorithme. (Glover *et al*., 1992) ont trouvé que les règles dynamiques sont plus robustes que les règles statiques. Plusieurs références ont utilisé la méthode tabou pour résoudre le VRP comme (Taillard, 1993) (Gendreau *et al*., 1994) (Xu *et al.*, 1996) (Rego *et al*., 1996) (Taillard *et al*., 1997) (Taillard *et al*., 1998) (Tan *et al.*, 2000).

Recherche à voisinage variable (Variable Neighbourhood Search ou VNS)

La méthode de recherche à voisinages variables a été proposée par (Mladenovic et Hansen, 1997). Une bonne introduction à la méthode VNS peut être trouvée dans (Hansen et Mladenovic, 1999). Cette méthode utilise méthodiquement plusieurs types de voisinages Nk (où k = 1,…, Kmax). Nk (s) représente l'ensemble de solutions dans le k^{er} voisinage de s. Le processus de cette méthode fonctionne comme décrit ci-dessous :

- Commencer par une solution initiale s, et poser k=1.
- Répéter les étapes 3, 4 et 5 jusqu'à ce que k = kmax
- Choisir aléatoirement une solution voisine s' dans Nk (s).
- Appliquer une procédure de recherche locale à s' afin d'obtenir une solution s".
- Si s" est meilleure que s, alors poser s = s" et k=1 pour concentrer la recherche autour de s". Sinon, élargir le voisinage en posant (k=k + 1).

L'idée de base de la méthode de VNS est d'utiliser plusieurs opérateurs de voisinage et de permettre ainsi de changer d'opérateur dès qu'il y a blocage dans un optimum local. Ceci permet de diversifier l'exploration de l'espace des solutions afin d'accéder à un plus grand nombre de régions intéressantes dans l'espace de recherche. Pour que cette méthode soit plus efficace, il est recommandé d'utiliser des structures de voisinage complémentaires, par exemple, des opérateurs de voisinages qui ne procèdent pas le même optimum local. Cette méthodes a été appliquée sur les problèmes de tournées comme par exemple dans (Polacek *et al*., 2004) (Fleszar *et al*., 2009)

La méthode GRASP

La méthode GRASP (Greedy Randomized Adaptive Search Procedure) est une procédure itérative introduite par (Feo et Resende, 1989). C'est une métaheuristique qui cherche à combiner les avantages des heuristiques constructives et des méthodes de voisinage. Elle est constituée de deux étapes : une étape constructive et une étape d'amélioration :

- En supposant qu'une solution est composée d'un ensemble de composantes, la phase constructive génère une solution pas à pas, en ajoutant à chaque itération une nouvelle composante à la solution partielle courante. La composante rajoutée est choisie aléatoirement dans une liste de candidats, notée RCL (pour *Restricted Candidate List*). Chaque composante est évaluée à l'aide d'un critère qui permet de mesurer le bénéfice qu'on peut espérer en rajoutant cette composante à la solution partielle. La liste de candidats contient les R meilleures composantes selon ce critère et cette liste est dynamiquement mise à jour après chaque itération.
- La deuxième phase d'amélioration est un processus de recherche locale qui peut être une méthode de descente ou une technique plus avancée telle que le recuit simulé ou la recherche tabou.

Les deux étapes ci-dessus sont répétées jusqu'à un critère d'arrêt et la meilleure solution est retournée à la fin.

Plusieurs références à GRASP ont été trouvées dans la littérature comme (Feo et Resende, 1995) (Pitsoulis et Resende, 2002) (Resende et Ribeiro, 2003) (Feo *et al*., 1994) (Kontoravdis et Bard, 1995) (Laguna *et al*., 1994) (Festa et Resende, 2002). Une autre source intéressante pour GRASP peut être trouvée sur un site de recherche[3].

Recherche Locale Guidée (Guided Local Search ou GLS)

La recherche locale guidée a été proposée par (Voudouris et Tsang, 1995). L'idée de base de la recherche locale guidée est de répéter des recherches locales en modifiant le coût des éléments caractérisant une solution en les pénalisant d'autant plus qu'ils sont fréquemment utilisés et que leur contribution à la dégradation de l'objectif est élevée. En d'autres termes, La recherche locale est guidée par des paramètres de pénalité. Si durant le processus de recherche, la recherche locale est piégée autour d'un minimum local, alors un changement des valeurs des paramètres de pénalité est appliqué, puis la recherche est de nouveau relancée.

La fréquence de la recherche locale et la mise à jour des paramètres de pénalisation peuvent être réitérés aussi souvent que cela est nécessaire pour régulariser la solution générée par la recherche. Cette méthode utilise la fonction d'objectif suivante pour une solution optimale locale s :

$$f'(s) = f(s) + \lambda \sum_{i=1}^{i=m} I_i(s)\, p_i$$

Où λ : paramètre de régularisation qui permet de faire varier l'importance du deuxième terme de cette fonction.
m : nombre d'éléments dans la solution s.
Pi : paramètre de pénalité notant l'importance d'avoir l'élément i dans la solution s.
Ii(s) : fonction indicatrice. Si l'élément i présent dans la solution s, alors Ii(s) = 1. Sinon Ii(s) = 0

La pénalité peut être vue comme une technique de diversification. Le but étant toujours de « s'échapper » d'un minimum local, pour explorer l'espace de recherche le plus largement possible et pour avoir davantage de chances de trouver l'optimum. La

[3] http://www.research.att.com/~mgcr

figure (1.9) présente l'approche GLS. Cette approche a été utilisée pour résoudre le problème du voyageur de commerce, voir (Voudouris et Tsang, 1999), et le problème de tournées de véhicules, voir (Kilby *et al.*, 1997).

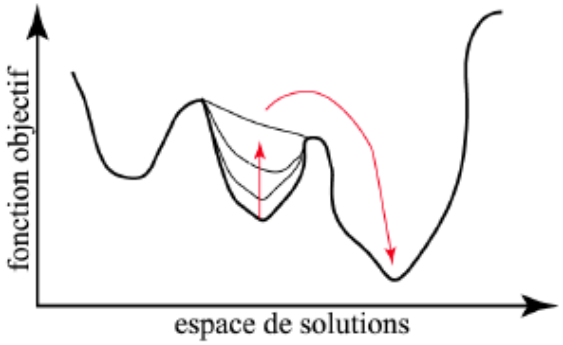


Fig. 1.9 – Illustration du principe de la recherche locale guidée

Recherche locale itérée (Iterated Local Search ou ILS)

Lorsque la procédure de descente est bloquée dans un optimum local, il existe plusieurs moyens de s'en échapper afin de relancer la recherche, sans nécessairement changer de voisinage. C'est le cas de la méthode de recherche locale itérée qui consiste à effectuer un mouvement aléatoire (perturbation) lorsqu'un optimum local est trouvé et de relancer par la suite la recherche locale, figure (1.10).

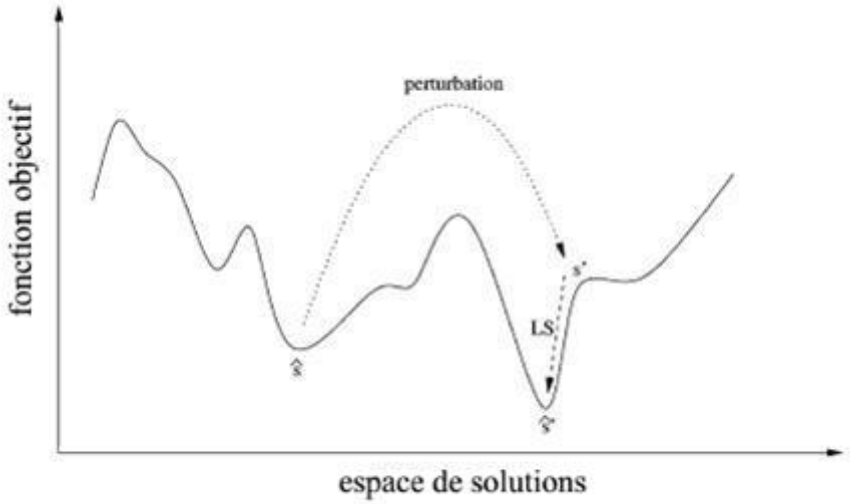


Fig. 1.10 – Illustration de la méthode de recherche locale itérée

Les perturbations sont très importantes. Elles doivent être effectuées de manière judicieuse, afin de minimiser les chances de retrouver l'optimum obtenu précédemment lors de la prochaine phase de descente. En effet, une perturbation trop petite peut être insuffisante et ne pas permettre de quitter l'optimum local atteint. D'un autre côté, si la perturbation est trop forte, la recherche locale est plus aléatoire.

Plusieurs méthodes se sont inspirées de ce principe (Baxter, 1981), tels que des chaînes de Markov (*large-step Markov chains*) (Martin et Otto, 1992), ou la descente réitérée (*iterated descent*) de (Baum, 1986). ILS a été appliqué avec succès aux problèmes de tournées dans (Martin et Otto 1992) (Johnson et McGeoch, 1997) (Besten *et al*., 2001) (Tounsi et Ouis, 2008) (Thierens, 2009) .

b. Métaheuristiques basées sur la population

Contrairement aux méthodes précédentes qui essayaient d'améliorer un unique "individu solution", les métaheuristiques basées sur la population travaillent explicitement avec une population de solutions. Le principe de base est que ces méthodes traitent une population de solutions globalement. Et à chaque itération, elles construisent une nouvelle population basée sur la précédente. En d'autres mots, elles évoluent à partir d'une population de solutions, en vue d'obtenir une solution ou un ensemble de solutions les plus adaptées. Cette évolution se fait à partir de transformations et de coopérations entre les individus qui représentent individuellement une solution de l'espace total de recherche du problème. Parmi ces méthodes, on peut essentiellement distinguer :

- les algorithmes évolutionnaires,
- les algorithmes de colonies de fourmis,
- la recherche par dispersion,

qui sont présentés de manière synthétique dans la suite.

Algorithmes évolutionnaires (AE* ou encore *evolutionary algorithms, EA)

Les algorithmes évolutionnaires sont sans doute les algorithmes basés sur la population les plus étudiés. Leur succès pour résoudre des problèmes d'optimisation difficiles dans divers domaines a promu le champ connu sous le nom de calcul évolutionnaire (*Evolutionary Computation* ou *EC*) (Bäck *et al.*, 1997). Généralement, les algorithmes évolutionnaires sont inspirés par des concepts issus de la théorie de l'évolution de Darwin. Ils sont des techniques d'optimisation itérative et stochastique, car ils utilisent itérativement des processus aléatoires. Ils font évoluer un ensemble de solutions (une population) à un problème donné en utilisant une fonction objectif afin de mesurer ses valeurs, dans l'optique de trouver les meilleurs résultats. La figure (1.11) montre un schéma général de fonctionnement d'un AE.

Une population de solutions du problème est d'abord initialisée, puis évaluée. Certaines solutions de la population sont ensuite sélectionnées pour former la population de parents. La sélection favorise les solutions ayant les meilleures valeurs de fonction objectif. Les parents sont ensuite recombinés et modifiés (phase de reproduction) pour produire une nouvelle population (enfants) en appliquant des opérateurs génétiques. Par cette phase, les enfants héritent de certaines propriétés de leurs parents. Ce schéma (évaluation, sélection puis reproduction) produit une évolution dans laquelle la qualité des solutions s'améliore d'itération en itération. Lors de la reproduction, les opérateurs génétiques perturbent les parents afin d'explorer l'espace de recherche. Il existe deux types principaux d'opérateurs génétiques :

croisement et mutation. L'opérateur de croisement crée des nouvelles solutions en combinant des parties de deux (ou plusieurs) solutions. L'opérateur de mutation est un opérateur unaire, qui crée de nouvelles solutions en modifiant l'une des solutions de la population. Les enfants sont évalués (on met à jour leurs valeurs de fonction objectif). Enfin, un sous ensemble de solutions est choisi parmi les parents et les enfants, pour remplacer la population courante par une nouvelle population pour la génération suivante (phase de remplacement). Ce processus est répété jusqu'à ce qu'une condition d'arrêt soit satisfaite. L'algorithme retourne la (ou les) meilleure(s) solution(s) qu'il a identifiée(s), qui est supposée être une solution proche de l'optimale ou optimale.

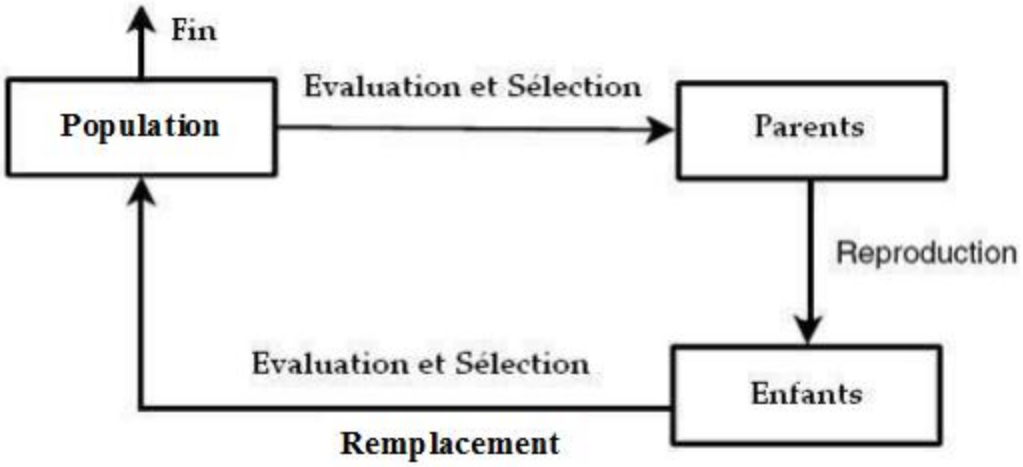


Fig. 1.11 – Fonctionnement général d'un algorithme évolutionnaire

De nombreux état de l'art et études concernent les algorithmes évolutionnaires, par exemple (Bäck, 1996) (Fogel, 1994) (Spears *et al.*, 1993) (Michalewicz et Michalewicz, 1997) (Calégary *et al.*, 1999). Fondamentalement, quatre types d' algorithmes ont été identifiés comme faisant partie de la famille des algorithmes évolutionnaires : la programmation évolutionnaire (PE) développée par (Fogel, 1962) (Fogel *et al.*, 1966), les stratégies évolutionnaires (SE) proposées par (Rechenberg, 1965) (Rechenberg, 1973) (pour plus d'information voir également (Schwefel, 1984)) , les algorithmes génétiques (AG) proposés par (Holland,1975) (voir aussi (Goldberg, 1989) (Michalewicz et Michalewicz, 1997) (Vose, 1999) (Reeves et Rowe, 2002)et la programmation génétique (PG) (Koza, 1992) (Schoenauer et Michalewicz, 1997) propose une comparaison entre ces différents algorithmes évolutionnaires. Les paragraphes suivants présentent rapidement ces quatre classes.

Diverses classes d'algorithmes évolutionnaires

Algorithmes génétiques (AG ou encore *Genetic Algorithms ou GA)*

Les algorithmes génétiques sont les plus populaires des algorithmes évolutionnaires. Ils ont été exposés par (Holland 1975) puis vulgarisés par (Goldberg, 1989). Ces

algorithmes s'inspirent des principes de la génétique (la survie des individus les mieux adaptés à un environnement, la recombinaison génétique, quelquefois l'apparition d'une mutation). Pour cette raison, ces algorithmes accordent une grande importance à la distinction entre la représentation génétique d'un individu (génotype) et sa représentation réelle (phénotype).

Le principe de base des algorithmes génétiques est le suivant : "A chaque génération, un nouvel ensemble de créatures artificielles est créé en utilisant des parties des meilleures solutions précédentes avec éventuellement des parties innovatrices" (Goldberg, 1989).

Dans ce qui suit, on présente un schéma général d'algorithme génétique :

1. Créer la population initiale (de taille N)
2. Tant que le critère d'arrêt n'est pas vérifié faire
a. évaluation de chaque individu de la population
b. sélection proportionnelle des parents (par couples d'individus ou par groupes plus larges) compte tenu de leurs évaluations (fitness).
c. reproduction :
- combiner les parents afin de produire une (ou plusieurs) nouvelle(s) solution(s); (croisement)
- appliquer un opérateur génétique qui crée des enfants dont une partie des gènes sont aléatoires; (mutation)
d. remplacement : sélectionner N solutions pour constituer une nouvelle génération de taille N.
3. Fin du tant que

L'opérateur principal utilisé par cet algorithme est l'opérateur de recombinaison (appelé aussi croisement). Cet opérateur récupère des parties du génotype de deux ou plusieurs solutions (parents), qu'il combine pour construire un (ou plusieurs) nouveau(x) génotype(s) (enfant(s)), héritant ainsi de certaines de leurs caractéristiques. Mais l'utilisation du croisement seul ne permet pas d'introduire du matériel génétique nouveau, car cet opérateur combine le matériel déjà présent dans la population. Ainsi, les algorithmes génétiques utilisent un autre opérateur (la mutation) comme opérateur secondaire permettant de remédier à ce problème et d'introduire de nouveaux gènes, inexistants dans la population.

Pour la mise en œuvre et pour régir l'efficacité d'un algorithme génétique, il faut déterminer:

- la méthode de génération de solutions initiales,
- la taille de la population,
- le codage des solutions,
- la fonction de fitness, qui représente la qualité d'une solution vis à vis de la fonction objectif,
- l'opérateur de sélection des solutions à recombiner,
- l'opérateur de croisement,
- l'opérateur de mutation,

- l'opérateur de sélection des solutions pour remplacement,
- le critère d'arrêt.

L'un des défauts les plus fréquents de cette méthode est la convergence prématurée vers une population de mauvaise qualité. Une introduction aux algorithmes génétiques peut être trouvée dans (Goldberg, 1989) avec quelques preuves de convergence. Les AG ont été appliqués par exemple au TSP dans (Potvin, 1996) (Zhao *et al*., 2009), au VRP dans (Baker et Ayechew, 2003) (Prins, 2004) (Zhan-feng *et al*., 2009), au VRP avec fenêtres de temps (Blanton et Wainwright, 1993) (Thangiah *et al*., 1993) (Thangiah, 1995) (Potvin et Bengio, 1996) et au VRP avec contraintes de précédence (Potvin *et al.*, 1996). Ces méthodes ont été utilisées dans notre étude et le chapitre 3 fera une présentation plus approfondie des algorithmes génétiques appliqués dans ce cadre.

Programmation évolutionnaire (PE ou encore *Evolutionary Programming ou EP)*

La programmation évolutionnaire a été développée par (Fogel, 1962) (Fogel *et al*., 1966), et ensuite étendue par (Burgin, 1973) (Atmar, 1976). Dans cette méthode, il n'y a aucune contrainte sur la représentation des solutions alors que c'est le cas dans l'algorithme génétique. Ici, après la création de la population initiale, la programmation évolutionnaire applique directement l'opérateur de mutation et n'utilise aucun opérateur de croisement (Bäck *et al.*, 1993) en tant qu'opérateur génétique. Le mécanisme de choix de parent est déterministe. Le processus de sélection des survivants (remplacement) est probabiliste et basé sur un tournoi stochastique (Eiben et Smith, 2003). Cette classe d'algorithmes évolutionnaires est la moins utilisé parmi les membres de la famille des AE à cause de sa similitude avec les stratégies d'évolution (voir paragraphe suivant). Les différences principales entre la programmation évolutionnaire et les stratégies d'évolution sont présentées dans le prochain paragraphe. Une excellente explication de PE est donnée dans (Fogel *et al.*, 1966) (Fogel, 1995).

Stratégies d'évolution (SE ou encore *Evolution Strategies ou ES)*

Dans les stratégies d'évolution, introduites par (Rechenberg, 1965) (Rechenberg, 1973), il y a une différence entre la taille de la population des parents (μ) et celle de la population des enfants ($\lambda \geq \mu$). Le premier algorithme basé sur les stratégies d'évolution (Rechenberg, 1973) (Schwefel, 1981) travaille sur une population composée d'un seul individu, représenté par un vecteur de nombres réels. Plus récemment, les stratégies d'évolution reposent sur une population de plusieurs solutions. Cette méthode, comme la programmation évolutionnaire, utilise une représentation réelle des solutions (elles-mêmes) contrairement aux algorithmes génétiques. En fait, la stratégie d'évolution de base génère λ enfants en faisant subir une mutation à μ parents en utilisant une mutation spécifique basée sur une distribution gaussienne. Classiquement, elle ajoute une valeur aléatoire, tirée au sein d'une distribution normale, aux valeurs réelles du codage. Puis elle utilise une sélection de remplacement élitiste: les parents sont remplacés (de manière déterministe) par les μ meilleurs individus.

Ces derniers appartiennent soit à la population des enfants uniquement, et dans ce cas, elle est présenté sous le sigle (μ, λ) −ES), soit aux populations des parents et des enfants, et elle est alors appelée (μ+λ) −ES. Le croisement est rarement utilisé et s'il

l'est, il ne sera qu'un opérateur secondaire. Les deux différences principales entre les stratégies d'évolution et la programmation génétique sont donc :

1. la sélection de remplacement : typiquement, la programmation génétique utilise une sélection de tournoi stochastique alors que les stratégies d'évolution utilisent une sélection élitiste déterministe,
2. l'opérateur de croisement : aucun mécanisme de croisement n'est utilisé dans la programmation génétique. En revanche, quelques formes de croisement ont été mises en application comme opérateurs secondaires dans les stratégies d'évolution.

Plus d'informations sur les stratégies d'évolution peuvent être trouvées dans (Schwefel, 1987) (Kursawe, 1992) (Kursawe, 1993) (Schwefel et Rudolph, 1995). (Mester et Bräysy, 2007) (Repoussis *et al*., 2010) présentent également des applications de ces méthodes au VRP.

Programmation génétique (PG ou encore *Genetic Programming ou GP)*

La programmation génétique est une approche spécialisée des algorithmes génétiques. Elle a été développée par (Koza, 1992). Ce dernier a traité beaucoup de problèmes d'intelligence artificielle. Il a appliqué la programmation génétique comme un sous-domaine des algorithmes génétiques sur une population de programmes informatiques. La différence fondamentale entre la programmation génétique et l'algorithmique génétique est la représentation des individus. Ceux-ci sont des programmes écrits sous forme d'arbres non linéaires dans la programmation génétique. Comme dans l'algorithme génétique, après création de la population initiale et évaluation des solutions, les opérateurs génétiques de croisement et mutation sont appliqués, sous forme d'opérateurs propres à la représentation des solutions utilisée. L'opérateur de croisement est basé sur l'échange de sous-arbres entre deux arbres (parents) tandis que la mutation est basée sur un changement aléatoire dans un arbre, comme par exemple choisir un nœud d'un arbre et remplacer la branche dont il est la racine par une autre branche générée aléatoirement (Ferreira, 2006). Ensuite, les meilleures solutions sont choisies pour former la population suivante. Ce processus est répété jusqu'à la satisfaction d'un critère d'arrêt.

Algorithmes mémétiques

Le terme « algorithme mémétique » trouve son origine dans le concept de « meme » qui est une unité d'information qui évolue avec les échanges d'idées entre individus (The Selfish Gene) (Dawkins, 1989). (Moscato, 1989) a inspiré l'idée de ces algorithmes à partir de certains modèles d'adaptation dans la nature, qui combinent l'évolution adaptative de populations d'individus avec l'apprentissage des individus au cours de leur vie.

Principalement, les algorithmes mémétiques sont des extensions des algorithmes évolutionnaires utilisant des procédures de recherche locale pour améliorer leurs individus. Il existe plusieurs possibilités de combiner ces deux types de méthodes. Souvent, l'algorithme mémétique travaille comme l'algorithme génétique et débute avec la génération d'une population initiale. Chaque individu est évalué. Ensuite,

l'algorithme répète itérativement les phases de croisement (pour obtenir la population des enfants) et de recherche locale appliquée à chaque enfant (pour l'améliorer). Les meilleures solutions parmi la population courante et la population d'enfants sont transmises par élitisme à la nouvelle population. Ce processus est répété jusqu'à ce que la condition d'arrêt soit vérifiée. Donc, l'opérateur de recherche locale a remplacé ou a succédé à la mutation, et permet d'intensifier la recherche dans diverses zones pointées par les mécanismes génétiques (sélections, croisement). (Radcliffe et Surry, 1994) et (Ozdemir, 2002) proposent une autre possibilité de combinaison entre algorithme génétique et recherche locale : l'algorithme mémétique applique une recherche locale à chaque solution avant qu'elle soit inclue dans la population de l'algorithme génétique. (Merz et Freisleben, 1997) ont également utilisé cette approche pour résoudre le TSP. Ils l'ont appelée Recherche Locale Génétique (Genetic Local Search ou GLS). A nouveau, une recherche locale est appliquée sur les solutions obtenues par application d'un opérateur génétique. Par conséquent, toutes les solutions dans la population représentent des minima locaux. Plusieurs noms ont été utilisés dans la littérature pour présenter cette méthode, comme : algorithmes génétiques hybrides (Fleurent et Ferland, 1994), recherche locale génétique (Ulder *et al*., 1991) (Kolen et Pesch, 1994) (Merz, 2000), algorithmes évolutionnaires Baldwiniens (Ku et Mak, 1998), algorithmes évolutionnaires Lamarkiens (Morris *et al*., 1998).

Algorithmes de colonies de fourmis

Les colonies de fourmis sont basées sur le comportement réel des fourmis à la recherche de nourriture. Ce comportement, décrit dans (Deneubourg *et al.*, 1990), leur permet de trouver les plus courts chemins entre les sources de nourriture et leur nid. Ceci est possible grâce à la trace chimique (appelée phéromone) qui est une substance que chacune des fourmis dépose sur le sol. Lorsqu'une fourmi doit choisir entre deux directions, la bonne route est avec une plus grande probabilité celle comportant une plus forte concentration de phéromone. Les fourmis qui empruntent les sentiers les plus courts arrivent rapidement à se procurer de la nourriture et rentrent au nid en déposant de la phéromone sur leur chemin de retour. De plus, ces phéromones s'évaporent au cours du temps. Cela efface progressivement les traces les moins fréquentées. Notons que lorsque la trace de phéromone est trop faible, l'exploration des fourmis devient complètement aléatoire. En s'inspirant du comportement ci-dessus, une famille d'algorithmes de colonies de fourmis a été proposée par (Dorigo, 1992), puis (Dorigo *et al.*, 1996). L'idée de base des algorithmes de colonies de fourmis consiste à travailler sur une population de solutions. Chaque fourmi se déplace sur l'espace de recherche. Une structure de donnée commune, partagée par toutes les fourmis, contient l'information sur la phéromone accumulée dans cet espace. Les fourmis marquent les meilleures solutions, et tiennent compte des marquages précédents pour optimiser leur recherche.

Pour plus de compréhension, on donne un exemple du premier algorithme de colonies de fourmis proposé. Il est appelé le système de colonie de fourmis (*Ant Colony System* ou *ACS*). Il vise notamment à résoudre le TSP. Il repose sur un ensemble de fourmis, chacune parcourant un trajet parmi ceux possibles. Parallèlement pour chacune des fourmis, et jusqu'à satisfaction d'un critère d'arrêt, les étapes suivantes sont répétées itérativement:

- construire une nouvelle solution : une fourmi choisit un trajet, et trace une piste de phéromone,
- l'ensemble des fourmis parcourt un certain nombre de trajets, chaque fourmi déposant une quantité de phéromone proportionnelle à la qualité du parcours,
- évaluer la qualité de la solution : chaque arête du meilleur chemin est plus renforcée que les autres,
- mettre à jour les traces : l'évaporation fait disparaître les mauvaises solutions.

Figure (1.12), qui est pris d'un site de l'encyclopédie des fournis[4], illustre ces étapes. Généralement, cet algorithme peut être amélioré de diverses manières. Des extensions du système de colonie de fourmis décrit ci-dessus sont regroupées sous le terme d'Optimisation par Colonies de Fourmis (*Ant Colony Optimisation* ou *ACO*) (Dorigo *et al.*, 1999). Ce type d'extension consiste à réaliser une recherche locale sur chaque solution produite par l'algorithme constructif du système de fourmis. Après l'étape 1 de l'algorithme précédent, on ajoute l'application d'une recherche locale sur chacune des solutions.

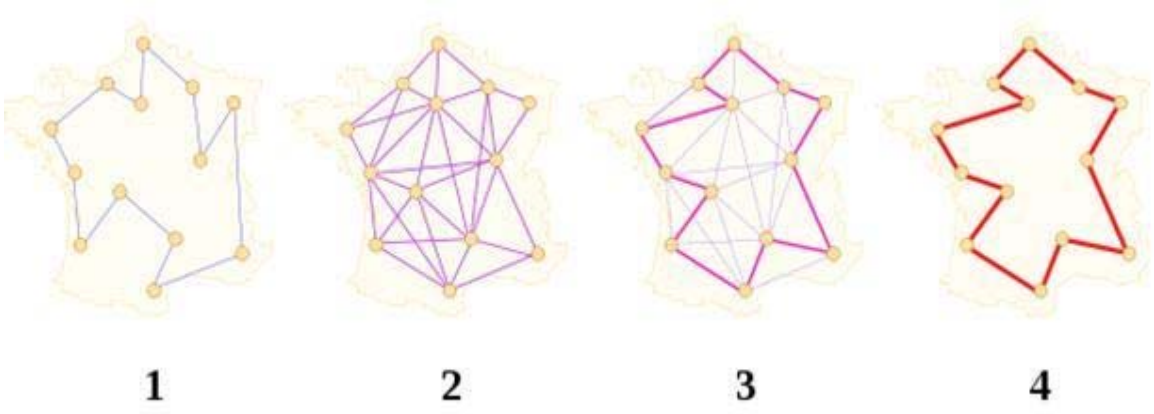


Fig. 1.12 – Illustration du fonctionnement d'un algorithme de colonies de fourmis

Une autre extension efficace du système de colonie de fourmis est le *Max-Min Ant System* ou *MMAS*, où seules les meilleures fourmis tracent des pistes et où le dépôt de phéromones est limité par une borne supérieure (empêchant une piste d'être trop renforcée) et une borne inférieure (laissant la possibilité d'être explorée à n'importe quelle solution). Cet algorithme atteint de meilleurs résultats que l'original, et évite notamment une convergence prématurée. Plusieurs références appliquent ce groupe de métaheuristiques pour le VRP, parmi lesquelles (Stützle et Hoos, 2000) (Gambardella *et al*., 2003) (Wade et Salhi, 2004) (Gajpal et Abad, 2009) (Yang et Zhuang, 2010) et pour les état de l'art et livres (Dorigo et Stützle, 2004) (Rizzoli *et al.*, 2007) (Yu *et al.*,2009) (Zhang et Tang, 2009) .

[4] http://www.kimonte.com/ algorithme_de_colonies_de_fourmis.php

Recherche par dispersion (Scatter Search ou SS)

La recherche par dispersion, ou « Scatter Search » en anglais, a été proposée par (Glover, 1977). Elle consiste à générer un ensemble de solutions dispersées de bonne qualité à partir d'un ensemble de référence. D'abord, de nouvelles solutions candidate sont créées par combinaisons linéaires d'un nombre variable de solutions de cet ensemble. Les solutions créées ne sont pas forcément des solutions admissibles. C'est pourquoi, une procédure de réparation est appliquée sur chaque candidat pour obtenir un ensemble de solutions admissibles. Ensuite, une recherche locale est appliquée sur ce dernier. Elle sert d'une part à intensifier la recherche et d'autre part à obtenir des solutions dispersées. Notons qu'au sein de cette méthode c'est une procédure de combinaison entre solutions qui réalise la diversification de la recherche. On retrouve donc dans cette méthode une génération de nouvelles solutions similaire à celle utilisée dans les algorithmes génétiques. Enfin, le nouvel ensemble de solutions de référence pour l'itération suivante est sélectionné à partir des solutions dispersées générées et de l'ensemble précédent de référence. Ces étapes sont répétées jusqu'à satisfaction d'un critère d'arrêt. Une description plus détaillée de cette métaheuristique est donnée dans (Glover, 1998) (Glover *et al*., 2000) (Laguna, 2002). De plus, des exemples d'application de cette approche au VRP sont disponibles dans (Scheuerer et Wendolsky, 2006) (Tang *et al.*, 2010).

c. Métaheuristiques hybrides

La tendance est actuellement d'avoir recours à des méthodes hybrides. Il existe plusieurs types possibles d'hybridations. Le mode d'hybridation qui semble le plus fécond concerne la combinaison entre les méthodes métaheuristiques basées sur la recherche locale et les méthodes métaheuristiques basées sur la population. L'idée essentielle de cette hybridation consiste à exploiter pleinement la puissance des deux types de méthodes par application de la recherche locale aux solutions dans une population avant de les combiner. Par exemple, une méthode évolutive peut détecter de bonnes régions dans l'espace de recherche, alors qu'une recherche locale explore efficacement les régions prometteuses en limitant le risque de passer à côté d'une solution optimale sans la voir. L'hybridation peut aussi désigner la combinaison des méthodes métaheuristiques avec des méthodes exactes. Par exemple, une métaheuristique peut fournir des bornes à une méthode de type Branch and Bound. Il est également possible d'hybrider des méthodes heuristiques et des méthodes métaheuristiques comme par exemple la méthode GRASP qui hybride des heuristiques constructives et des méthodes de voisinage.

La combinaison augmente la puissance de ces méthodes. Malheureusement, les temps de calcul nécessaires peuvent devenir prohibitifs à cause du nombre d'individus manipulés dans la population. Pour résoudre ce problème, on peut appliquer la parallélisation de ces algorithmes sur des machines parallèles (Verhoeven et Aarts, 1995) (Talbi *et al.*, 1998).

Généralement, les méthodes hybrides ont permis de produire de bons résultats. Quelques exemples de métaheuristiques hybrides peuvent être trouvés dans (Jog et Gucht, 1987) (Mülhenbein *et al.*, 1988) (Jog *et al.*, 1991) (Ulder et al., 1991) (Freisleben et Merz, 1996) pour le problème du voyageur de commerce. Il existe

également des approches de ce type pour le problème de tournée de véhicules (Prins, 2004) (Erbao et Mingyong, 2009) (Zhen et Zhang, 2009). Pour plus d'informations sur cette classe de métaheuristiques, voir (Talbi, 2002) qui en donne une description détaillée et une classification.

1.6 Conclusion

Ce chapitre a démontré les enjeux économiques importants liés au VRP. Tout d'abord, il intervient dans de nombreux domaines de la planification industrielle. Pour la production de biens, il intervient pour le transport des produits au sein des ateliers de production, mais également dans tous les maillons de l'ensemble de la chaine logistique, depuis les systèmes de collecte chez les fournisseurs jusqu'à la distribution chez les clients. Mais les modèles de tournées sont aussi utilisés pour résoudre des problèmes liés à différents processus de fabrication, comme l'optimisation des temps de découpe, celle du temps de placement et montage de composants électroniques ou la minimisation de temps de changements d'outils. En production de services, les problèmes de tournées interviennent pour le transport de marchandises, comme la collecte d'ordures ménagères, et pour le transport de personnes, tel que les ramassages scolaires, le transport à la demande, l'organisation de soins à domicile.

Cela a suscité une très forte activité de recherche sur ce sujet. Ceci a provoqué une multiplication des variantes du problème résolues. Pour représenter les différents cas rencontrés sur le terrain, les chercheurs ont ajoutés au problème de tournées de base une grande variété d'hypothèses, contraintes ou objectifs. Ces derniers peuvent porter sur les clients à desservir (comme les fenêtres temporelles à respecter chez les clients et la minimisation des retards associés), les véhicules (pour représenter une flotte hétérogène, dont les véhicules sont de capacité variable par exemple, ou prendre en compte les contraintes liées à des véhicules multi-compartiments) ou encore sur le réseau sur lequel les véhicules transitent (comme les sens uniques).

Les problèmes de tournées étaient déjà difficiles à résoudre dans des versions simples, suscitant le développement de méthodes approchées pour les cas incluant une centaine de clients. L'ajout de contraintes supplémentaires augmente encore leur complexité, ce qui a provoqué la proposition de très nombreuses heuristiques et métaheuristiques pour les résoudre.

De ce fait, la littérature scientifique est très volumineuse, et diversifiée. L'ampleur des études bibliographiques à mener par les jeunes chercheurs découvrant ce domaine est très grande. L'identification des variantes de problèmes de tournées par sigles, dans lesquels divers préfixes et suffixes indiquent les contraintes supplémentaires prises en compte dans chaque variante montre alors ses limites. Premièrement, il est difficile d'identifier quelle variante correspond au problème réel étudié. Ensuite, il faut un travail fastidieux pour déterminer si des approches ont déjà été proposées pour ce cas, et identifier toutes les références bibliographiques correspondantes lorsqu'il en existe. En particulier, en absence de référentiel commun, les auteurs n'utilisent pas forcément le même sigle pour désigner des cas identiques ou quasi identiques. Enfin, la comparaison des approches en termes de performances est ardue car il est nécessaire de détailler beaucoup d'articles pour identifier ceux qui traitent des variantes suffisamment

similaires pour être comparables. Le chapitre suivant illustre les incohérences auxquelles peut conduire cette notation par sigle. Il propose une notation pour pallier cet inconvénient, et illustre les divers apports qu'elle peut générer.

Chapitre 2

Nouvelle notation pour classifier les problèmes de tournées de véhicules et applications

Sommaire

2.1 Introduction

Pour distinguer les variantes de problèmes de tournées de véhicules entre elles, les auteurs utilisent généralement des sigles dans lesquels divers préfixes et suffixes indiquent la présence de différentes hypothèses ou contraintes. La multiplication des variantes abordées et leur complexité grandissante ont multiplié le nombre et la longueur des sigles utilisés. Ce système d'identification montre alors ses limites. Comme il n'existe aucune norme commune, et par conséquent aucun contrôle de la pertinence et de la cohérence des sigles utilisés, des redondances et/ou des incompatibilités apparaissent. De nouveaux suffixes ou préfixes sont souvent créés selon les besoins, par chacun des auteurs, sans qu'aucune liste des sigles existants ne soit tenue à jour. Selon les auteurs, deux variantes différentes de problème peuvent être désignées par le même sigle ou au contraire des sigles distincts peuvent être affectés à deux problèmes strictement identiques. Cette divergence de points de vue entre auteurs conduit à l'émergence de définitions et de sigles contradictoires, ce qui rend les études bibliographiques difficiles. En outre, l'utilisation d'un sigle devient insuffisante pour représenter des variantes de plus en plus complexes.

Dans ce chapitre, nous proposons une nouvelle notation pour les problèmes de tournées de véhicules. Cette notation a pour principale propriété d'identifier sans ambiguïté ces problèmes en éliminant les défauts cités ci-dessus. Cela sera illustré par des exemples dans la section suivante (2.1.1). Elle propose un formalisme commun permettant de positionner (classifier) les travaux de la littérature. La notation proposé ici ressemble à la classification proposée dans (Desrochers *et al*., 1990) par le niveau de détails mais est basée sur un schéma du type $\pi/\alpha/\beta/\gamma$ tel que celui utilisé avec succès en ordonnancement d'ateliers de production. D'un point de vue général, le champ π caractérise le type de problème, alpha décrit la structure du système, des ressources et des demandes, bêta décrit des contraintes et gamma indique les critères à optimiser (objectifs).

La section (2.3) détaille la notation proposée après que la section (2.2) ait présenté divers schémas de classification proposés jusqu'ici pour les problèmes de tournées. Enfin, la notation proposée est mise en œuvre pour classer quelques articles du domaine correspondant à diverses variantes afin d'illustrer ses possibilités d'utilisation dans la section (2.5).

2.1.1 Limites de l'identification des variantes du VRP par sigles

Cette section démontre à travers quelques exemples de la littérature que l'identification des problèmes par sigles, conjuguée avec l'absence de gestion centralisée des sigles, peut conduire et a déjà conduit d'une part

- à utiliser deux notations différentes pour un même problème
- ou inversement à utiliser la même notation pour représenter deux problèmes différents.

Les divers préfixes et suffixes entourant le sigle VRP peuvent être combinés pour indiquer la présence de différentes hypothèses ou contraintes. Ainsi, par exemple,

- l'appellation SDVRPTW désigne le « *Split Delivery Vehicle Routing Problem with Time Windows* », dans lequel chaque client peut être livré en plusieurs fois, mais sous des contraintes de fenêtres temporelles;
- l'appellation VRPMVTTW ou *Vehicle Routing Problem with Multiple Vehicle Types and Time Windows*, est un autre problème similaire au précèdent sauf que la contrainte de « *Split Delivery* » est remplacée par une contrainte de flotte hétérogène de véhicules.

Face à la complexification croissante des variantes de problème considérées, les sigles fleurissent dans la littérature, de plus en plus longs, sans pour autant que cela contribue à un positionnement clair des travaux dans la littérature. En effet, les nouveaux suffixes ou préfixes sont souvent créés selon les besoins, par chacun des auteurs, sans qu'aucune liste des sigles existants ne soit tenue à jour. Il en résulte des redondances (deux sigles distincts pour désigner des variantes équivalentes) et/ou des incompatibilités. Par exemple, à ce jour,

- le préfixe S est utilisé par certains auteurs (Shen *et al*., 2006) pour désigner le caractère stochastique du problème,
- alors que d'autres, comme (Hayari *et al.*, 2003), l'interprètent comme étant l'aspect sélectif : seul un sous-ensemble de clients sera finalement visité si la capacité des moyens disponibles ne permet pas de répondre à l'ensemble des demandes.

(Shen *et al.*, 2006) ont étudié un cas de problème de tournées de véhicules où les demandes et les temps de visite des clients sont stochastiques (*VRP with stochastics demands and stochastics travel times*). Alors que (Hayari *et al.*, 2003) ont traité un problème de tournées de véhicules où les tournées peuvent ne pas inclure tous les clients mais où chaque client visité doit l'être dans une fenêtre donnée de temps. Dans ce dernier cas, les clients visités sont choisis pour optimiser plusieurs objectifs qui sont : le profit total, la distance totale parcourue, la durée totale et la charge totale des véhicules. Les différences entre les caractéristiques des deux problèmes traités sont très grandes, mais la même notation SVRP est utilisée dans les deux cas. De même, le préfixe SD, est employé par certains auteurs pour désigner le caractère stochastique comme (Cordeau *et al.*, 2005), tandis que d'autres auteurs l'utilisent lorsque la demande des clients peut être divisée sur plusieurs tournées (*Vehicle Routing Problem with Split Delivery*) comme (Archetti *et al.,* 2002). D'autre part, (Parragh *et al.*, 2006a) ont identifié plusieurs notations existantes dans la littérature pour le problème VRP with Mixed Linehauls and Backhauls : PDP (*Pickup and Delivery Problem*) dans (Mosheiov, 1998) et MVRPB (*Mixed Vehicle Routing Problem with Backhauls*) dans (Ropke et Pisinger, 2006).

Ainsi, ces exemples issus de la littérature montrent de manière flagrante les limites de ce système d'identification. En outre, l'utilisation d'un sigle devient insuffisante pour représenter la conjonction d'hypothèses, de contraintes et d'objectifs des variantes de plus en plus complexes abordées dans la pratique. C'est pourquoi la section (2.3) présente une proposition de notation des problèmes de tournées de véhicules. La principale difficulté rencontrée dans cette démarche est de sélectionner les paramètres les plus pertinents pour rendre la classification suffisamment complète et discriminante tout en étant la plus concise possible. La description de la notation proposée est

précédée d'une présentation de travaux précédents sur la classification des problèmes de tournées.

2.2 Quelques classifications existantes pour les problèmes de tournées

Il existe principalement deux types de classifications, en dehors de celle associée à la désignation par sigle. Généralement, face à l'ampleur de la littérature consacrée à ce domaine, soit les auteurs proposent une classification relativement détaillée mais restreinte à une sous-classe de problèmes, soit ils proposent une classification pour l'ensemble des problèmes de tournées mais qui présente un niveau de détail relativement faible. Ainsi, (Parragh *et al.*, 2006a) (Parragh *et al.*, 2006b) sont un bon exemple du type de classification généralement proposées pour des sous-classes de VRP. Ces auteurs considèrent ce qu' ils appellent le *General Pickup and Delivery Problem* ou *GPDP*. Ils proposent une classification (figure (2.1)) limitée au *Vehicle Routing Problem with Backhauls (VRPB)* et au *Vehicle Routing Problem with Pick-up and Delivery (VRPPD)*. Ces deux variantes de problèmes se caractérisent par la présence de deux types de sites, soit receveurs (*linehauls*), soit livreurs (*backhauls*). Elles se distinguent par le rôle joué par le dépôt. Dans le VRPB, toutes les entités livrées proviennent d'un dépôt et toutes les entités prélevées sont retournées au dépôt, alors que dans le VRPPD les entités livrées peuvent également être obtenues chez les sites livreurs. (Parragh *et al.*, 2006a) classent les articles dédiés au VRPB en trois catégories : *VRP with Clustered Backhauls (VRPCB)* où les livraisons ont lieu avant le premier ramassage (Brandao, 2006), *VRP with Mixed Linehauls and Backhauls (VRPMB)* où ramassages et livraisons sont mêlés dans une même tournée (Crispim et Brandao, 2005) et enfin *VRP with Simultaneous Delivery and Pickup (VRPSDP)* où des clients sont receveur et livreur en même temps (Hoff et Løkketangen, 2006). D'autre part, (Parragh *et al.*, 2006b) classent les VRPPD en deux catégories, selon que les demandes des clients sont indépendantes ou non. Si elles sont indépendantes, chaque unité collectée (au dépôt ou dans un site livreur) peut être employée pour livrer n'importe quel client receveur (unpaired pickup and delivery points). Généralement, cela est possible lorsque le problème est mono produit (tous les produits transportés sont du même type), et le problème est alors appelé Pick-up and Delivery VRP (PDVRP) (Chalasani et Motwani, 1999). Lorsque les demandes des clients sont liées (paired pickup and delivery points), chaque transport doit lier une origine et une destination précises. Il s'agit du Pickup and Delivery Problem (PDP) pour le transport de marchandises (Ambrosini *et al.*, 2004) et du Dial-A-Ride Problem (DARP) pour le transport de personnes (Aldaihani et Dessouky, 2003).

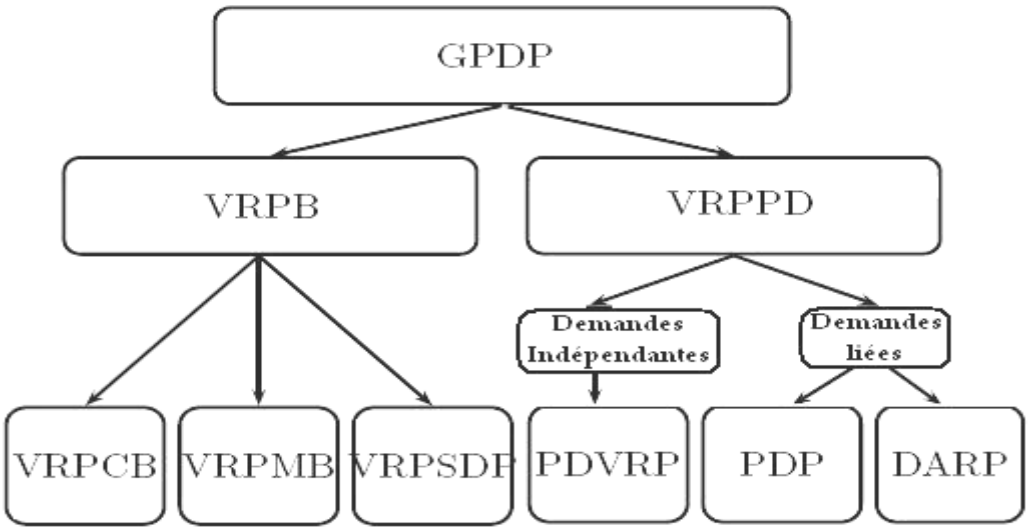


Fig. 2.1 – Classification proposée par (Parragh *et al*., 2006a) et (Parragh *et al*., 2006b)

A l'inverse, (Ghiani *et al.*, 2003) ont classé l'ensemble des VRP mais uniquement en fonction de la nature des données du problème (voir figure(2.2)). Si au moins un élément du problème est aléatoire, le problème est dit stochastique. Dans le cas contraire, il est déterministe. De plus, si toutes les données sont parfaitement connues dès le début de la résolution, il est dit statique. Enfin il est dynamique si l'une au moins des données nécessaires à la planification dépend du temps. Dans ce cas, toutes les données ne sont pas disponibles au début de la résolution, de nouvelles données se présentent aléatoirement en cours de résolution et doivent être intégrées dans celle-ci.

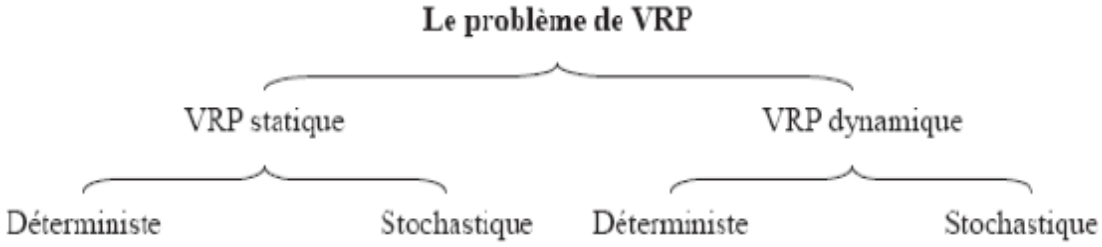


Fig. 2.2 – Classification des problèmes de VRP selon la nature des données

L'une des rares classifications qui s'adresse à l'ensemble des « *Vehicle routing and scheduling problems* » tout en ayant un niveau de détail élevé est celle que proposèrent (Desrochers *et al.*, 1990). Ces auteurs proposent un langage pour décrire des classes de problèmes. Ils utilisent quatre champs décrivant respectivement les adresses (dépôts et sites à visiter), les véhicules, les autres caractéristiques du problème (celles qui ne sont liées ni aux adresses ni aux véhicules) et les objectifs. Une structure hiérarchique est utilisée, composée de champs, sous-champs et paramètres qui constituent les symboles non terminaux. Les symboles terminaux sont les valeurs que prennent les paramètres pour une instance de problème donnée. Des opérateurs logiques (ou exclusif, et) sont

utilisés pour lier les symboles terminaux. Et une valeur par défaut, qui correspond aux valeurs les plus connues ou les plus fréquemment utilisées dans la littérature, est employée pour rendre la notation plus concise. Cette notation est très intéressante, notamment par son niveau de détail élevé. Les auteurs illustrent son utilisation à travers quatorze exemples pour démontrer qu'elle permet de représenter des variantes assez complexes. Elle permet notamment de préciser

- si les demandes se situent sur les nœuds, sur les arcs ou constituent des paires origine-destination ;
- s'il est nécessaire de sélectionner un sous-ensemble de clients, en précisant éventuellement s'il faut desservir au moins un client dans chaque sous-ensemble d'une partition donnée ;
- si les véhicules comportent des compartiments dédiés ou interchangeables,
- si les demandes et/ou les véhicules sont soumis à des contraintes de fenêtres temporelles spécifiant quand le service peut être rendu

L'une de ses limites est d'être globalement prévue pour des données statiques et déterministes. Seule la description des demandes inclut un paramètre pour indiquer un éventuel caractère stochastique. D'autre part, certains symboles terminaux sont numériques ou exprimés par des opérateurs (±, /) plutôt que par des mots ou des abréviations de mots, ce qui rend leur mémorisation ardue. Notons aussi que même si son niveau de détail est élevé, cela ne permet pas de représenter certaines caractéristiques importantes liées, par exemple, à des problèmes de tournées périodiques (voir 1.3). Les auteurs se contentent de préciser si le problème est périodique ou pas dans les contraintes du champ « adresse » alors que des informations sur la planification du service sont primordiales pour appréhender la complexité du cas traité. Son principal atout est son fort pouvoir descriptif. Mais a priori cela n'a pas suffi pour la faire adopter par la communauté car elle n'a que peu été utilisée par la suite. C'est pourquoi la nouvelle notation proposée ici s'appuie sur un niveau de détail relativement similaire à celle de (Desrochers *et al.*, 1990) tout en utilisant des schémas plus standards et des moyens mnémotechniques pour faciliter son adoption. La section suivante décrit notre proposition de manière plus détaillée.

2.3 Notation proposée

Pour augmenter nos chances de succès, la notation proposée s'appuie sur un schéma standard, des règles simples et systématiques de construction, et une sémantique des symboles qui favorise la mémorisation.

2.3.1 Structure standard

La structure de la notation proposée est une adaptation de l'une des classifications les plus utilisée en ordonnancement d'ateliers de production, qui fut proposée par (Rinnooy Kan, 1976) et décrite également dans (Graham *et al.*, 1979) et (Blazewicz *et al.*, 1993). La notation de (Rinnooy Kan, 1976) inclut différents champs α/β/γ qui décrivent

respectivement l'environnement de production, les tâches et les ressources et enfin le critère à optimiser. Elle a déjà été adaptée pour classer divers types de problèmes, notamment les problèmes d'ordonnancement de robots sur les lignes de traitement de surface (*Hoist Scheduling Problems* ou *HSP*) dans (Bloch et Manier, 1999) et (Manier et Bloch, 2003), et les problèmes d'équilibrage de lignes d'assemblage (*Assembly Line Balancing* ou *ALB*) dans (Boysen *et al.*, 2006) et (Boysen *et al.*, 2007).

Dans le cadre des problèmes de tournées de véhicules, l'adaptation proposée consiste à utiliser quatre champs π/α/β/γ pour décrire respectivement le contexte, les ressources (fixes ou mobiles) et les demandes, d'autres contraintes que celles propres aux ressources ou aux demandes et enfin les objectifs à optimiser. Les sections suivantes détaillent la notation ainsi proposée pour les problèmes de tournées de véhicules.

2.3.2 Syntaxe et règles de construction

La notation proposée se construit ou se décrypte, pour un problème donné, à l'aide de règles simples et systématiques décrites ci-dessous. Elle comporte quatre champs π/α/β/γ, dont l'un (α) décomposé en sous champs (α1, α2 et α3). Chaque champ (π, α, β, γ) ou sous champ (α1, α2 et α3) contient plusieurs paramètres. La séparation

- entre champs se fait par des « / »,
- entre sous-champs ou entre paramètres d'un même champ se fait par « , ».

Chaque paramètre correspond à un couple (clé, valeur) où la clé est le nom de ce paramètre. Les séparateurs « / » sont présents même en cas d'absence du champ ou du sous-champ précédent. Par exemple la notation π/α1, α2, α 3//γ exprime l'absence du champ β.

La section suivante (Section 2.3.3. Sémantique des symboles) décrit précisément l'ensemble des couples utilisés. Les clés (noms de paramètres) ont été choisies pour faciliter leur mémorisation:

- en réutilisant lorsque cela était possible des sigles déjà utilisés dans la littérature,
- ou à défaut en utilisant des termes ou abréviations de termes dont le sens est en rapport avec le paramètre.

Pour instancier la valeur d'un paramètre donné, il existe trois niveaux (possibilités) :

- niveau 1 : le paramètre n'apparaît pas dans la notation, ce qui signifie qu'il prend sa valeur par défaut, correspondant au cas le plus fréquemment rencontré dans la littérature. Ceci permet de ne pas surcharger la notation et d'obtenir le résultat le plus concis possible, en particulier pour les problèmes de base de VRP.
- niveau 2 : la clé seule apparaît sans autre précision de valeur, ce qui signifie que l'hypothèse, la contrainte ou l'objectif correspondant apparaît dans le problème de manière statique et déterministe.
- niveau 3 : la clé est suivie d'une ou plusieurs valeurs qui sont réunies dans ce cas par l'opérateur de concaténation « : » pour décrire des cas plus complexes. Les valeurs utilisables sont de trois types. Il peut s'agir

  1. de valeurs propres à la clé considérée : Après l'opérateur de concaténation «: », un ensemble de valeurs générales possibles sont utilisables par chacun pour

apporter de l'information complémentaire au lieu de se contenter de signaler la seule prise en compte du paramètre. Ces valeurs sont à choisir dans un ensemble décrit entre accolades { }, dans lequel les valeurs possibles sont séparées par des « - ». Pour certains paramètres les valeurs ne sont pas exclusives, plusieurs peuvent être utilisées ensemble. Par exemple, dans l'ensemble {Ve - Wo - Dp - C - Net} des valeurs générales du paramètre Tw, il est possible d'utiliser à la fois Ve et C pour spécifier qu'il existe des contraintes de fenêtres temporelles pour les véhicules mais également pour les clients.

2. de la valeur générique OT, signifiant « Other », prévue pour représenter les cas rencontrés dans la littérature et qui ne peuvent pas être représentés à l'aide des valeurs générales prédéfinies précédentes.

De plus, l'ajout des lettres D ou S permet de spécifier le caractère dynamique ou stochastique. Et les abréviations définies dans la description détaillée qui suit peuvent être ajoutées pour préciser sur quel élément particulier portent la contrainte et/ou les caractères stochastique ou dynamique.

Finalement, le signe ↔ représente l'existence d'une relation entre deux types d'éléments. Par exemple, lorsqu'il y a des contraintes de compatibilité entre des véhicules et des clients (ce qui se note Ve↔C).

2.3.3 Sémantique des symboles

Les paragraphes suivants détaillent les clés et valeurs des paramètres constituant les différents champs et sous-champs. Ils sont émaillés d'exemples ponctuels pour illustrer les principes décrits dans la section précédente, en particulier en ce qui concerne la concaténation de différentes valeurs. Les tableaux 1.a et 1.b résument la notation globale. L'annexe (B) présente la signification des sigles existants dans ces tableaux.

2.3.3.1 Champ π

Ce champ décrit le problème. Il comporte le sigle RP pour signifier que l'on se réfère à la classification des Problèmes de tournées (*Routing Problems*). Ce sigle a été choisi pour ne pas réutiliser l'un des sigles existants, comme VRP, et éviter ainsi toute ambiguïté quant à la définition du problème de base. Il ne signifie en aucun cas que cette notation est supposée apte à représenter tous les types de « *routing problems* » existants. De plus, le champ π ne contient qu'un paramètre, l'horizon de planification H, qui par défaut vaut une période, mais qui peut être également Multi période (MP). Cela correspond au Problème de Tournées de Véhicules Multi-Périodique (*Periodic Vehicle Routing Problem* ou *PVRP*) dans lequel chaque client doit être visité k fois au cours de l'horizon ($1 \leq k \leq M$), et les demandes quotidiennes sont fixes (Lacomme *et al.*, 2005). Comme par exemple dans les références (Francis et Smilowitz, 2006), (Mingozzi, 2005) et (Alonso *et al*., 2006) dans le tableau (2.2).

2.3.3.2 Champ α

Le champ α définit la structure du système, en trois sous-champs

- α1 : description des ressources fixes : dépôt(s) (Dp) et réseau(x) (Net)
- α2 : description des ressources mobiles : véhicules (Ve) et travailleurs (Wo)
- α3 : description des demandes (Dd) : clients (C), quantités (Q) et localisations (Lo).

a. Sous-champ α1 (ressources fixes)

Le sous-champ α1, description des ressources fixes (dépôt(s) **(Dp)** et réseau(x) **(Net)**), a trois paramètres

- Le type de dépôt(s) :
  - si tous les dépôt(s) sont livreurs (*backhaul)* le paramètre prend sa valeur par défaut,
  - s'ils sont tous receveurs (*linehaul)* sa valeur est **Dp:Li**,
  - si deux types différents de dépôts coexistent (certains livreurs (b*ackhaul*) , d'autres receveurs (*linehaul*) la valeur est **Dp:(B, Li),**
  - si certains dépôts peuvent être à la fois livreurs (*backhaul*) et receveurs (*linehaul*) le paramètre vaut **Dp:(B+Li)** comme (Ramdane Cherif, 2002)**,**
  - dans le cas d'un simple entrepôt qui ne sert qu'à héberger les véhicules (donc ni livreur, ni receveur), la valeur est **DP:W** (*warehouse*).

Remarque : la distinction entre collecte et livraison peut sembler inutile quand toutes les tâches sont du même type, car le passage d'un cas à l'autre est une simple inversion de flux, toutefois, il a paru préférable de distinguer les deux types pour plus de clarté par rapport aux problèmes combinés (voir par exemple le cas du PDP).

- Le nombre de dépôts pour chaque type : égal à un (par défaut) ou supérieur à un (Multi-dépôt, **MDp**). Cela permet de représenter le *Multi-Depot Vehicle Routing Problem (MDVRP)*, qui comporte plusieurs dépôts dans lesquels sont localisés les véhicules comme (Ho *et al.*, 2008) (Mingozzi, 2005).
- Le type de réseau (**Net**): le réseau peut être non orienté (par défaut), lorsque chaque tronçon de route a deux sens de parcours ; orienté (**ONet**) comme (Jih et Hsu, 2004), si chaque tronçon a un seul sens de parcours ; ou mixte (**MixNet**) contenant à la fois des arcs et des arêtes.

b. Sous-champ α2 (ressources mobiles)

Le sous-champ α2 détermine toutes les informations qui concernent les ressources mobiles [véhicules (**Ve**) et travailleurs (**Wo**)]. Il a trois paramètres

- Le nombre de véhicules
  - est illimité : valeur par défaut,
  - ou limité par une borne constante dans le cas statique et déterministe : **NVe** si la limité est strictement supérieure à 1 et **NVe :1** si la limite égale 1,
  - ou variable au cours du temps dans le cas dynamique : **NVe:D**

- ou défini par une loi stochastique dans le cas stochastique : **NVe: S**.

Dans beaucoup de travaux le nombre de véhicules est limité par une borne constante (NVe) comme (Cordeau et Laporte, 2002) (Alba et Dorronsoro, 2004) (Prins, 2004) (Tavares *et al*., 2003) (Alvarenga *et al*., 2007) (Ombuki *et al*., 2006) (Le Bouthillier, 2000) (Berger *et al*., 2001) (Lau *et al*., 2003) (Dell'Amico *et al*., 2006) (Jih et Hsu, 2004) (Vianna *et al*., 1999) (Francis et Smilowitz, 2006) (Mingozzi, 2005) (Alonso *et al*., 2006), voir tableau (2.2). Un autre exemple pour la limite égale 1 (NVe :1) est (Jih et Hsu, 2004).

- Le type de flotte
  - cette dernière peut être homogène (si elle est composée de véhicules identiques) : valeur par défaut,
  - ou hétérogène, lorsque les véhicules ont des caractéristiques différentes, par exemple en termes de capacité ou de vitesse : **Het**, comme Het (Liu et Shen,1999) (Dell'Amico *et al.*, 2006) (Alonso *et al.*, 2006)
- Le nombre de travailleurs peut-être
  - illimité : valeur par défaut,
  - limité de manière statique et déterministe : **NWo**,
  - limité de manière dynamique : **Nwo:D**
  - ou limité de manière stochastique : **NWo:S**.

Remarque : le terme de travailleur représente toute personne qui contribue au service : conducteurs, réparateurs.

c. Sous-champ *α*3 (demandes)

Ce sous-champ caractérise les demandes (**Dd**) : clients (**C**), quantités (**Q**) et localisations (**Lo**) en six paramètres

- la localisation (**Lo**) de la demande peut se trouver
  - sur les nœuds, comme dans le VRP : valeur par défaut,
  - sur les arcs, comme pour l'ARP (Ramdane Cherif, 2002), *Arc Routing Problem* : **A**,
  - sur les arcs et les nœuds, c'est-à-dire mixte, comme dans le GRP ou *General Routing Problem* (voir section 1.4) : **Mix**.
- le type de clients (**C**) spécifie si
  - tous les clients sont receveurs (*linehaul*) : valeur par défaut,
  - tous les clients sont livreurs (backhaul) : **C:B**,
  - certains clients sont receveurs, et d'autres sont livreurs : **C:(Li, B)** comme (Pankratz, 2005) (Jih et Hsu, 2004),
  - ou des clients peuvent être receveurs et livreurs à la fois : **C:(Li +B)** comme (Ramdane Cherif, 2002).
- le nombre de types de produits (**Pr**): égal à un par défaut, ou supérieur à un (Multi-produits **Mpr**).

- la nature de la demande (**Dd**) est
  - statique : valeur par défaut,
  - stochastique : **Dd:S**,
  - ou dynamique : **Dd:D**.

Remarque : dans ces dernier cas, il est possible de préciser sur quel paramètre porte l'aspect dynamique ou stochastique : le nombre de clients (Dd:D:C ou Dd:S:C), les quantités (Dd:D:Q ou Dd:S:Q) ou les localisations (Dd:D:Lo ou Dd:S:Lo)), ou plusieurs de ces éléments (ou non exclusif).

- le coût de service (**Ser**) peut
  - être nul : valeur par défaut,
  - correspondre au temps pendant lequel le véhicule est arrêté chez le client (service time) : **Ser:T**,
  - ou correspondre à d'autres coûts, par exemple de chargement/déchargement, fixes ou proportionnels à la charge : **Ser:OT**.

Nous donnons des exemples pour (Ser:T), (Cordeau et Laporte, 2002) (Alba et Dorronsoro, 2004) (Prins, 2004) (Ho *et al*., 2008) (Tavares *et al*., 2003) (Alvarenga *et al.*, 2007) (Labadi *et al*., 2008) (Ombuki *et al*., 2006) (Le Bouthillier, 2000) (Berger *et al*., 2001) (Bräysy,2003) (Lau *et al*., 2003) (Liu et Shen,1999) (Dell'Amico *et al*., 2006) (Pankratz, 2005) (Francis et Smilowitz, 2006) (Ramdane Cherif, 2002).

Remarque : Si le coût de service est non nul (Ser:T ou Ser:OT), il est (par défaut) statique et déterministe, ou variable au cours du temps de façon dynamique (**Ser:T :D** ou **Ser:OT :D**), ou encore défini par une loi stochastique (**Ser:T :S** ou **Ser:OT : S**).

- de même le coût de transport (**Tr**) peut
  - être nul : valeur par défaut,
  - être positif, et inclure potentiellement des coûts fixes (par exemple l'acquisition de véhicules) ou proportionnels au nombre de produits, des coûts distinguant les trajets en haut le pied des trajets au cours desquels le service est rendu, des coûts énergétiques. La notation prévoit deux types de coût classiquement utilisés,
    - la distance parcourue : **Tr:Dis**
    - le temps de trajet (travel time) : **Tr:T**,
    - et elle regroupe les autres cas en un seul (Other) : **Tr:OT**.

Plusieurs articles utilisent (Tr:Dis) comme critère d'optimisation : (Cordeau et Laporte, 2002) (Alba et Dorronsoro, 2004) (Najera, 2007) (Tavares *et al*., 2003) (Alvarenga *et al*., 2007) (Labadi *et al*., 2008) (Ombuki *et al*., 2006) (Le Bouthillier, 2000) (Berger *et al*., 2001) (Bräysy,2003) (Lau *et al*., 2003)(Liu et Shen,1999) (Pankratz, 2005)(Vianna *et al*., 1999) (Alonso *et al*., 2006) et pour (Tr:T) comme (Cordeau et Laporte, 2002) (Alba et Dorronsoro, 2004) (Prins, 2004) (Ho *et al*., 2008) (Tavares *et al*., 2003) (Alvarenga *et al*., 2007) (Labadi *et al.*, 2008) (Ombuki *et al*., 2006) (Le Bouthillier, 2000) (Berger *et al*., 2001) (Bräysy,2003) (Lau *et al*., 2003) (Liu et Shen,1999) (Dell'Amico *et al*., 2006) (Pankratz, 2005) (Jih et Hsu, 2004) (Francis et Smilowitz, 2006) (Ramdane Cherif, 2002)

Remarque : de même que pour le coût de service, le coût de transport s'il est non nul est par défaut statique et déterministe (Tr:Dis, Tr:T ou Tr:OT) ou dynamique

(**Tr:Dis:D**, **Tr:T :D** ou **Tr:OT :D**) ou stochastique (**Tr:Dis:S**, **Tr:T : S** ou **Tr: OT : S**).

Le tableau 1.a résume l'ensemble des éléments relatifs aux champs π et α décrits ci-dessus.

2.3.3.3 Champ β (contraintes)

Le troisième champ, β, décrit les contraintes du problème. En général, une contrainte est une restriction ou une condition portant sur une variable du problème. Dans la littérature des problèmes de tournées, de nombreuses contraintes existent en fonction du problème étudié.

Cette section tente de recenser toutes les contraintes rencontrées dans la littérature :

- la capacité (**Cap**) (par véhicule, dans le(s) dépôt(s) ou chez les clients) est soit
  - illimitée : valeur par défaut,
  - soit limitée statique et déterministe, en précisant quel est (sont) le(s) type(s) d'éléments concerné(s), par exemple **Cap:Ve**, **Cap:Dp** ou **Cap:C**, ou **Cap:Ve:C** (ou non exclusif).
- l'autonomie (**Aut**) représente soit l'autonomie des véhicules, (durée maximale entre le départ d'un véhicule et son retour au dépôt), soit le temps de travail maximal des travailleurs, et peut être
  - illimitée : valeur par défaut,
  - ou limitée, statique et déterministe : **Aut:Ve** ou **Aut:Wo**.

  Pour (Aut:Ve), quelques références dans le tableau (2.2) ont cette contrainte (Cordeau et Laporte, 2002) (Alba et Dorronsoro, 2004) (Prins, 2004) (Pankratz, 2005) (Alonso *et al.*, 2006).
- les fenêtres de temps (*time windows*) (**TW**) permettent de spécifier un (ou plusieurs) intervalle(s) de temps à respecter, que ce soit
  - pour servir un client : **TW:C**,
  - entrer dans le dépôt : **TW:Dp**,
  - respecter la réglementation du temps de travail : **TW:Wo**,
  - ou encore représenter la disponibilité des véhicules : **TW:Ve**
  - ou de certaines parties du réseau : **TW : Net**.

  Ceci permet de représenter la variante du VRP à fenêtre de temps la plus répandue, où les fenêtres sont liées au client, mais également certaines variantes du problème incluant des fenêtres de temps relatives aux horaires d'ouverture du dépôt ou aux horaires de travail du personnel (Ombuki *et al.*, 2006).
- le nombre de clients non servis représente le caractère sélectif ou non du problème. C'est-à-dire le fait que tous les clients ne sont pas forcément servis si le nombre et la capacité des véhicules ne le permettent pas. Dans ce cas, il faut choisir les clients à satisfaire en priorité (Hayari *et al*., 2003). Par défaut, ce nombre est nul, sinon il est potentiellement non nul (>=0) : **NoSer,** comme (Lau *et al*., 2003).

- la prise en compte des conditions de trafic (**Tra**) : par défaut, le problème ne prend pas en compte les conditions de trafic. Dans le cas contraire, il faut déterminer la nature de ces conditions
  - statiques : **Tra**,
  - stochastiques : **Tra:S,**
  - ou dynamiques : **Tra:D**.
- la possibilité d'avoir plusieurs tournées (multi-trips) par véhicule (**MT**) précise si les véhicules peuvent exécuter plusieurs tournées, comme (Alonso *et al.*, 2006) (voir section 1.4).
- le partage des demandes (*split delivery*) (**SD**) : par défaut, la demande des clients doit être satisfaite en une seule fois, mais dans certaines variantes du problème chaque demande (même composée de plusieurs produits) peut être divisée entre plusieurs véhicules (SD). En d'autres mots, dans le cas de SD, chaque client peut être visité plus d'une fois si cela est nécessaire, sa demande est divisée sur plusieurs tournées. Par conséquent, la demande de chaque client peut alors être plus grande que la capacité des véhicules (Archetti *et al.*, 2002). Dans d'autres variantes comme le VRP à compartiments (EL Fallahi *et al.*, 2006), la demande concerne plusieurs produits, le partage (*split*) est toléré par demande mais interdit par produit : **SD:NoPr**.
- la planification du service : dans certaines variantes du VRP comme le « *Periodic VRP* » ou le « *Split Delivery VRP* », la demande peut être satisfaite en plusieurs visites de véhicules (plusieurs tournées). Une contrainte de planification de service est alors définie soit
  - par une fréquence de visite (c'est-à-dire un nombre de visites à respecter sur un horizon d'une ou plusieurs périodes) : **F**. Quelques exemples sont trouvés dans (Vianna *et al.*, 1999) (Francis et Smilowitz, 2006) (Mingozzi, 2005) (Alonso *et al.*, 2006),
  - ou par un espacement (*spacing*) minimum et/ou maximum en périodes entre deux services successifs : **Spa :Min~Max**, comme (Francis et Smilowitz, 2006).
  - ou encore par une liste de combinaisons de périodes par client quand le client ne souhaite pas, par exemple, être servi à des périodes données : **Comb**, comme (Mingozzi, 2005) (Alonso *et al.*, 2006).
- l'affectation des dépôts : Par défaut, un véhicule est affecté à un dépôt, et n'en change pas quelle que soit la période considérée. Parfois, le dépôt d'affectation d'un véhicule peut changer d'une période à l'autre : **DpVar**.
- le retour de chaque véhicule à son dépôt d'origine est obligatoire par défaut. Mais dans certains cas, comme dans l'OVRP (*Open Vehicle Routing Problem*), les véhicules ne sont pas obligés de retourner au dépôt, ou lorsqu'ils y sont obligés, ils revisitent les clients dans l'ordre inverse : **NoR**.
- l'origine des véhicules : par défaut, au cours d'une même période, tous les véhicules partent du même dépôt, mais il existe des cas où les véhicules partent de dépôts différents : **MDpP,** comme (Ho *et al.*, 2008).

	Libellé de paramètre	Valeur par défaut	Valeurs générales
π	Horizon de planification (H)	1 seule période	**MP** (Multi Période)
Ressources Fixes α1	Type de dépôt	Livreur (Backhaul)	**Dp :{ Li - (B+Li) - (B, Li) –W }** **tels que Dp:Li** [Receveur (Linehaul)] **Dp:(B, Li)** [Deux types: Livreur(Backhaul) , Receveur (Linehaul)] **Dp: (B+Li)** [Livreur et Receveur en même temps] **Dp:W** [Entrepôt (Warehouse)]
	Nombre de dépôts (Dp)	1 seul dépôt	**MDp** (Multi-dépôts)
	Type de réseau (Network ou Net)	Réseau non orienté	**{ ONet - MixNet }** **tels que ONet** [Réseau orienté] **MixNet** [Réseau Mixte]
Ressources mobiles α2	Nombre de véhicules (**NV**e)	Illimité	**{ NVe, NV**e : 1 **}** (Limité , statique et déterministe) **tels que NVe : 1** (un nombre de véhicule fixé à 1) **NVe** (un nombre de véhicule fixé > 1)
	Type de Flotte	Homogène	**Het** (Hétérogène)
	Nombre de travailleurs (Workers ou NWo)	Illimité	**NWo** (Limité, statique et déterministe)
Demandes α 3	Localisation (Lo)	Noeuds	**{ A - Mix }** **tels que A** [Arcs] **Mix** [Mixtes(Arcs+Noeuds]
	Type de clients (C)	Receveur (Linehaul)	**C: {B - (Li,B) - (Li + B) }** **tels que C:B** [Livreur (Backhaul)] **C:(Li,B)** [deux types: Receveur (Linehaul), Livreur (Backhaul)] **C:(Li+B)** [Receveur (Linehaul) et livreur(Backhaul) en même temps]
	Nombre de types de produits (Pr)	1 seul type de produits	**MPr** (Multi-produits)
	Nature de demande (**Dd**)	Statique	**Dd : { S - D:{C – Q - Lo}}** **tels que Dd :S** (Demande stochastique) **Dd : D: C** (Nombre de clients dynamique) **Dd : D: Q** (Quantité dynamique) **Dd : D: Lo** (Localisation des clients dynamique)
	Coût de service (**Ser**)	Ø	**Ser:T** (Temps statique et déterministe)
	Coût de transport(**Tr**)	Ø	**Tr:{Dis - T}** **tels que Tr:Dis** (Distance statique et déterministe) **Tr:T** (Temps statique et déterministe)

Tab. 2.1.a – Synthèse de la notation des problèmes de tournées, champs π et α

2.3 Notation proposée

	Libellé de paramètre	Valeur par défaut	Valeurs générales
Contraintes β	Capacité (**Cap**)	Illimité	**Cap: {Ve- Dp - C - Ve:C}** (Limité statique et déterministe) **tels que** **Cap: Ve** (capacité de véhicules**)** **Cap:Dp** (capacité de stockage dans les dépôts) **Cap:C** (capacité de stockage chez les clients) **Cap:Ve:C**
	Autonomie (**Aut**)	Illimité	**Aut: :{Ve - Wo}** (Limité statique et déterministe) **tels que** **Aut: :Ve** (Autonomie des véhicules) **Aut: : Wo** (Temps de travail maximal des travailleurs)
	Time windows **(Tw)**	Ø	**Tw :{Ve- Wo- Dp- C- Net}** **tels que** **Tw : Ve** (TW pour la disponibilité des véhicules) **Tw : Wo** (Tw pour les horaires de travail des travailleurs) **Tw : Dp** (TW pour les horaires d'ouverture du dépôt) **Tw : C** (TW pour les horaires d'ouverture des clients) **Tw : Net** (TW pour certaines parties du réseau)
	Nombre de clients non servis **(NoSer)**	Ø	**NoSer** (>=0)
	Conditions de trafic **(Tra)**	Ø	**Tra (statique et déterministe)**
	Split Delivery (**SD**)	Ø	**{SD-SD :NoPr}** **tels que** **SD** (partage (split) possible par demande et par produit) **SD :NoPr** (compartiment : split par demande seulement)
	Multi-trips (**MT**)	Ø	**MT**
	Retour du véhicule à son dépôt original	Obligatoire	**NoR** (N'est pas obligatoire)
	Dépôt de départ des véhicules	D'un même dépôt	**MDpP** (Les véhicules partent de dépôts différents)
	Planification du service	Ø	**{ F - Spa:Min~Max - Comb } tels que** **F** (Fréquence du service) **Spa:Min~Max** (Espacement minimum (∞ si illimité) et maximum (∞ si illimité) entre deux services successifs pour un même client) **Comb** (Choix de service selon un ensemble de combinaisons de périodes)
	Dépôt de départ d'un véhicule pendant l'horizon de planification	Un véhicule part toujours d'un même dépôt	**DpVar** (Le dépôt de départ d'un véhicule est variable)
	Demandes appairées	Unpaired	**Paired** (appairées)
	Compatibilités	Ø	**α i $\leftrightarrow$ α j** i,j $\in$ {1,2,3} tels que $\alpha 1 \in$ {Dp,Net} $\alpha 2 \in$ {Ve,Wo} $\alpha 3 \in$ {C, Pr}
	Source des produits livrés	Dépôts	**{ C- (Dp ,C) }** **tels que** **C** (Clients) **(Dp, C)** (Dépôts et Clients)
	Précédence (**Prec**)	Ø	**Prec**
Objectifs γ	Coût à minimiser	Trouver des solutions faisables	**{ Tr:Dis - NVe - NoSer - Ser:T - Tr:T - WT – Over: β - OT }** **Tels que** **Tr:Dis** **(**Distance totale) **NVe** **(**Nombre de véhicules utilisés) **NoSer** **(**Nombre de clients non servis) **Ser:T** **(**Temps de service) **Tr:T** **(**Temps de transport) **WT** **(**Temps d'attente chez les clients (Waiting Time)**)** **Over: β** [Violation d'une contrainte : (TW, Cap, Aut…)] **OT** (Other)

Tab. 2.1.b – synthèse de la notation des problèmes de tournées, champs β et γ

- l'appariement des demandes : les points à visiter au cours des tournées sont généralement indépendants (***Unpaired***), mais parfois les demandes des clients sont liées (***Paired***). Ainsi, dans les variantes de type PDP ou DARP, chaque transport lie une origine et une destination spécifiées par un même client, comme (Pankratz, 2005) (Jih et Hsu, 2004).
- les contraintes de compatibilité : par défaut il n'existe aucune contrainte de compatibilité, sinon plusieurs types de contraintes peuvent exister. Elles joignent certains paramètres des champs α1, α2 et α3 deux à deux pour exprimer des relations. Cela permet de représenter différents cas, par exemple :
  - une contrainte d'accès des véhicules à certaines rues, certaines parties du réseau ou à des dépôts,
  - le fait que des travailleurs doivent posséder une compétence particulière pour réaliser certaines demandes de clients,
  - le fait que des véhicules ne peuvent pas transporter certains produits,
  - le fait qu'un produit ne peut pas être stocké dans un dépôt donné…

Par exemple dans (Dell'Amico *et al.*, 2006) et (Alonso *et al.*, 2006) il existe une contrainte liant le service des clients à certains véhicules (Ve↔C).

- la source des produits livrés : les produits proviennent
  - d'un ou plusieurs dépôts : valeur par défaut,
  - des clients comme c'est le cas dans le *Pickup and Delivery Problem* ou PDP, et le *Dial-A-Ride Problem* ou DARP : **C**,
  - ou des deux (dépôts et clients): **(Dp,C)**.
- les contraintes de précédence (**Prec**): par défaut, il n'existe pas de contraintes de précédence entre les visites des clients, mais dans certaines variantes du problème les véhicules doit visiter certains clients avant les autres, comme par exemple dans le *VRP with Clustered Backhauls* ou VRPCB, le *Pickup and Delivery Problem* ou PDP comme (Pankratz, 2005) (Jih et Hsu, 2004) et le *Dial-A-Ride Problem* ou DARP.

2.3.3.4 Champ γ (objectifs)

Ce champ décrit les critères à optimiser. Le problème peut consister simplement à déterminer les solutions faisables (par défaut) ou au contraire à minimiser un coût total. Toutefois, la littérature fait état de diverses fonctions objectif, par exemple minimiser

- le nombre de véhicules utilisés : **NVe**,
- la distance totale : **Tr:Dis**,
- le temps de parcours : **Tr:T**,
- le temps de service : **Ser:T**,
- le nombre de clients non servis : **NoSer**, comme (Lau *et al.*, 2003);
- le temps d'attente chez les clients (*Waiting Time*): **WT**,

- la violation d'une contrainte : **Over**. Dans ce cas il est possible de préciser de quelle contrainte appartenant au champ **β** il s'agit, comme par exemple
  - minimiser le retard chez les clients pour le problème de VRP avec fenêtres de temps (TW) : **Over: TW**, comme (Berger *et al.,* 2001) (Lau *et al.,* 2003) (Jih et Hsu, 2004) (Lau *et al*., 2003) ;
  - minimiser les heures supplémentaires des travailleurs : **Over: Aut:Wo**,
  - minimiser les dépassements de capacité : **Over:Cap**, comme (Jih et Hsu, 2004) (Alba et Dorronsoro, 2004).

Notons que certains de ces critères apparaissent déjà dans les champs précédents. Ceci s'impose pour distinguer les paramètres qui doivent être pris en compte (mais pas forcément comme objectif à optimiser) de ceux qui constituent effectivement les objectifs du problème. Par exemple, le coût de transport Tr:Dis peut avoir à être considéré pour vérifier la contrainte d'autonomie dans un problème pour lequel l'objectif à minimiser est le nombre de clients non servis NoSer. Tr:Dis n'apparaîtra alors que dans le sous-champ α3, alors que NoSer apparaîtra dans β et dans γ. Inversement, Tr:Dis peut être l'objectif à minimiser (apparaissant alors dans α3 et dans γ) dans un problème où NoSer n'apparaît que dans β, uniquement pour signifier le caractère sélectif du problème, sans être lui-même un objectif.

2.4. Applications de la notation proposée

La notation proposée dans la section précédente peut être utilisée pour classer les différents articles de la littérature, pour des objectifs et des niveaux de détails différents. Elle peut servir notamment à

1. identifier plus précisément quels travaux de la littérature portent effectivement sur des variantes identiques ou quasiment identiques entre elles ou à un problème de VRP réel faisant l'objet d'une étude,
2. identifier les variantes très étudiées ou a contrario celles n'ayant pas encore fait l'objet de recherche,
3. déterminer, en fonction des valeurs de certains paramètres de la notation (notamment les contraintes présentes)
   a) quels sont les types de méthodes de résolution utilisés,
   b) pour une méthode en particulier (dans le cadre de notre étude les algorithmes évolutionnaires), quels sont les opérateurs les plus courants,
   c) comparer les méthodes et/ou les opérateurs en termes de performances (en utilisant des résultats présents dans la littérature et/ou des campagnes d'expérimentation)

Cette section illustre ces niveaux d'utilisation par des applications de la notation proposée à plusieurs articles de la littérature. Pour être complètement significative et représentative, cette application aurait nécessité une étude bibliographique très volumineuse, car la littérature dédiée à chacune des nombreuses variantes du VRP est très fournie. Cela représente un travail trop conséquent pour être mené à bien complètement dans le cadre de cette thèse. De plus, étant donné le sujet de la thèse, un intérêt particulier a été porté aux algorithmes évolutionnaires (14 références sur 22,

correspondant aux cellules grisées dans la première colonne du tableau 2.3). Par conséquent, l'échantillon de références utilisé (22 articles en tout) ne permet pas de dégager des conclusions générales fiables sur la littérature dédiée au VRP. Il est toutefois possible d'illustrer les éléments et la démarche qui seraient utilisables dans chaque niveau d'application, à titre d'exemple. Les conclusions qui en résultent restent propres à l'échantillon considéré mais donnent un aperçu de ce que peut apporter la notation selon ces différents points de vue. Pour mener une étude plus large et représentative, il suffirait de généraliser les raisonnements présentés. Pour chaque niveau d'utilisation, un tableau de synthèse présente le résultat. Un commentaire est donné sur l'usage de la notation et sur le tableau pris dans son ensemble. Chaque article cité dans ces tableaux fait également l'objet d'un commentaire.

2.4.1. Le tableau des applications et l'état de l'art

Le tableau 2.2 montre les notations résultant de l'utilisation des tableaux 2.1.a et 2.1.b. pour 22 références de la littérature consacrée au VRP. Ces articles ont été choisis pour couvrir un certain nombre de variantes rencontrées (et les sigles associés). La première colonne fournit la référence de l'article, la seconde le sigle utilisé par les auteurs pour désigner la variante de problème résolue, et la troisième la notation obtenue. Le tableau a été trié pour regrouper les variantes désignées par des sigles identiques.

Dans (Cordeau et Laporte, 2002), les auteurs passent en revue dix des plus importantes approches par recherche tabou pour le problème de tournées de véhicules. Ils les ont comparées et les résultats numériques montrent que les solutions obtenues par l'heuristique (Rochat et Taillard, 1995) sont presque optimales.

(Alba et Dorronsoro, 2004) ont proposé l'utilisation d'algorithmes génétiques cellulaires (*Cellular Genetic Algorithms* ou CGA). C'est une sous-classe d'algorithmes génétiques (AG) dans lesquels la population est structurée de telle façon que les individus ne peuvent interagir qu'avec leurs voisins. Ils utilisent différentes versions d'algorithmes incluant ou non des techniques de recherche locale (2-opt et/ou λ-échange) qui peuvent être applicables après chaque génération sur tous les individus. Cela définit quatre algorithmes différents de type CGA, basés sur un codage direct pour représenter les solutions. La différence entre les algorithmes est simplement la méthode de recherche locale utilisée. Le premier n'en contient pas. La deuxième méthode utilise la méthode 2-Opt. Les deux derniers algorithmes étudiés utiliser 2-Opt et λ-échange ensemble. Ces derniers algorithmes diffèrent par la valeur de λ utilisée ($\lambda = 1$ et $\lambda = 2$). Les opérateurs de recombinaison et mutation utilisés sont *Edge Recombination Crossover* (ou ERX) et les mutations d'insertion, d'échange ou d'inversion. La fonction de *fitness* utilisée est calculée en additionnant le coût total de toutes les routes (distance) et des pénalités si la capacité et/ou le temps d'autonomie d'un véhicule sont dépassées (désignés par (Over:Cap) et (Over:Aut:Ve) dans la notation). Ces auteurs ont choisi les *benchmarks* de (Christofides *et al.*, 1979) pour tester les algorithmes proposés. Leur conclusion indique que les algorithmes de type CGA avec opérateur de recherche locale (spécialement avec $\lambda = 1$) sont capables de trouver des solutions optimales des problèmes testés avec de faibles temps de calcul.

Référence	Sigle	Notation
(Cordeau et Laporte, 2002)	VRP	RP/NVe, Ser:T, Tr:Dis, Tr:T/ Cap:Ve, Aut:Ve/ Tr:Dis, Ser:T, Tr:T
(Alba et Dorronsoro, 2004)	VRP	RP/ NVe, Ser:T, Tr:Dis, Tr:T/ Cap:Ve, Aut:Ve/ Tr:Dis, Over:Cap, Over:Aut:Ve.
(Prins, 2004)	DVRP	RP/ Nve, Ser:T, Tr:T/ Cap:Ve, Aut:Ve/ Tr:T
(Ho *et al.*, 2008)	MDVRP	RP/ MDp, Ser:T, Tr:T/ Cap:Ve, MDpP/ Tr:T
(Najera, 2007)	CVRP	RP/ Tr:Dis,/ Cap:Ve/ Tr:Dis
(Tavares *et al.*, 2003)	CVRP	RP/ NVe, Tr:Dis/ Cap:Ve/ Tr:Dis
	VRPTW	RP/ NVe, Ser:T ,Tr:Dis, Tr:T/ Cap:Ve, Tw:C, Tw:Dp/ Tr:Dis
(Alvarenga *et al.*, 2007)	VRPTW	RP/ Nve, Ser:T,Tr:Dis, Tr:T/Cap:Ve, Tw:C,Tw:Dp/ Tr:Dis
(Labadi *et al.*, 2008)	VRPTW	RP/Ser:T, Tr:Dis,Tr:T/Cap:Ve, Tw:C,Tw:Dp/ Tr:Dis
	VRPTW	**RP/ Ser:T, Tr:Dis,Tr:T/Cap:Ve, Tw:C,Tw:Dp/ NVe,Tr:Dis**
(Ombuki *et al.*, 2006)	VRPTW	**RP/**NVe**,Ser:T,Tr:Dis,Tr:T/Cap:Ve,Tw:C,Tw:Dp/NVe,Tr:Dis**
(Le Bouthillier, 2000)	VRPTW	**RP/** NVe**,Ser:T**, **Tr:Dis, Tr:T/ Cap:Ve, Tw:C,Tw:Dp/ Nve, Tr:Dis**, Tr:T, WT, OT
(Berger *et al.*, 2001)	VRPTW	**RP/** NVe**,Ser:T**, **Tr:Dis, Tr:T/ Cap:Ve, Tw:C,Tw:Dp/NVe, Tr:Dis**, Over:TW, OT
(Bräysy,2003)	VRPTW	**RP/Ser:T,Tr:Dis, Tr:T/Cap:Ve, Tw:C,TW:Dp/ NVe, Tr:Dis** , WT
(Lau *et al.*, 2003)	m-VRPTW	**RP**/ Nve**, Ser:T**, **Tr:Dis, Tr:T / Cap:Ve, TW:C, TW:Dp**, NoSer/ **Nve, Tr:Dis,** NoSer, Over:TW, OT
(Liu et Shen,1999)	VRPMVTTW	RP/Het, Ser:T, Tr:Dis, Tr:T/ Cap:Ve, Tw:C,TW:Dp / Tr:Dis, OT
(Dell'Amico *et al.*, 2006)	FSMVRPTW	RP/NVe, Het, Ser:T, Tr:T/ Cap:Ve,TW:C, Ve↔C / Tr:T, Ser:T, NVe,WT
(Pankratz, 2005)	PDPTW	RP/ Dp:W, C:(Li,B), Ser:T, Tr:T, Tr:Dis/ Cap:Ve, Tw:C, Tw:Dp, Aut:Ve, Paired, Prec/ Tr:Dis
(Jih et Hsu, 2004)	1-PDPTW	RP/ Dp:W,C, (Li,B),ONet, NVe:1,Tr:T/Cap:Ve, Tw:C, NoR, Paired, Prec/ Tr:T, WT, Over:TW, Over:Cap.
(Vianna *et al.*, 1999)	PVRP	RP(MP)/NVe, Tr:Dis/ Cap:Ve,F/Tr:Dis
(Francis et Smilowitz, 2006)	PVRP-SC	RP(MP)/ NVe, Ser:T,Tr:T/ Cap:Ve, F, Spa:Min~Max / Ser:T, Tr:T, OT
(Mingozzi, 2005)	MDPVRP	RP(MP)/MDp,NVe, Tr:OT/ Cap:Ve, F, Comb/ OT
(Alonso *et al.*, 2006)	SDMTPVRP ou PVRPMVTAR	RP (MP)/ NVe, Het, Tr:Dis/ Cap:Ve, Aut:Ve, MT, F, Comb, Ve↔C/ Tr:Dis
(Ramdane Cherif, 2002)	CARP	RP /Dp:(B+Li) , A , C :(Li +B), Ser:T,Tr :T / Cap:Ve/ Ser:T, Tr:T

Tab. 2.2 – Synthèse des premier et second niveaux d'application de la notation proposée

Dans (Prins, 2004) l'objectif du travail est de présenter un algorithme génétique hybride efficace pour une version du VRP nommée Distance-constrained VRP (ou DVRP). Cet article propose une procédure de partage en tournées (*splitting*) pour couper les solutions, qui constituent en quelque sorte des « macro-tours » lorsqu'elles sont représentées par le codage indirect. La fonction objectif est le coût total de la solution en termes de temps. Cet auteur a utilisé deux opérateurs de croisement (OX et LOX) et une procédure d'amélioration par recherche locale (*local search* ou LS). Les benchmarks utilisés appartiennent à deux groupes. Le premier contient 14 problèmes classiques proposés par (Christofides *et al.*, 1979) et le deuxième contient 20 *benchmarks* présentés par (Golden et al, 1998). En termes de résultats moyens, l'AG proposé surpasse toutes les métaheuristiques publiées pour les *benchmarks* de (Christofides *et al.*, 1979), sauf une, et devient le meilleur algorithme disponible pour les 20 *benchmarks* générés par (Golden et *et al.*, 1998).

Deux approches d'algorithmes génétiques hybrides (HGA1 et HGA2) ont été développées dans (Ho *et al.*, 2008) pour résoudre le VRP Multi-Dépôt (MDVRP). L'objectif est de réduire au minimum le temps total de livraison. La différence majeure entre les deux approches proposées repose dans la génération des solutions initiales. Elles sont générées aléatoirement dans HGA1. Tandis que HGA2 utilise la méthode de (Clarke et Wright, 1964) (*saving method*) et l'heuristique du plus proche voisin (Nearest Neighbor Heuristic ou NNH) (Reinelt, 1994) pour générer la population initiale. Les chromosomes sont une forme directe de représentation des solutions. Ensuite, l'AG comprend le croisement *order crossover* (OX), une mutation heuristique, et une mutation d'inversion. La version originale de la mutation heuristique génère, à partir d'un parent, un ensemble de chromosomes transformés par l'échange de certains gènes pour former un voisinage. Seule la meilleure solution dans ce voisinage est utilisée comme résultat de cette mutation. Pour le MDVRP, (Ho *et al.*, 2008) adaptent la mutation heuristique originale. Ainsi dans cet article tous les voisins générés sont conservés. Ensuite, pour accroître l'efficacité des approches proposées, une heuristique nommée *Iterated Swap Procedure* (ou ISP) (Ho et Ji, 2003) (Ho et Ji, 2004) est adoptée pour améliorer les chromosomes générés par tous les opérateurs génétiques utilisés. La performance des approches proposées est évaluée au moyen de deux problèmes à deux dépôts, générés de façon aléatoire (50 clients et 100 clients). Les résultats de comparaison entre HGA1 et HGA2 montrent que la performance de HGA2 est supérieure à celle de HGA1 en termes de qualité des solutions. Les solutions initiales générées par HGA2 sont meilleures que celles générées par HGA1. Et HGA2 a généré des solutions finales présentant de meilleurs temps de livraison que celles de HGA1.

(Najera, 2007) a introduit un algorithme génétique pour résoudre le CVRP et trouver la distance totale minimale. L'auteur a repris le codage direct, l'opérateur de croisement générique (*generic crossover*) et les quatre opérateurs de mutation de (Tavares *et al*., 2003) : insertion, échange (*swap*), inversion et déplacement. Bien que (Najera, 2007) ait utilisé l'idée de codage de (Tavares *et al*., 2003) il l'a légèrement modifié. L'individu se compose de trois parties : le nombre de véhicules, la permutation des clients et l'itinéraire pour chaque véhicule. Le nombre de véhicules est donné dans le premier gène du chromosome. Les gènes suivants contiennent la permutation des clients. La dernière partie du chromosome montre l'itinéraire spécifique pour chaque véhicule en indiquant la gamme de gènes pour chaque itinéraire. Une autre différence entre les deux

articles est la sélection pour reproduction. (Tavares *et al*., 2003) a utilisé la sélection par tournoi tandis que (Najera, 2007) a utilisé la sélection par roulette (*roulette wheel*). Enfin, (Najera, 2007) a utilisé le croisement générique (*generic crossover*) pour résoudre le CVRP tandis que (Tavares *et al*., 2003) l'a utilisé pour traiter le VRPTW et a utilisé une version spécifique de ce croisement pour le CVRP. (Najera, 2007) a sélectionné dix-huit benchmarks connus de (Augerat *et al.*, 1995) (sets A et B) et de (Christofides et Eilon, 1969). Ses résultats sont comparés avec les résultats publiés dans (Tavares *et al.,* 2003). Les meilleurs résultats de son approche d'algorithme génétique ne sont pas aussi bons que ceux déjà publiés. L'auteur suppose qu'il est nécessaire d'explorer plus l'espace de recherche en utilisant un autre opérateur de croisement peut être plus intelligent.

(Tavares *et al.*, 2003) ont réalisé que la représentation des solutions utilisée joue un rôle très important dans les problèmes de tournées. Ainsi, ils ont proposé un nouveau schéma de représentation, appelé (*Genetic Vehicle Representation* ou GVR), pour résoudre deux variantes de problème (VRPTW et CVRP). Une solution candidate précise le nombre de véhicules nécessaires, la partition des demandes entre les véhicules, ainsi que l'ordre des clients dans chaque tournée (codage direct). GVR permet d'ajuster facilement le nombre de véhicules nécessaires. Pour VRPTW, ces auteurs utilisent un croisement dit générique (*generic crossover*) tandis que pour CVRP, ils utilisent une version spécifique de ce croisement développée par leurs soins. Quatre opérateurs de mutation ont été pris en compte : insertion, *swap*, inversion et déplacement (*displacement*). Dans ce dernier cas, une sous-chaine est sélectionnée dans l'une des routes, puis elle est insérée quelque part dans une autre route. Si la capacité du véhicule est dépassée, la route est divisée pour résoudre ce problème. La sélection pour reproduction est basée sur un tournoi. Pour CVRP, les problèmes de test utilisés sont 12 *benchmarks* choisis parmi les benchmarks de (Augerat *et al*, 1995) (sets A et B) et de (Christofides et Eilon, 1969). Pour VRPTW, 12 *benchmarks* de (Solomon, 1987) ont été choisis (quatre cas de chaque type de problèmes C1, R1 et RC1). Une comparaison a été faite entre GVR et un codage indirect nommé *Path Representation* (PR) qui soit utilise des opérateurs génétiques communs (croisement PMX+ Mutation échange) soit utilise les opérateurs génétiques de GVR (dans ce cas il est nommé *modified RP*). Dans ce dernier codage, le chromosome peut être considéré comme une route, et le groupement des demandes est déterminé à la phase d'interprétation du chromosome, en prenant en compte la contrainte de capacité pour obtenir la définition complète des solutions. Les résultats ont révélé que GVR a été capable de trouver de nouvelles meilleures solutions pour CVRP (pour les instances A60k9, A69k9, B57k7, B78k10 et E76k14), tandis que PR (*Path Representation*) n'a pas pu atteindre de meilleure solution. Pour VRPTW, GVR a également été performant. Il a pu découvrir quelques solutions optimales (pour C101 et C102) et fourni de bonnes solutions pour les autres problèmes, alors que le modèle PR a eu une performance faible surtout avec les opérateurs génétiques communs.

(Alvarenga *et al.*, 2007) étudient un problème de tournées de véhicules avec fenêtres de temps et obtiennent de bons résultats (distances) en considérant la distance totale comme objectif principal. Ils utilisent un algorithme génétique avec une formulation de partition d'ensemble (nommée *set partitioning two-phase*). Le codage utilisé est un codage direct : les auteurs ont considéré que chaque route est un chromosome, et qu'une solution complète est représentée par un ensemble de chromosomes. L'article propose

par conséquent un croisement spécifique adapté à cette représentation. Il utilise huit opérateurs de mutation basés sur des heuristiques et nommés : *random customer migration*, *bringing the best customer*, *re-insertion using stochastic PFIH*, *similar customer exchange*, *customer exchange with positive gain*, *merge two routes*, *reinserting customer* et *route partitioning*. Les 56 *benchmarks* de (Solomon, 1987) sont utilisés pour les tests. Pour quelques instances, cette méthode a donné des améliorations des résultats en termes de minimum de distance mais généralement avec une croissance du nombre de véhicules.

(Labadi *et al.*, 2008), par exemple, utilisent une méthode mémétique (algorithme génétique + recherche locale) pour traiter un problème de tournées de véhicules avec des fenêtres de temps. Leurs travaux s'appuient sur un codage indirect pour représenter les solutions sous forme de chromosomes. Ces derniers fixent l'ordre de visite des clients mais sans délimiter les tournées, c'est une séquence des clients visités. Les auteurs adaptent la procédure de (Prins, 2004) (nommée Split) à la variante avec fenêtres temporelles pour découper ce « macro-tour » en plusieurs tournées tout en respectant la capacité des véhicules, et les contraintes de fenêtres temporelles. Le croisement utilisé est OX (Order Crossover). Les auteurs n'utilisent pas à proprement parler d'opérateur de mutation. A la suite du croisement, une procédure de recherche locale basée sur quatre types de mouvements est appliquée à chaque individu créé avec une certaine probabilité, pour chercher à l'améliorer. Les mouvements possibles sont

- Or-opt, qui replace un ou deux clients consécutifs dans une tournée ou dans deux tournées distinctes,
- 2-opt, qui inverse une chaine de clients dans une tournée,
- l'échange de deux clients au sein d'une même tournée ou entre deux tournées,
- 2-opt*, qui échange la fin de deux tournées d'une certaine manière afin d'éviter d'inverser le sens de parcours pour minimiser les risques de violation de fenêtres temporelles.

De plus la diversité est assurée par le respect d'une règle nommée « cost spacing rule », selon laquelle un individu ne peut être conservé que s'il présente une différence de coût suffisante avec les solutions déjà présentes. Le jeu de problèmes de référence (benchm*arks)* fourni par (Solomon, 1987) est utilisé pour comparer les résultats avec d'autres résultats déjà existants. (Labadi *et al.*, 2008) ont étudié deux types de fonction objectif :

- le premier considère la distance totale parcourue comme seul objectif,
- le deuxième considère deux objectifs qui sont de minimiser le nombre de véhicules et ensuite de minimiser la distance totale [objectifs de (Solomon, 1987)].

Pour le premier cas (désigné par le sigle MA1), l'algorithme est très rapide et montre de très bonnes performances pour minimiser la distance totale. Par contre, pour le cas de deux objectifs (avec la priorité de minimiser le nombre de véhicules), l'algorithme donne des résultats un peu moins bons. Les auteurs ont conseillé d'utiliser un autre opérateur de croisement pour améliorer les résultats.

Les auteurs ne précisent rien quant au temps de service des clients (Ser:T) mais il apparaît dans la notation de cette référence dans le tableau 2.2, car ils ont utilisé les 56 *benchmarks* de (Solomon, 1987) pour évaluer leur approche et que ces *benchmarks* tiennent compte des temps de service.

(Ombuki *et al.*, 2006) représentent un problème de tournées à fenêtres temporelles comme un problème multi-objectif dont les deux dimensions objectif sont le nombre de véhicules et le coût total en termes de distance. Ils ont utilisé un algorithme génétique basé sur la dominance au sens de Pareto, qui fournit un ensemble de solutions correspondant aux meilleurs compromis identifiés entre ces deux objectifs. Le codage des solutions est indirect, le croisement est développé plus spécifiquement pour les problèmes de tournées, nommé Best Cost Route Crossover. Les 56 *benchmarks* de (Solomon, 1987) sont à nouveau ceux choisis pour valider la méthode. Celle-ci s'est avérée efficace pour trouver de nouvelles solutions (surtout pour les problèmes de type R2 et RC2) améliorant l'objectif de distance mais pas le nombre de véhicules.

(Le Bouthillier, 2000) a proposé une combinaison de diverses métaheuristiques pour résoudre le problème de tournées de véhicules avec fenêtres de temps dures. Il a choisi de combiner de la recherche tabou, des algorithmes génétiques combinant OX (*Order Crossover*) et ER (*Edge Recombination*) comme opérateurs de croisement, ainsi que des optimisations locales de type λ-opt pour une adaptation avec fenêtres de temps. Un opérateur de mutation par insertion est appliqué avec une faible probabilité. La fonction de *fitness* proposée pour représenter la qualité des solutions est une agrégation de différents objectifs. C'est une somme du temps total de parcours, de la distance totale, de l'attente, du nombre de véhicules et du temps restant chez les clients avant la fin de la fenêtre de temps (représenté par OT dans la notation). Les problèmes de test sont les 56 problèmes créés par (Solomon, 1987). L'article présente aussi des résultats sur une extension des problèmes de (Solomon, 1987), soit 60 problèmes à 1000 clients et un dépôt qui respectent la même structure que les précédents. Les performances de ces travaux sont très acceptables : cette approche retrouve généralement des solutions comparables aux meilleures valeurs connues et améliore même la meilleure solution trouvée au moment de la comparaison pour le problème R207.

(Berger *et al.*, 2001) ont traité un problème à contraintes de fenêtres temporelles dites « molles » (ou *Soft time windows*), où les contraintes de fenêtres de temps peuvent être violées. Dans ce cas, le retard (*delay*), est toléré mais au prix d'une pénalité dans la définition de la qualité d'une solution. Ces auteurs utilisent un algorithme génétique hybride parallèle qui emploie deux populations. La première, pop1, est utilisée pour minimiser la distance totale parcourue, tandis que la seconde, pop 2, se concentre sur la réduction des violations de contraintes temporelles. Deux opérateurs de recombinaison sont considérés : IB-X (*insertion-based crossover*) et IRN-X (*insert in route neighborhood crossover*). Six opérateurs de mutation différents sont envisagés : AC-M (*ant colony mutation*), LNSB-M (*Large Neighborhood Search-based mutation*), EE-M (*edge exchange mutation*), RS-M (*repair solution mutation*), RSS-M (*reinsert shortest Solomon mutation*) et RC-M (*reorder customers mutation*). Ces auteurs ont trouvé trois nouveaux résultats (pour les instances R108, RC105 et RC106) qui continuent d'être les meilleures solutions connues et publiées jusqu'à présent en prenant en compte plusieurs objectifs, qui diminuent le nombre de véhicules, la distance totale parcourue, le retard total et le nombre de clients servis en retard (désigné par OT dans la notation).

Pour établir la notation de cette référence, comme dans (Labadi *et al.*, 2008), il n'existe pas de description claire du temps de service des clients dans l'article. Mais comme les *benchmarks* de (Solomon, 1987), utilisés pour l'expérimentation, contiennent des temps de service, nous avons décidé de faire apparaître (Ser:T) dans la notation de (Berger *et al.*, 2001).

(Bräysy, 2003) a considéré une variante du problème de tournées de véhicules avec des fenêtres de temps dures (Time Windows). Pour résoudre ce problème, cet auteur a proposé une approche de recherche à voisinage variable (*Variable Neighbourhood Search* ou VNS). Plusieurs problèmes issus de la littérature sont choisis pour tester la performance de l'approche proposée. Les résultats expérimentaux ont donné des solutions intéressantes surtout en termes de nombre total de véhicules pour l'ensemble des 56 problèmes de (Solomon, 1987). (Ser:T) apparaît dans la notation de (Bräysy, 2003) malgré son absence dans le texte car les instances de Solomon qui servent de benchmarks incluent un temps de service.

(Lau *et al.*, 2003) ont considéré une variante du problème de VRP avec des fenêtres de temps et un nombre limité de véhicules. Une solution faisable est une solution qui peut contenir des clients non servis et/ou les contraintes de fenêtres de temps sont relaxées en utilisant la notion de pénalité de retard. Pour résoudre le problème, ces auteurs ont proposé une approche de recherche tabou. Les résultats expérimentaux fournis sont comparables aux meilleurs résultats existants pour le VRPTW. De plus, quand le nombre de véhicules diminue, les tournées sont plus complètes. L'approche proposée est bonne tant du point de vue de l'optimalité que de celui de la stabilité. Le même algorithme peut être employé également pour donner des résultats raisonnablement bons pour le problème standard de VRPTW. Le paramètre (OT) dans la notation de cet article indique le nombre de client servis en retard.

(Liu et Shen, 1999) ont décrit une méthode de construction des itinéraires pour le problème de tournées de véhicules avec flotte hétérogène et fenêtres de temps (*Vehicle Routing Problem with Multiple Vehicle Types and Time Windows* ou VRPMVTTW). L'objectif est de minimiser la somme des frais d'acquisition de véhicules (OT dans la notation) et de la distance totale parcourue. Plusieurs heuristiques d'insertion (*insertion-based savings*) sont présentées. Leurs performances ont été testées sur vingt-quatre problèmes issus de la littérature puis modifiés, qui comportent 100 sites clients. Les résultats expérimentaux ont démontré que les heuristiques, qui considèrent un processus de construction d'une route séquentiel, donnent une solution de meilleure qualité que de celles fournies par les autres heuristiques examinées.

(Dell'Amico *et al.*, 2006) ont traité un problème de VRP avec véhicules hétérogènes et fenêtres de temps pour les clients (*Fleet Size and Mix Vehicle Routing Problem with Time Windows* ou FSMVRPTW). Ils ont développé une heuristique d'insertion constructive (*M-parallel Regard-Tw*) et un algorithme métaheuristique pour trouver la solution de ce problème. Les instances utilisées pour tester la performance de leur approche sont issues de la littérature. Le résultat montre que la méthode proposée est robuste et efficace et qu'elle a de meilleures performances que des résultats précédemment publiés.

(Pankratz, 2005) est le premier à appliquer un algorithme génétique pour résoudre un problème de PDPTW avec plusieurs véhicules et des contraintes dures de fenêtre de temps (*multi-vehicle PDPTW with hard time window constraints*). L'objectif est de

minimiser la distance totale parcourue. Comme cet algorithme génétique utilise une adaptation du codage *group oriented genetic encoding* qui fut introduit par Falkenauer(1998), il est nommé *Grouping Genetic Algorithm* (ou GGA). Dans ce type de représentation, chaque gène dans un chromosome représente une séquence de toutes les demandes qui sont affectées à un même véhicule (codage direct). La longueur d'un chromosome (en nombre de gènes) est variable et dépend du nombre de véhicules dans la solution. Les opérateurs de croisement et de mutation de GGA sont des adaptations du croisement *group-oriented crossover* et de la mutation *group-oriented mutation* présentés dans Falkenauer (1998). Afin de tester l'algorithme proposé, deux ensembles de *benchmarks* pour le PDPTW sont utilisés : 9 *benchmarks* de 100 nœuds donnés par (Nanry et Barnes, 2000) et 56 *benchmarks* de 100 nœuds donnés par (Li et Lim, 2001). Les résultats montrent que le GGA est une technique appropriée pour résoudre le PDPTW. Il est capable de trouver des solutions de haute qualité. L'auteur a utilisé des *benchmarks* qui contiennent des temps de service. Par conséquent, (Ser:T) apparaît dans la notation de cette référence alors même qu'il n'était pas présent clairement dans l'article. Ici l'adaptation du codage est liée à la classe de problème étudiée, puis les opérateurs de croisement sont adaptés à ce codage particulier.

(Jih et Hsu, 2004) ont présenté une approche nommée *Family Competition Genetic Algorithm* (ou FCGA) pour résoudre une variante appelée Single-Vehicle Pickup and Delivery Problem with Time Window (1-PDPTW). Dans FCGA, et pour chaque génération, la population (P) contient m individus où chaque individu Ii dans P est considéré comme père Fi pour créer une famille Ci de u enfants. En détails, le processus de construction d'une famille fonctionne comme suit : pour produire un enfant cj, un second parent Aj est choisi au hasard à partir de la population P. Ensuite, un opérateur de croisement est appliqué entre Fi et Aj, et l'individu résultant subit une mutation pour produire un enfant cj. Ce processus est répété jusqu'à ce que le nombre d'enfants (u) soit atteint, ce qui construit la famille Ci pour le père Fi. En conclusion, l'algorithme obtient m groupes d'enfants (famille) Ci pour chaque solution Fi. Puis, le meilleur enfant de chaque famille est recueilli dans un ensemble temporaire (T). Enfin, les m meilleurs individus parmi la population P et l'ensemble T sont sélectionnés pour former la prochaine génération. Ces auteurs ont comparé l'algorithme proposé avec l'algorithme génétique classique (GA) en utilisant quatre opérateurs de croisements qui sont (OX), (UOX), Merge crossover #1 (MX1) et Merge crossover #2 (MX2). Les expériences utilisent également deux opérateurs de mutation : échange et inversion, mais sous les noms de 2-point mutation et 2-opt mutation. La fonction objectif contient le temps total de transport qui inclut le temps d'attente et une pénalité si un véhicule est surchargé ou arrive en retard à l'un des sites à visiter. Les auteurs ont généré un ensemble de tests aléatoires pour leurs expériences. Ils concluent que tous les opérateurs ne sont pas également utiles à FCGA. En particulier, FCGA est mieux adapté pour les opérateurs génétiques qui explorent l'espace de recherche de manière uniforme, comme UOX. Ainsi, par exemple l'application de FCGA avec Merge crossover #1 (MX1) et Merge crossover #2 (MX2) n'améliore pas sensiblement les solutions. En général, les temps d'exécution de l'AG classique sont plus courts que ceux de FCGA, mais par contre FCGA améliore la qualité des solutions en utilisant UOX. Les résultats montrent que FCGA a réussi à trouver des solutions optimales pour des problèmes comportant moins de 50 tâches, et des bonnes solutions (presque optimales) pour la plupart des problèmes contenant jusqu'à 100 tâches.

(Vianna *et al*, 1999), une métaheuristique hybride évolutive parallèle (*PARallel Hybrid Evolutionary Metaheuristic* ou PAR-HEM) est utilisée pour résoudre un problème périodique de tournées de véhicules (*Period Vehicle Routing Problem* ou PVRP). Dans cette variante, chaque client a une demande quotidienne connue, et l'horizon de planification est fixe et égal à M jours. L'algorithme proposé est basé sur les concepts utilisés au sein des algorithmes génétiques parallèles et des heuristiques de recherche locale. Il emploie un modèle d'algorithmes génétiques par ilôts (*Island model*) où la population de chromosomes est divisée en plusieurs sous-populations qui évoluent et migrent parallèlement leurs individus (chromosomes) entre elles afin d'augmenter la diversification. Le taux de migration est faible. Celle-ci est exécutée uniquement lorsque le renouvellement d'une sous-population est nécessaire. La représentation des solutions peut être considérée comme un codage direct où chaque chromosome contient deux vecteurs. Le premier vecteur présente les jours de visite chez chaque client. Chaque gène est associé à un client, et contient une valeur entière. Celle-ci est traduite en une chaine binaire dont la longueur est égale au nombre de jours de l'horizon de planification. Les 1 apparaissant dans cette chaine binaire correspondent aux jours de visite de ce client. Le second vecteur, associé au premier, indique les quantités cumulées desservies chaque jour pendant l'horizon de planification. Les auteurs ont donc utilisé un croisement spécifique adéquat pour cette représentation adaptée à la périodicité du problème (exprimée par MP dans le champ π de la notation). L'opérateur de mutation exécute des échanges faisables en des points choisis aléatoirement dans un chromosome. En termes de résultat, l'algorithme a été appliqué sur différents problèmes posés par (Christofides et Beasley, 1984) et (Golden *et al.*, 1995). PAR-Hem s'est montré très performant pour résoudre le PVRP, non seulement en ce qui concerne le temps d'exécution mais aussi en ce qui concerne la qualité des solutions.

Dans (Francis et Smilowitz, 2006), une formulation pour un problème périodique de tournées de véhicules avec choix de services (*Period Vehicle Routing Problem with Service Choice* ou PVRP-SC) a été présentée. Ce problème est caractérisé principalement par le fait que le service est déterminé par la fréquence de visite des clients. Ils ont proposé un modèle d'approximation continue pour le traiter. La méthode d'approximation continue a été implémentée à l'aide de AMPL (*Modeling Language for Mathematical Programming*) (Fourer *et al*., 2003) avec le solveur KNITRO (Waltz, 2004). Pour plus d'information, voir http://www.ziena.com/documentation.htm l). La notation de cette référence utilise OT pour représenter le choix de service.

Plusieurs variantes du VRP (MDPVRP, PVRP, MDVRP, CVRP) ont été abordées dans (Mingozzi, 2005). L'objectif est de minimiser le coût total mais l'article ne détaille pas clairement ce que ce coût inclut, ce qui explique que la notation utilise OT pour désigner l'objectif de cet article. Cet auteur a décrit une formulation par programmation en nombres entiers du MDPVRP qui est une extension de la formulation du problème de partition d'ensemble (*Set Partitioning* formulation). Il a proposé une méthode exacte de résolution qui utilise une variable de tarification (*variable pricing*) afin de réduire l'ensemble des solutions pour permettre de résoudre le problème en un temps raisonnable.

(Alonso *et al.*, 2006) ont considéré un problème périodique de tournées de véhicules qui inclut, en plus des contraintes classiques, la possibilité pour un véhicule de faire plusieurs allers/retours par jour au dépôt (tant que le temps maximum de travail du

véhicule n'est pas dépassé) ainsi que des contraintes d'accès des véhicules à certains clients. En effet, ces auteurs ont considéré des contraintes de compatibilité entre les véhicules (une ressource mobile α2=Ve) et les clients (un paramètre décrivant les demandes α3=C). Les auteurs ont appelé ce problème *Site-Dependent Multi-Trip Periodic Vehicle Routing Problem* ou *Periodic Vehicle Routing Problem with Multiple Vehicle Trips and Accessibility Restrictions* (SDMTPVRP ou PVRPMVTAR). Ils ont utilisé un algorithme de recherche tabou pour résoudre ce problème.

Enfin, (Ramdane Cherif, 2002) a traité le CARP (*Capacitated Arc Routing Problem*) ,qui est la version de base des problèmes de tournées sur arcs dits avec capacités ou multi-véhicules, en utilisant les deux types de codage (direct et indirect). Dans cet article, des quantités sont associées aux liens du graphe. Elles sont soit à livrer comme dans le cas du sablage des rues ou soit à collecter comme dans l'application de collecte des déchets. La capacité limitée des véhicules oblige à construire plusieurs tournées. Le CARP consiste à déterminer un ensemble de tournées de coût total minimal, tel que :

- chaque tournée est assurée par un véhicule qui part et revient au dépôt,
- la somme des demandes traitées par une tournée ne dépasse pas la capacité de véhicule
- chaque arête nécessitant un service est effectivement traitée, par une seule tournée et lors d'un seul passage.

Cet auteur a testé trois croisements pour le CARP: le croisement à un point de coupure (X1 crossover), LOX (*Linear Order Crossover*) et OX (*Order Crossover*). En plus, il a utilisé trois types de mutation. Le premier est l'échange (ou Swap). Il tire au sort deux positions distinctes et permute les tâches situées à ces positions. Enfin, il inverse avec une probabilité de 0.5 le sens de traitement de chacune des tâches, avec une probabilité de 0.5. Le deuxième est (Move) qui tire au sort deux positions distinctes *p* et *q.* Ensuite la tâche *u,* positionnée au rang *p,* est déplacée après celle de rang *q* (*q* = 0 signifiant une insertion en tête). Enfin, le sens de traitement de *u* est inversé (remplacement par *inv(u)*) avec une probabilité de 0.5. Le dernier type est une recherche locale (*local search* ou LS) qui remplace l'opérateur de mutation.

2.4.2. Analyse des applications considérées

Cette section présente une analyse des applications considérées en comportant tout d'abord l'identification et la comparaison des variantes des problèmes étudiées, ensuite nous estimerons l'intérêt porté aux diverses variantes de problème et enfin nous présenterons une comparaison en détails des méthodes utilisées (types, détails, performances) en fonction des variantes.

2.4.2.1 Identification et comparaison des variantes de problème

Dans le tableau 2.2 nous avons montré par des exemples que la notation permet de distinguer les différences entre deux variantes désignées par le même sigle par des auteurs différents. Par exemple, (Cordeau et Laporte, 2002) et (Alba et Dorronsoro, 2004) étudient tous les deux le VRP. Mais mettre en vis-à-vis les notations associées à ces articles montre clairement que ces deux variantes de problèmes, bien qu'étant identiques par ailleurs, n'optimisent pas les mêmes objectifs. Elles ont comme objectif

commun de minimiser la distance (Tr:Dis dans le champ γ). Mais (Cordeau et Laporte, 2002) considèrent que la contrainte de capacité est une contrainte dure et minimisent (en plus de la distance) le temps de transport et de service (Tr:T et Ser:T dans le champs γ. . Au contraire, (Alba et Dorronsoro, 2004) tolèrent des dépassements de capacité et d'autonomie des véhicules, en incluant un objectif pour minimiser ces dépassements (Over:Cap et Over:Aut:Ve) dans le champ γ..

Une comparaison des résultats de ces deux articles est donc délicate. Il faut s'assurer que les solutions finales fournies respectent bien strictement les contraintes de capacité et d'autonomie avant d'effectuer la comparaison en termes de valeurs des objectifs et de temps d'exécution des algorithmes. La notation permet de mettre en évidence cette difficulté alors que l'utilisation du seul sigle VRP ne la fait pas apparaître.

A l'inverse, la notation permet de vérifier que (Tavares *et al.,* 2003) et (Alvarenga *et al.*, 2007) ont bien traité la même variante de problème (voir notations identiques dans les deux cellules grisées de la troisième colonne) et que leurs résultats sont donc objectivement comparables. De la même manière, dans le cadre de la résolution d'un problème de tournées rencontré dans la réalité, il serait possible de lui associer une notation, puis de rechercher ensuite dans la littérature s'il existe une ou plusieurs références ayant une notation identique ou, à défaut, la plus similaire possible. Cela permet de retrouver rapidement les approches de résolution disponibles pour traiter un cas réel s'il en existe, avant de lancer le développement d'une approche spécifique.

Enfin, il n'existe pas, dans le tableau 2.2, d'articles pour lesquels les notations sont identiques alors que les sigles utilisés par les auteurs sont différents. On peut citer, à titre d'exemple, le fait que le même paramètre indiquant une flotte hétérogène de véhicules (Het dans le champ α2) apparaît de la même manière pour (Liu et Shen, 1999), (Dell'Amico *et al.*, 2006) et (Alonso *et al.*, 2006) alors que

- il est représenté par les lettres MVT, qui signifie *Multiple Vehicle Types*, dans le sigle VRPMVTTW (pour *Vehicle Routing Problem with Multiple Vehicle Types and Time Windows*) utilisé par (Liu et Shen, 1999);
- (Dell'Amico *et al.*, 2006) l'indiquent par MV, qui signifie *Mix Vehicle,* dans le sigle FSMVRPTW (pour *Fleet Size and Mix Vehicle Routing Problem with Time Windows*) qu'ils emploient;
- il n'apparaît pas dans le sigle PVRPMVTAR (pour *Periodic Vehicle Routing Problem with Multiple Vehicle Trips and Accessibility Restrictions*) utilisé par (Alonso *et al.*, 2006), dans lequel la sous-chaine MVT signifie *Multiple Vehicle Trips.*

Ainsi la notation permet de lever l'ambiguïté dans ce cas là, de la même façon que dans les deux précédents.

Plus globalement, cette section a montré qu'elle permet de remplir le premier objectif fixé qui portait sur l'identification de la variante traitée par un article de la littérature,

- pour déterminer par exemple si elle correspond à un problème réel à résoudre,

- ou pour vérifier si elle est suffisamment similaire à la variante résolue par un autre article pour que les deux approches puissent être comparées en termes de performances.

La section suivante aborde le second objectif, qui consiste non plus à examiner les références bibliographiques une par une ou deux par deux, mais par groupes de références plus importants pour faire apparaître des classes de VRP plus ou moins étudiées par les chercheurs.

2.4.2.2 Classification, et estimation de l'intérêt des variantes de problème

Si le tableau 2.2 couvrait suffisamment largement la littérature du VRP, nous pourrions tirer des conclusions quant aux variantes les plus étudiées ou au contraire détecter les voies qui sont encore à explorer. Dans ce cadre, le raisonnement repose sur le décompte et la comparaison du nombre d'occurrences d'un paramètre ou d'un sous-ensemble de paramètres de la notation dans le tableau.

Par exemple, la comparaison des objectifs visés par les vingt-deux références citées montre que

- la distance totale parcourue (Tr:Dis) apparaît dix-sept fois
- le temps total de transport (Tr:T) n'apparaît que sept fois,
- le nombre de véhicules (Nve) n'apparait que six fois,
- les autres critères cités apparaissent moins de 5 fois.

Par conséquent, la distance totale parcourue est en très large majorité le critère minimisé.

Plus généralement, en regroupant les variantes par groupes de références ayant d'importantes parties communes dans leurs notations, il est possible de distinguer des classes de problèmes étudiés. Par exemple, (Labadi *et al*., 2008) (Ombuki *et al.*, 2006), (Le Bouthillier, 2000), (Berger *et al.*, 2001) (Bräysy,2003) et (Lau *et al.*, 2003) présentent neuf paramètres communs dans leur notation (**RP/Ser:T,Tr:Dis, Tr:T/Cap:Ve, Tw:C,TW:Dp/ NVe, Tr:Dis,** en gras dans la seconde colonne du tableau 2.2). Ils constituent ainsi un groupe de références que l'on pourrait désigner comme une classe de problèmes de tournées à fenêtres de temps (au dépôt et chez les clients), avec contrainte de capacité des véhicules et prise en compte du temps (de transport et de service), et minimisation du nombre de véhicules et de la distance totale parcourue.

Renouveler cette démarche pour constituer différentes classes puis comptabiliser et comparer ensuite l'effectif de ces classes mettrait en évidence les cas les plus étudiés jusqu'à présent et ceux pour lesquels il n'existe encore que peu d'approches de résolution, voire aucune.

Enfin, si pour chacune de ces utilisations, on rajoute une dimension temporelle, il est possible d'observer l'évolution des sujets de recherche considérés. Par exemple, classer les articles d'une classe par ordre chronologique permet

- de dater l'apparition de ce cas dans la littérature,
- d'identifier la dernière référence connue lui étant consacrée,

- d'identifier des tendances en termes de croissance ou décroissance de l'intérêt des chercheurs pour cette classe.

Naturellement, le format papier n'est pas idéal pour présenter ces différents usages. La forme électronique (a minima dans un tableur, pour effectuer des tris différents selon la recherche effectuée) permet d'effectuer ces travaux de classement, analyse, recherche puis synthèse plus facilement. Même si ces démarches sont présentées ici à partir de tableaux statiques et de taille réduite, elles ont vocation à être effectuées sur un outil reposant sur une base de données dont la structure correspond aux paramètres de la notation. Le développement de cet outil n'est encore qu'au stade de projet pour l'instant, mais nous espérons réunir les moyens nécessaires à sa mise en œuvre, pour ensuite pouvoir le mettre à la disposition de la communauté scientifique sous forme de portail web ou d'outil collaboratif.

Malgré cette difficulté technique, l'échantillon de références présentées dans le tableau 2.2 a permis de montrer que la notation remplit également son rôle dans le cadre du second objectif fixé. Elle permet

- de détecter quel est l'intérêt porté aux différentes variantes, en comptant les occurrences d'un sous-ensemble de paramètres,
- de regrouper les références bibliographiques par classes de problèmes, avec une précision plus fine que l'identification par sigle, en identifiant les articles dont les notations sont identiques ou quasi identiques.

Il est alors possible de chercher des liens, entre une variante (ou une de classe de problèmes) donnée et le type de méthodes de résolution ayant été développées pour la résolution. C'est ce troisième objectif qui est illustré dans la section 2.4.2.3.

2.4.2.3 Identification et comparaison des méthodes utilisées en fonction des variantes

Ajouter au tableau précédent une colonne « méthode de résolution » ouvre un nouvel angle de lecture et d'analyse de la littérature des problèmes de tournées. Celui-ci, à nouveau peut se décliner de deux manières :

- en conservant les classements précédents (par classes de variantes étudiées),
- en remplaçant ces classements par variantes de problèmes par des classements par méthodes de résolution utilisées.

Dans le premier cas, le but est d'analyser les liens entre les méthodes utilisées et les paramètres ou sous-ensemble de paramètres de la notation (c'est à dire les classes de problème considérées). Il peut s'agir par exemple de comparer les méthodes en termes

- de nature des méthodes utilisées en fonction de la classe considérée, voire même en fonction de la présence ou de l'absence d'une contrainte (par exemple, les fenêtres temporelles) ;
- de nombre d'occurrences : quelles sont les méthodes les plus (ou les moins) utilisées jusqu'à présent, pour quelles classes de problèmes, …

- d'évolution des méthodes employées au fil des ans (date d'apparition d'une méthode, tendances et évolutions du type d'approches proposées en fonction des classes considérées...)

Dans le second cas, la recherche d'information est en sens inverse. Il peut s'agir de déterminer pour un certain type de méthode

- s'il a déjà été utilisé pour résoudre un (ou plusieurs) problème(s) de tournées,
- à quelle(s) classe(s) de problèmes il a été appliqué
- quel est le sous-ensemble de références à étudier lorsque l'on débute soit même le développement d'une approche de ce type,

Le tableau 2.3 permet d'illustrer ces usages de la notation. Il présente les références des articles (colonne 1), la notation associée (colonne 2) et le type de méthode de résolution utilisé (colonne 3). Ceci permet d'analyser la littérature du VRP sous l'angle des types d'algorithmes développés pour le résoudre, comme le montrent les paragraphes suivants.

a. Types de méthodes de résolution utilisées

Etant donné la prépondérance des articles utilisant des algorithmes évolutionnaires (liée au sujet de la thèse), les 14 articles correspondants ont été exclus du raisonnement qui permet d'estimer quelles sont les méthodes les plus utilisées. En effet, l'intérêt particulier qui leur a été porté dans notre étude bibliographique les avantagerait fortement au détriment des autres types de méthodes. Le reste du tableau 2.3 montre que de nombreux articles décrits dans cette sélection reposent sur des heuristiques d'insertion (Dell'Amico *et al*., 2006) et (Liu et Shen, 1999) et/ou des métaheuristiques (Bräysy, 2003), en particulier la recherche tabou (Cordeau et Laporte, 2002), (Lau *et al.*, 2003) et (Alonso *et al.*, 2006).

Pour plus d'informations sur les approches du VRP par heuristiques et métaheuristiques, le lecteur pourra se reporter aux états de l'art de (Gendreau *et al*., 1997), (Toth et Vigo, 2002a), (Gendreau *et al*., 2003), (Bräysy et Gendreau, 2005a) et (Bräysy et Gendreau, 2005b) , (Gendreau *et al*., 2008).

Le tableau 2.3 permet également de raisonner par variantes :

- soit en observant les méthodes utilisées pour résoudre une variante particulière de problèmes. Par exemple, les articles comportant le paramètre TW:C dans leur notation, c'est à dire les variantes de problèmes avec fenêtres temporelles, ont été abordés par heuristiques d'insertion (Liu et Shen, 1999), par métaheuristiques (Bräysy, 2003)(Lau *et al.*, 2003), ou en utilisant les deux à la fois (Dell'Amico *et al.*, 2006);
- soit pour observer l'utilisation d'un type de méthodes donné. Ainsi, la méthode tabou a été utilisée pour traiter des variantes aux notations très variées : aussi bien pour le VRP de base (Cordeau et Laporte, 2002) que pour des problèmes à fenêtres de temps (Lau *et al*., 2003) ou des problèmes périodiques (Alonso *et al.*, 2006).

Référence	Notation	Type d'approche
(Cordeau et Laporte, 2002)	RP/NVe, Ser:T, Tr:Dis, Tr:T/ Cap:Ve, Aut:Ve/ Tr:Dis, Ser:T, Tr:T	Recherche tabou
(Alba et Dorronsoro, 2004)	RP/ NVe, Ser:T, Tr:Dis, Tr:T/ Cap:Ve, Aut:Ve/ Tr:Dis, Over:Cap, Over:Aut:Ve.	Algorithmes évolutionnaires
(Prins, 2004)	RP/ Nve, Ser:T, Tr:T/ Cap:Ve, Aut:Ve/ Tr:T	Algorithmes évolutionnaires
(Ho *et al.*, 2008)	RP/ Mdp, Ser:T, Tr:T/ Cap:Ve, MDpP/ Tr:T	Algorithmes évolutionnaires
(Najera, 2007)	RP/ Tr:Dis,/ Cap:Ve/ Tr:Dis	Algorithmes évolutionnaires
(Tavares *et al.*, 2003)	RP/ NVe, Tr:Dis/ Cap:Ve/ Tr:Dis	Algorithmes évolutionnaires
	RP/ NVe, Ser:T ,Tr:Dis, Tr:T/ Cap:Ve, Tw:C, Tw:Dp/ Tr:Dis	Algorithmes évolutionnaires
(Alvarenga *et al.*, 2007)	RP/ Nve, Ser:T,Tr:Dis, Tr:T/Cap:Ve, Tw:C,Tw:Dp/ Tr:Dis	Algorithmes évolutionnaires
(Labadi *et al.*, 2008)	RP/Ser:T, Tr:Dis,Tr:T/Cap:Ve, Tw:C,Tw:Dp/ Tr:Dis	Algorithmes évolutionnaires
	RP/ Ser:T, Tr:Dis,Tr:T/Cap:Ve, Tw:C,Tw:Dp/ NVe,Tr:Dis	Algorithmes évolutionnaires
(Ombuki *et al.*, 2006)	**RP/**NVe**,Ser:T,Tr:Dis,Tr:T/Cap:Ve,Tw:C,Tw:Dp/NVe,Tr:Dis**	Algorithmes évolutionnaires
(Le Bouthillier, 2000)	**RP/** NVe**,Ser:T**, **Tr:Dis, Tr:T/ Cap:Ve, Tw:C, Tw:Dp/ Nve, Tr:Dis**, Tr:T, WT, OT	Algorithmes évolutionnaires
(Berger *et al.*, 2001)	**RP/** NVe**,Ser:T**, **Tr:Dis, Tr:T/ Cap:Ve, Tw:C,Tw:Dp/NVe, Tr:Dis**, Over:TW, OT	Algorithmes évolutionnaires
(Bräysy,2003)	**RP/Ser:T,Tr:Dis, Tr:T/Cap:Ve, Tw:C,TW:Dp/ NVe, Tr:Dis** , WT	Recherche à voisinage variable
(Lau *et al.*, 2003)	**RP**/ Nve**, Ser:T**, **Tr:Dis, Tr:T / Cap:Ve, TW:C, TW:Dp**,NoSer/**Nve,Tr:Dis,** NoSer, Over:TW, OT	Recherche tabou
(Liu et Shen,1999)	RP/Het, Ser:T, Tr:Dis, Tr:T/ Cap:Ve, Tw:C,TW:Dp / Tr:Dis, OT	Heuristiques d'insertion
(Dell'Amico *et al.*, 2006)	RP/NVe, Het, Ser:T, Tr:T/ Cap:Ve,TW:C, Ve↔C / Tr:T, Ser:T, NVe,WT	Métaheuristique/ Heuristique d'insertion
(Pankratz, 2005)	RP/ Dp:W, (Li,B), Ser:T, Tr:T, Tr:Dis/ Cap:Ve, Tw:C, Tw:Dp, Aut:Ve, Paired, Prec/ Tr:Dis	Algorithmes évolutionnaires
(Jih et Hsu, 2004)	RP/Dp:W,(Li,B),ONet, NVe:1,Tr:T/Cap:Ve, Tw:C , NoR,Paired, Prec/ Tr:T,WT, Over:TW,Over:Cap	Algorithmes évolutionnaires
(Vianna *et al.*, 1999)	RP(MP)/NVe, Tr:Dis/ Cap:Ve,F/Tr:Dis	Algorithmes évolutionnaires
(Francis et Smilowitz, 2006)	RP(MP)/ NVe, Ser:T,Tr:T/ Cap:Ve, F, Spa:Min~Max / Ser:T, Tr:T, OT	Approximation continue
(Mingozzi, 2005)	RP(MP)/MDp,NVe, Tr:OT/ Cap:Ve, F, Comb/ OT	Programmation en nombres entiers
(Alonso *et al.*, 2006)	RP (MP)/ NVe, Het, Tr:Dis/ Cap:Ve, Aut:Ve, MT, F, Comb, Ve↔C/ Tr:Dis	Recherche tabou
(Ramdane Cherif, 2002)	RP /Dp:(B+Li) , A , C :(Li +B), Ser:T,Tr :T / Cap:Ve/ Ser:T, Tr:T	Algorithmes évolutionnaires

Tab. 2.3 – Application de la notation pour l'étude des méthodes de résolution

Le faible nombre de références du tableau, associé à la prépondérance des algorithmes évolutionnaires, ne permettent pas de dégager de conclusions quant à des évolutions au fil des années de recherche. En considérant les références utilisant les algorithmes évolutionnaires, on peut néanmoins observer que ce type de méthodes est appliqué au VRP depuis plus de 10 ans au moins, et que cela ne semble pas diminuer avec le temps.

Référence	**Codage**	**Croisement**	**Opérateurs d'amélioration ou de diversification**
(Prins, 2004)	indirect	Order Crossover Linear Order Crossover	recherche locale
(Jih et Hsu, 2004)	indirect	Order Crossover Uniform order-based Merge crossover #1 Merge crossover #2	Échange et inversion
(Ombuki *et a.l*, 2006)	indirect	Best Cost Route Crossover	Constrained Route Reversal
(Labadi *et al*.,2008)	indirect	Order Crossover	Recherche locale et Cost spacing rule
(Vianna *et al*., 1999)	direct	Croisement spécifique	Échange
(Le Bouthillier, 2000)	direct	Order Crossover Edge Recombination	Insertion
(Berger *et al.*, 2001)	direct	Insertion-based Insert in route neighborhood	Ant colony, Large Neighborhood Search-based, Edge exchange, Repair solution, Reinsert shortest Solomon Reorder customers
(Tavares *et al*., 2003)	direct	Avec TW: spécifique. Sans TW: générique	Insertion, swap, inversion et déplacement
(Alba et Dorronsoro, 2004)	direct	Edge Recombination	Insertion, échange ou inversion
(Pankratz, 2005)	direct	*group-oriented*	*group-oriented*
(Najera, 2007)	direct	Croisement générique	Insertion, swap, inversion et déplacement
(Alvarenga *et al*., 2007)	direct	Croisement spécifique	Random customer migration, Bringing the best customer, Re-insertion using stochastic PFIH, Similar customer exchange, Customer exchange with positive gain, Merge two routes, Reinserting customer and route partitioning
(Ho *et al*., 2008)	direct	Order Crossover	Inversion et heuristique modifiée
(Ramdane Cherif, 2002)	indirect direct	X1 Crossover Order Crossover Linear Order Crossover	Echange Move recherche locale

Tab. 2.4 – Synthèse des codages et opérateurs évolutionnaires des références étudiées

Enfin, le tableau 2.3 montre qu'une étude généralisée à tous les types de méthode existants pourra difficilement utiliser un niveau de détail élevé quant à la composition

interne des approches proposées dans chaque article (modèle utilisé, opérateurs développés, innovations proposées...). Par contre si l'on réduit les champs des observations, à un seul type de méthode par exemple, il est alors possible d'accroître le niveau de détail sans générer un travail gigantesque. La section suivante illustre cette démarche en s'appuyant sur le tableau 2.4.

b. Détail des méthodes de résolution utilisées en réduisant le champ des recherches

Le tableau 2.4 se focalise sur un seul type de méthodes, les algorithmes évolutionnaires. Pour les quatorze références correspondantes, il donne un résumé du type de codage (seconde colonne) et des opérateurs génétiques utilisés (types de croisement dans la troisième colonne, types de mutations dans la dernière colonne).

Pour pouvoir analyser le tableau en fonction des paramètres apparaissant dans les notations, et donc des classes de problèmes considérés, il aurait fallu ajouter également la colonne « notation » comme dans les tableaux 2.2 et 2.3. Mais à nouveau les limites du format papier ne le permettaient pas. Il est nécessaire ici d'utiliser les deux tableaux (2.2 et 2.4) simultanément pour réaliser ce travail. Sous forme électronique, ce problème serait résolu naturellement.

Les termes techniques cités qui relèvent de l'algorithmique évolutionnaire, en particulier la description détaillée des opérateurs, ont déjà été expliqués, pour certains, dans le chapitre 1 (principalement dans la présentation des méthodes heuristiques). La section 3.3 apporte des informations complémentaires, en particulier sur les opérateurs de croisement, auxquelles le lecteur pourra également se référer. Enfin, Il pourra trouver plus de références sur l'application des algorithmes évolutionnaires au VRP dans les états de l'art de (Bräysy *et al.*, 2005) et (Potvin, 2009).

L'une des premières étapes lors du développement d'un algorithme évolutionnaire consiste à choisir comment représenter les solutions de l'espace de recherche sous forme d'individus au sein d'une population, autrement dit, déterminer quel type de codage utiliser. Les paragraphes suivants débutent donc l'analyse des méthodes de résolution décrites dans le tableau 2.4. en examinant la répartition des articles entre les deux principales formes de codage existante : directe et indirecte.

Analyse du type de codage utilisé

Le tableau 2.4 fait apparaître une prépondérance du codage direct (10 occurrences) sur le codage indirect (5 occurrences). Dans le premier cas, il suffit d'interpréter simplement le contenu du chromosome pour le traduire en une solution du problème. En particulier, le chromosome décrit l'ordre de visite des clients mais également l'affectation des séquences de visite aux véhicules (répartition en tournées). Dans le second, le chromosome ne contient pas la définition complète d'une solution. Une procédure supplémentaire doit être appliquée pour construire cette dernière à partir des informations présentes dans le chromosome, comme (Prins, 2004). Le faible nombre de références dédié au codage indirect ne doit cependant pas être forcément interprété négativement. En effet ces références sont peu nombreuses mais ont toutes été publiées entre 2004 et 2008 sauf (Ramdane Cherif, 2002) qui a utilisé les deux codages. Par

contre, pour le codage direct, trois des références citées ont été publiées à des dates antérieures à 2004 (Le Bouthillier, 2000),(Berger *et al.*, 2001) et (Tavares *et al.*, 2003).

La différence de nombre d'articles entre codage indirect et codage direct traduit donc peut-être tout simplement le fait que les codages de type indirect ont été « inventés » plus tardivement.

Les individus ainsi codés subissent ensuite dans la majorité des cas une opération de croisement pour générer de nouveaux individus pour la génération suivante. La seconde phase d'analyse du tableau 2.4. examine donc cet aspect dans les paragraphes suivants.

Analyse des opérateurs de croisement utilisés

Le tableau 2.4 témoigne également de la grande variété des opérateurs de croisement utilisés dans la littérature. Le croisement le plus utilisé semble néanmoins être *Order Crossover* (OX), qui est un croisement classique, par ailleurs utilisé dans d'autres problèmes d'optimisation basés sur des permutations. Le tableau 2.4 en présente 6 occurrences, dont 5 à des dates antérieures à 2004. Une référence récente toutefois continue à l'utiliser (Ho *et al.*, 2008). Il existe toutefois des opérateurs de croisement développés plus spécifiquement pour les problèmes de tournées comme (Ombuki *et al.*, 2006) et (Alvarenga *et al.*, 2007).

Cette variété d'opérateurs de croisement sera encore soulignée dans le chapitre 3, qui détaille le fonctionnement de nombreux croisements rencontrés dans la littérature. Les paragraphes suivants, quant à eux, ré-itèrent une démarche d'analyse similaire à celle faite pour les croisements, mais pour les autres types d'opérateurs, utilisés par les algorithmes pour améliorer ou diversifier la recherche.

Analyse des opérateurs d'amélioration et de diversification utilisés

La mutation repose le plus fréquemment sur les principes d'insertion (8 occurrences), d'inversion (6 occurrences), et d'échange (9 occurrences), en particulier pour les variantes de base du VRP comme (Alba et Dorronsoro, 2004) et (Najera, 2007).

Mais de nombreuses autres heuristiques ont également été proposées comme opérateurs d'amélioration (diversification), en particulier des règles heuristiques comme (Alvarenga *et al.*, 2007), ou des métaheuristiques, comme les colonies de fourmis (Berger *et al.*, 2001) ou des approches de recherche locale comme (Labadi *et al.*, 2008).

L'analyse des approches évolutionnaires proposée jusqu'ici examinait l'usage des codages et opérateurs de façon globale, sans les lier aux variantes (classes de problèmes) étudiées par les auteurs. La section suivante est complémentaire, elle montre comment certains des paramètres de la notation ont une influence sur les algorithmes développés par les auteurs.

Liens entre variantes et méthodes de résolution

En termes de classes de VRP, les variantes avec fenêtres temporelles semblent avoir été les plus étudiées par algorithmes évolutionnaires. Le paramètre Tw:C apparaît ainsi dans 9 notations parmi les 14 établies. Mais tous les auteurs ne considèrent pas cette contrainte de la même manière. Dans (Le Bouthillier, 2000)(Ombuki *et al.*, 2006)(Alvarenga *et al.*, 2007)(Labadi *et al.*, 2008) c'est une contrainte dure (*Hard time*

windows). Dans ce cas, les véhicules peuvent arriver en avance chez un client mais doivent attendre que ce client soit prêt pour le servir. Bien que tous ces auteurs respectent strictement cette règle, seul (Le Bouthillier, 2000) fait apparaître explicitement le temps d'attente (w*aiting time* WT) comme objectif à minimiser. Dans le cas où les véhicules arrivent trop tard chez un client (après la limite haute de la fenêtre de temps de service, le client ne peut pas être servi, et par conséquent une telle solution est tout simplement rejetée pour cause d'infaisabilité. D'autres auteurs, comme (Berger *et al.*, 2001), considèrent un autre type de problème, appelé problème de tournées de véhicules avec contraintes de fenêtres de temps « molles » (*Soft time windows*). Dans ce cas, un retard est toléré chez le client (*delay time*). Il est indiqué dans la notation par Over:TW, et il fait partie des objectifs à minimiser. Ainsi, l'apparition de contraintes de fenêtres temporelles peut inciter les auteurs à inclure des éléments supplémentaires dans leurs algorithmes pour intégrer la notion de faisabilité des solutions.

L'objectif est de tolérer la génération de solutions infaisables (c'est-à-dire ne respectant pas la contrainte de fenêtre temporelle) puis d'ajouter

- lorsque c'est possible, une procédure de réparation des solutions infaisables pour les transformer en solutions respectant la contrainte,
- ou un objectif supplémentaire à minimiser, représentant en quelque sorte un « niveau d'infaisabilité » des solutions proposées. Dans ce cas, les auteurs peuvent opter pour le développement
  - d'une approche multi-objectif, dans laquelle la violation de contrainte est quantifiée par un objectif à minimiser (le retard chez les clients par exemple),
  - d'un algorithme parallèle dans lequel l'une des populations est utilisée pour minimiser la violation de contrainte, comme dans (Berger *et al.,* 2001),
  - d'un algorithme dont la fonction objectif inclut dans une agrégation les objectifs initiaux de la variante de problème étudiée et une ou plusieurs formes de pénalité représentant les violations de contraintes, comme dans (Jih et Hsu, 2004) qui ont ajouté une pénalité à la fonction objectif si un véhicule arrive en retard à l'un des sites à visiter.

Les contraintes de fenêtres temporelles ne sont pas les seules contraintes qui sont parfois considérées comme des contraintes « molles ». (Jih et Hsu, 2004) tolèrent également des surcharges des véhicules (Over:Cap) et (Alba et Dorronsoro, 2004) tolèrent de plus des dépassements en termes d'autonomie des véhicules (Over:Aut:Ve). Les deux articles traitent l'infaisabilité potentielle des solutions générées par l'inclusion d'une pénalité dans la fonction objectif.

L'examen du tableau 2.4 ne permet pas de conclure que l'apparition d'un paramètre dans la notation incite particulièrement les auteurs à choisir un type d'opérateur de croisement ou de mutation plutôt qu'un autre ou à développer des opérateurs spécifiques. Ce cas, ce présente ponctuellement, par exemple dans (Vianna *et al*., 1999), (Pankratz, 2005) et (Ho *et al.*, 2008).

Mais dans la majorité des cas, il semble que les auteurs utilisent plutôt des codages et opérateurs classiques, ou que l'adaptation des opérateurs est plutôt liée à la forme choisie pour l'implantation du codage (par exemple sous forme de plusieurs vecteurs plutôt que sous forme de séquence unique) qu'à l'apparition d'un paramètre particulier de la notation.

L'ensemble des observations faites sur les codages et opérateurs évolutionnaires dans cette section ont été possibles grâce à l'augmentation du niveau de détail utilisé pour l'application de la notation. Celui-ci était possible par la réduction de la portée de l'étude bibliographique à un type d'approche bien particulier. Néanmoins, ce niveau de détail était encore insuffisant pour réellement comparer les algorithmes en termes de performance. L'efficacité des algorithmes proposés a été succinctement présentée dans les commentaires des articles cités, mais sans qu'une comparaison objective ne soit réellement possible.

Pour franchir ce pas supplémentaire, il faut ajouter au tableau précédent la notion de résultats. Ceci est possible lorsque les auteurs utilisent des jeux de problèmes de référence connus tels que ceux de (Christofides *et al.*, 1979), (Christofides et Beasley, 1984), (Solomon, 1987), (Golden *et al.*, 1995), (Golden *et al.*, 1998) (Nanry et Barnes, 2000), (Li et Lim, 2001). La section suivante décrit ce travail.

c. Comparaison de performances des méthodes de résolution utilisées

Pour faire une comparaison des articles cités en termes de performances, il faut rajouter au tableau 2.4 des colonnes présentant les résultats obtenus par les auteurs sur un même jeu de *benchmarks*. Le tableau 2.5 présente 6 colonnes de ce type. C'est un extrait d'un document de travail plus complet, qui recense sous forme de tableur de nombreux résultats publiés dans la littérature, mais dont la taille ne permettait pas de l'inclure dans le mémoire de thèse de manière lisible. Il est accessible sous forme électronique sur le lien[5].

Le tableau 2.5 considère 5 des articles mentionnés dans le tableau 2.2. Il s'agit, pour quatre d'entre eux, des articles qui ont une notation très similaire (neuf paramètres communs) : (Ombuki *et al*., 2006), (Labadi *et al*., 2008), (Le Bouthillier, 2000), (Berger *et al*., 2001). Toutes ces approches, minimisent le nombre de véhicules puis la distance totale parcourue dans un problème de VRP à fenêtres temporelles, et ont été appliquées aux *benchmarks* de (Solomon, 1987).

De plus, l'une de ces références (Labadi *et al.*, 2008) présente une seconde approche minimisant uniquement la distance parcourue. Les deux algorithmes sont nommés MA1 pour le cas mono-objectif et MA2 pour le cas bi-objectif. Comme leur notation ne diffère que par la présence ou l'absence de Nve en tant qu'objectif, MA1 a été incluse dans le tableau, afin d'observer l'influence de cette différence. De plus, une cinquième référence, ayant une notation quasi identique à MA1, a également été ajoutée, pour confirmer ou infirmer les différences de résultats observées entre MA1 et MA2. Cela permet par ailleurs d'avoir une répartition égale dans le tableau entre codage direct (Le Bouthillier, 2000), (Berger *et al.*, 2001) (Alvarenga *et al.*, 2007) et codage indirect (Ombuki *et al.*, 2006), (Labadi *et al.* ,2008, MA1 et MA2), et d'inclure une publication récente pour le codage direct.

Le tableau présente les références des publications dans la première colonne, puis les résultats associés obtenus pour les différents groupes de *benchmarks* (désignés par C1, C2, R1, R2, RC1, RC2) du jeu de tests créé par (Solomon, 1987) dans les six colonnes suivantes. En effet, cet ensemble de problèmes de référence comprend au total 56 problèmes de VRP avec fenêtres temporelles. Chaque problème contient 100 clients et un dépôt central. L'objectif, dit « de Solomon », consiste à minimiser le nombre de véhicules en objectif principal, puis la distance totale parcourue en objectif secondaire.

[5] https://docs.google.com/fileview?id=0B4gItvSB0-LcYWOwNiY5NWUtODMwOS00MWI4LWFkY2EtYTM2ZTVkZWZlMzOz&hl=en

Ces problèmes peuvent être regroupés en 6 catégories en fonction de la loi de distribution des clients dans l'espace

- le type R correspond à une distribution aléatoire des clients dans l'espace,
- le type C correspond à une distribution des clients dans des regroupements (clusters),
- le type RC est une combinaison des deux modes précédents de distribution.

- la taille de la fenêtre de temps de chaque client
  - le type 1 met en place une fenêtre de temps étroite pour chaque client. Ceci implique un horizon de service court (la fenêtre de temps du dépôt central) ainsi que des tournées comportant peu de clients. La capacité des véhicules est également réduite,
  - le type 2 met en place des fenêtres de temps larges. L'horizon de service étant plus étendu, les tournées comprendront en principe plus de clients. La capacité des véhicules est également assez large.

De plus, dans ces *benchmarks*, le temps de service auprès de chaque client est fixé à

- 10 unités de temps pour les problèmes de type R et RC,
- 90 unités de temps pour les problèmes de type C.

Comme de nombreuses approches de résolution ont été appliquées dans la littérature aux 56 instances de VRP ainsi générées, il est difficile de faire des comparaisons instance par instance. (Bräysy et Gendreau, 2005b) et (Bräysy *et al.*, 2005) ont donc proposé d'utiliser deux indicateurs mieux adaptés pour classer les algorithmes proposés en fonction de leur performance : le nombre de véhicules cumulé (*cumulative number of vehicles* ou CNV) et la distance totale cumulée (*cumulative total distance* ou CTD) pour l'ensemble des 56 problèmes. Ce sont des valeurs moyennes obtenues en divisant ces deux indicateurs par le nombre d'instances qui apparaissent dans les colonnes C1 à RC2 du tableau 2.5. Dans chaque cellule, la valeur moyenne du nombre de véhicules est la première indiquée, celle de la distance totale parcourue est juste en-dessous.

Référence	C1	C2	R1	R2	RC1	RC2
(Ombuki *et al.*, 2006)	10 828,48	3 590,60	13,1 1204,48	4,5 893,03	13 1370,79	5,6 1025,31
(Labadi *et al.* ,2008) MA2	10 828,38	3 589,86	12,75 1188,01	3,09 920,86	12,37 1351,27	3,62 1087,18
(Le Bouthillier, 2000)	10 828,38	3 589,86	12,17 1209,27	2,82 965,91	11,50 1389,22	3,25 1143,70
(Berger *et al.*, 2001)	10 828,50	3 590,06	12,17 1251,40	2,73 1056,59	11,88 1414,86	3,25 1258,15
(Alvarenga *et al.*, 2007)	10 828,38	3 589,86	13,33 1196,8	4,64 899,9	13 1341,7	6 1015,9
(Labadi *et al.* ,2008) MA1	10 828,38	3 589,86	13,42 1184,16	5,36 879,51	13,13 1352,02	6,75 1009,37

Tab. 2.5 – Résultats en termes de nombres de véhicules et distances pour 5 articles étudiés

Les cellules grisées mettent en évidence les couples (nombre moyen de véhicules, distance totale parcourue en moyenne) non dominés. Ceci signifie qu'il n'existe pas dans la colonne consacrée à l'instance, d'approches proposant de meilleures valeurs pour les deux critères à la fois. A contrario, les cellules non grisées contiennent des couples de résultat dominées par au moins l'un des couples des cellules grisées. Par exemple, le résultat (10, 828.50) de (Berger *et al.*, 2001) pour la catégorie C1 est dominé par le couple (10, 828.38) des quatre autres algorithmes correspondant aux cellules grisées de la même colonne.

En se basant sur le nombre de cellules grisées associé à chaque algorithme, on obtient le classement suivant (du plus performant au moins performant) :

- (Le Bouthillier, 2000) et (Labadi *et al.* ,2008) (MA2) : 6 couples non dominés
- (Labadi *et al.* ,2008) (MA1) : 5 couples non dominés
- (Berger *et al.*, 2001) et (Alvarenga *et al.*, 2007) : 4 couples non dominés
- (Ombuki *et al.*, 2006) : 2 couples non dominés

En regardant de manière plus détaillée ce qui distingue les articles de classements identiques, on peut tirer les conclusions suivantes :

- Il n'existe pas de relations de dominance entre (Le Bouthillier, 2000) et (Labadi *et al.* ,2008), version MA2, car quelle que soit la catégorie de problèmes considérée, (Le Bouthillier, 2000) minimise mieux le nombre de véhicules, au détriment de la distance alors que pour (Labadi *et al.*, 2008) c'est l'inverse.
- La version MA1 de (Labadi *et al.* ,2008) présente le plus souvent des distances légèrement meilleures que celles de la version MA2, et par conséquent des nombres de véhicules supérieurs, sauf pour RC1, où MA2 présente une distance légèrement inférieure, ce qui explique qu'il est dominé dans ce cas.
- Comme pour les deux premiers du classement, (Berger *et al.*, 2001) et (Alvarenga *et al.*, 2007) sont à peu près équivalents mais le premier privilégie un peu plus le nombre de véhicules au détriment de la distance, et le second fait l'inverse. Mais ils semblent de plus être plus sensibles aux catégories de problèmes, (Berger *et al.*, 2001) étant plus performant pour des distributions plutôt aléatoires alors que (Alvarenga *et al.*, 2007) est plus efficace lorsqu'il existe des groupes (*clusters*). C'est peut-être la conjonction de ces deux phénomènes qui les pénalise par rapport aux premiers du classement.
- Enfin, même si la différence se fait souvent à quelques dixièmes de point près seulement, c'est (Ombuki *et al.*, 2006) qui paraît le moins performant, il ne présente de couples non dominés que pour les catégories de problèmes à fenêtres de temps larges, ce qui pourrait laisser supposer qu'il est mis en difficulté plus particulièrement par la contrainte de fenêtres temporelles.

Utiliser de plus les informations de la notation, issues du tableau 2.2, apporte des éléments supplémentaires à la réflexion. Les notations des articles considérés se distinguent essentiellement par le nombre et la nature des objectifs considérés. Parmi les quatre références qui minimisent l'objectif de Solomon (nombre de véhicules puis distance totale parcourue),

- deux n'ajoutent aucun objectif (Ombuki *et al.*, 2006) (Labadi *et al.* , 2008, version MA2).
- (Le Bouthillier, 2000) inclut trois objectifs supplémentaires qui sont le temps total de parcours, le temps d'attente, et le temps restant chez les clients avant la fin de la fenêtre de temps (représenté par OT dans la notation). Il agrège l'ensemble de ces critères dans une somme.
- Enfin, (Berger *et al.*, 2001) considèrent des contraintes molles et ajoutent en conséquence la minimisation du retard total et du nombre de clients servis en retard comme objectifs supplémentaires.

A l'inverse, (Labadi *et al.* , 2008, version MA1) et (Alvarenga *et al.*, 2007) comportent un objectif de moins puisqu'ils considèrent la distance comme seul objectif, sans chercher à minimiser le nombre de véhicules.

Il est donc intéressant d'analyser comment l'ajout ou le retrait d'objectifs influence les performances. Malheureusement, le classement précédent ne met pas en évidence une relation claire entre le nombre d'objectifs considéré et le rang des algorithmes dans le classement. En effet, les références qui respectent strictement l'objectif de Solomon se classent première et dernière. Les deux références qui ajoutent des objectifs supplémentaires sont première et troisième. Enfin, les références qui ont un objectif de moins sont deuxième et troisième. On observe simplement que

- les références comportant des objectifs supplémentaires minimisent mieux le nombre de véhicules,
- et les références qui ne considèrent pas le nombre de véhicules dans leur fonction objectif, présentent le plus souvent les nombres de véhicules les plus élevés et les distances les plus faibles, ce qui paraît logique, mais sans pour autant les reléguer en fin de classement.

L'impossibilité de dégager une relation claire entre objectifs et performances peut naturellement être liée à la faible taille de l'échantillon de références considéré. Identifier plus de références ayant des notations quasi identiques puis renouveler cette analyse permettraient peut-être de dégager des liens plus évidents, soit avec le nombre d'objectifs, soit avec leurs natures.

Mais il est possible aussi que l'influence de la fonction objectif n'apparaisse pas clairement parce qu'elle est mêlée à l'influence d'autres paramètres, en particulier le codage et les opérateurs utilisés dans les algorithmes. Pour analyser cet aspect, il est nécessaire de croiser les informations du tableau 2.4 et du tableau 2.5. Malheureusement, à nouveau, aucune différence vraiment marquée n'apparaît :

- En termes de codage,
  - les articles avec codage indirect se classent premier, deuxième mais également dernier,
  - alors que les références à codage direct occupent les deuxième et troisième rangs du classement.
- En ce qui concerne les opérateurs, les opérateurs et approches hybrides les plus sophistiqués ne montrent pas une efficacité supérieure. Ainsi,

- (Le Bouthillier, 2000) qui n'utilise que des opérateurs relativement classiques figure en tête de classement au même titre que l'approche mémétique de (Labadi *et al.*, 2008), et deux rangs devant (Berger *et al.*, 2001) dont l'approche hybride inclut des colonies de fourmis et méthodes de voisinage.
- L'opérateur de croisement OX apparaît dans les deux références qui sont les plus efficaces, et paraît donc efficace avec les deux types de codage.
- les opérateurs d'amélioration ou de diversification sont trop variés pour dégager une relation entre cet élément et le classement des algorithmes. La recherche locale apparaît aux classements un et trois, mais (Le Bouthillier, 2000) et (Alvarenga *et al.*, 2007) occupent les mêmes rangs en utilisant des heuristiques plus simples.

Ainsi, la section 2.4.2.3 a illustré les trois niveaux de détails utilisables pour analyser la littérature scientifique du VRP sous l'angle des approches de résolution utilisées, en liant les paramètres de la notation :

- aux types de méthodes utilisées, toutes approches confondues,
- aux types de codages, opérateurs de croisement et opérateurs d'amélioration ou de diversification, en limitant l'étude aux algorithmes évolutionnaires,
- aux performances des approches évolutionnaires de manière globale, puis en termes de codage et d'opérateurs.

Cette démarche a permis d'identifier quelques caractéristiques de l'échantillon de références considéré, comme

- la prépondérance de certains types de méthodes, en particulier les heuristiques et métaheuristiques,
- l'utilisation plus fréquente de codages de type direct, mais peut-être liée au fait que les codages de type indirect sont moins anciens,
- la grande diversité des opérateurs de croisement, amélioration et diversification, qui ne permet pas de mettre en évidence la supériorité de l'un d'entre eux, si ce n'est peut-être une utilisation un peu plus fréquente du croisement OX et des principes d'insertion, inversion et échange pour les autres opérateurs,

En termes de performance, en réduisant l'étude à quelques articles ayant des notations quasi-similaires, il a été possible de montrer le lien entre l'apparition d'un objectif supplémentaire (le nombre de véhicules) en tant qu'objectif, et le type de solutions retournées par les algorithmes. Il a été vérifié que, très logiquement, ces derniers fournissent de meilleurs compromis entre le nombre de véhicules et la distance parcourue lorsque le nombre de véhicules apparaît explicitement. Par contre, cela n'a pas permis de mettre en évidence la supériorité de certaines approches sur d'autres, en particulier des algorithmes basés sur des opérateurs relativement classiques sont à égalité dans le classement avec des approches mêlant principes évolutionnaires et recherches locales ou colonies de fourmi. De même, l'analyse détaillée en termes d'opérateurs n'a pas permis de dégager de conclusions claires à ce sujet.

Ceci est sans doute lié au nombre limité d'articles présents dans l'échantillon considéré, surtout pour la dernière phase, puisque la nécessité de comparer des articles

ayant des notations quasi identiques a encore réduit ce nombre. Trouver un groupe de références à notations similaires et utilisant les mêmes jeux de *benchmarks*, représente un travail bibliographique détaillé et fastidieux, qui n'a pu être complètement mené ici. Mais la démarche illustrée est généralisable. La suite de ce travail permettra peut-être de dégager plus clairement des conclusions en étoffant l'échantillon d'articles sous étude.

Mais il existe une autre explication possible, également très plausible. En effet, la performance d'un algorithme évolutionnaire est le fruit de la combinaison de plusieurs mécanismes (codage, sélections, croisement, mutation, …) dont les influences sur les résultats ne sont pas indépendantes. Analyser la performance uniquement à travers des informations disponibles dans la littérature demande de réunir et étudier un très grand nombre d'articles, d'autant plus grand quand la diversité des variantes, approches et opérateurs décrits augmente. Il faudrait par exemple qu'un même opérateur, tel que le croisement OX, apparaisse de nombreuses fois, combiné à des codages, sélections, et autres opérateurs divers, pour essayer de déterminer statistiquement s'il est plus efficace, et si cette efficacité est conditionnée par des choix spécifiques pour ces éléments.

Une démarche expérimentale, dans laquelle la conception de l'algorithme est plus maîtrisée, est sans doute plus appropriée pour cette analyse de performances. Il est alors possible de multiplier le nombre de tests, pour un certain nombre de combinaisons des choix possibles, tant pour le codage et les sélections que pour les opérateurs, puis d'analyser statistiquement les résultats pour essayer de dégager quelles sont les combinaisons les plus efficaces.

2.5 Conclusion

Ce chapitre s'est principalement intéressé aux problèmes posés par l'identification par sigle des problèmes de tournées dans la littérature. Dans un premier temps, il a illustré les difficultés que peuvent rencontrer les chercheurs lors de leurs études bibliographiques notamment pour identifier clairement les variantes traitées dans les articles, et ainsi délimiter la part de littérature scientifique la plus pertinente à exploiter.

Il a proposé un système de notation plus détaillé que les sigles. Ce système de notation s'appuie sur un schéma en quatre champs, $\pi/\alpha/\beta/\gamma$ pour décrire respectivement le contexte, les ressources (fixes ou mobiles) et les demandes, d'autres contraintes que celles propres aux ressources ou aux demandes et enfin les objectifs à optimiser. Il repose donc sur une structure standard, telle que celles classiquement utilisées en ordonnancement de la production.

De plus, les valeurs des paramètres inclut dans les champs ont été choisies, si possible, pour favoriser leur mémorisation, soit parce que ce sont des abréviations, soit parce qu'elles correspondent à des préfixes ou suffixes déjà utilisés dans l'identification par sigles.

De plus, la notation proposée se construit ou se décrypte, pour un problème donné, à l'aide de règles simples et systématiques. Pour obtenir le niveau de détail requis pour une identification claire des variantes, la notation comporte néanmoins un nombre important de champs, et peut paraître lourde à utiliser.

Un travail a été fait pour tenter de limiter ce nombre en ne conservant que les paramètres les plus pertinents, et dont l'absence nuisait à la description. En particulier,

lors de la comparaison des résultats fournis par deux algorithmes, l'absence ou la présence d'une contrainte peut changer la forme et les limites de l'espace des objectifs. Si cette contrainte n'apparaît pas dans la notation, les articles paraitront comparables, alors que certaines solutions retournées par l'un des algorithmes ne sont pas atteignables par l'autre du fait de cette contrainte. Ce fut l'un des critères essentiels pour la sélection des paramètres conservés dans la notation.

Pour faciliter son utilisation, ce système de notation a vocation à être mis à disposition de ses utilisateurs potentiels sous forme électronique. Typiquement, un portail collaboratif ou un service web permettant de décrire un problème de VRP pour obtenir la notation associée doit être développé. A l'inverse, ce système doit inclure une base de données des références bibliographiques ainsi notées, et un moteur de recherche pour pouvoir ensuite identifier les articles correspondant à une requête elle aussi basée sur un ou plusieurs des paramètres de la notation.

Comme ce développement n'a pas pu être mené dans le cadre de cette thèse, le chapitre s'est efforcé d'illustrer les diverses utilisations de la notation correspondantes malgré les difficultés techniques induites par le format papier. Il en ressort que les usages de ce système de notation sont multiples, qu'il peut générer des apports indéniables pour la communauté scientifique.

Néanmoins, il montre également ses limites, du moins en ce qui concerne l'analyse de la littérature. Notamment, lorsque le niveau de détail augmente. Etant donné la grande diversité des variantes de problèmes, des méthodes de résolution, et des types d'opérateurs utilisés, il est difficile de réunir suffisamment d'informations pour mener une analyse faisant émerger des conclusions claires. Ce chapitre n'a donc pas pu démontrer efficacement la supériorité d'un type d'algorithme, ou d'un opérateur particulier, ni de manière générale, ni en fonction des paramètres présents dans les notations (autrement dit des variantes de problèmes).

Pourtant établir une relation entre certains paramètres et l'efficacité des opérateurs pourrait permettre de guider la conception des algorithmes en fonction des caractéristiques de la variante de problème à résoudre. Le chapitre 4 ne va pas jusqu'à établir de telles règles, mais il donne un aperçu de la démarche pour un problème de VRP simple, correspondant à la notation $\pi/\alpha/\beta/\gamma$. Il illustre l'analyse statistique de l'efficacité de différents opérateurs pour cette variante du problème. Il resterait à renouveler la même démarche pour une variante ne se distinguant de celle-ci que par un paramètre (la présence de fenêtre temporelle par exemple), pour voir si l'apparition de cette contrainte supplémentaire influence fortement les résultats de l'analyse.

Chapitre 3

Estimation expérimentale de l'efficacité des opérateurs évolutionnaires, méthodologie

Sommaire

3.1 Introduction

La démarche présentée ici s'inscrit dans un projet plus vaste dont l'objectif est de guider la conception d'algorithmes évolutionnaires en fonction de la variante de problème de tournées à résoudre. La figure (3.1) en décrit les grandes lignes, elle fait apparaître 3 grands éléments de base :

- la notation proposée,
- une bibliothèque recensant les divers mécanismes évolutionnaires existants,
- une base de règles établissant des relations entre notation et sous-ensembles d'éléments de la bibliothèque.

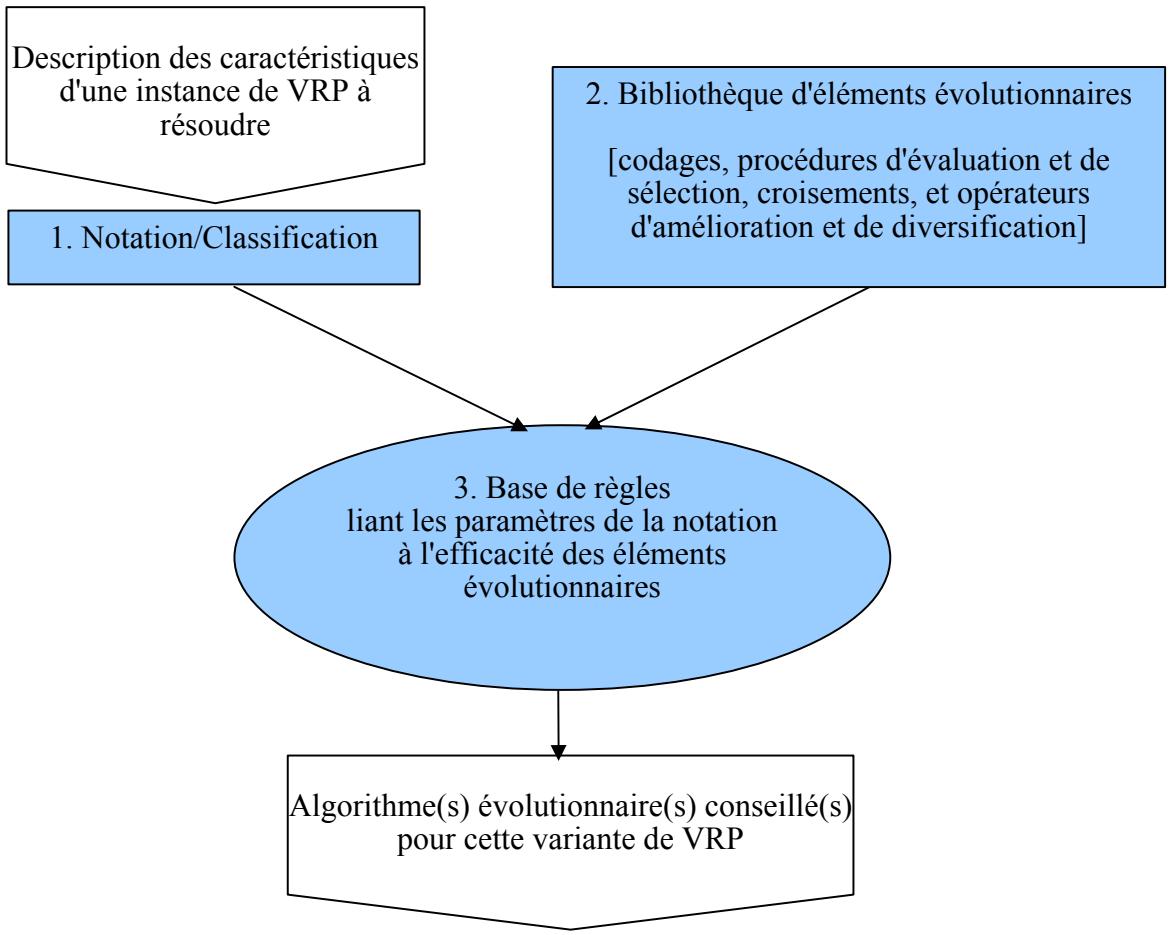


Fig. 3.1 – Illustration du projet global d'aide à la conception d'algorithmes

La notation et son utilisation ont fait l'objet du chapitre 2. Ce chapitre est plus spécifiquement consacré aux deux autres éléments. La section 3.2 illustre le recensement des mécanismes évolutionnaires existants dans la littérature pour la constitution de la bibliothèque d'éléments évolutionnaires. Elle détaille en particulier, après avoir rappelé les principes de base des algorithmes évolutionnaires, le

fonctionnement de plusieurs opérateurs de croisement et de mutation. La section 3.3, quant à elle, décrit la méthodologie proposée pour estimer l'efficacité de ces différents éléments pour la résolution d'une variante de VRP par expérimentation. Un exemple de mise en œuvre de cette méthodologie sera présentée dans le dernier chapitre du mémoire, qui décrit la campagne de tests réalisée et analyse ses résultats.

3.2 Algorithmes évolutionnaires : rappels et description de mécanismes de la littérature

Le but de cette section est de décrire plusieurs types d'éléments évolutionnaires rencontrés dans la littérature. L'ensemble de ces descriptions sera utilisé pour définir une base d'éléments possibles pour concevoir un algorithme évolutionnaire dédié à la résolution d'une variante du VRP. La base finale devra réunir le plus grand nombre de catégories possible pour les codages, les procédures d'évaluation, les mécanismes de sélection, les croisements et les opérateurs d'amélioration et de diversification. Au stade actuel, le recensement présenté ici s'est limité à :

- cinq types de codage,
- deux types de création de population initiale,
- trois types d'évaluation des individus,
- cinq types de sélection des parents,
- vingt-deux types de croisement,
- six types de mutation,
- quatre types de stratégie de remplacement pour la constitution de la génération suivante,
- trois types de critères d'arrêt.

Ce travail est une première pierre pour la constitution de la base. Il a également contribué au choix des éléments retenus pour la conception des versions d'algorithmes comparées dans les tests du chapitre 4. La section présente ces différents éléments dans l'ordre où ils interviennent dans la conception d'un algorithme évolutionnaire, ce qui permet de rappeler synthétiquement les concepts fondamentaux de ce type d'algorithmes.

3.2.1 Représentation des solutions ou codage

Dans un algorithme évolutionnaire, les solutions sont représentées par leur génotype (encore appelé chromosome ou individu), qui s'exprime sous la forme d'un phénotype. Le génotype se compose de plusieurs gènes, où chaque gène a une valeur possible (allèle) et une position spécifique dans son chromosome (locus). Le génotype représente le codage tandis que le phénotype représente la solution elle-même. Ainsi, dans le cas d'un **codage direct**, le génotype et le phénotype sont similaires. Le génotype contient toutes les informations nécessaires à la définition complète du phénotype. Par contre, ils présentent des différences plus importantes dans le cas d'un **codage indirect**.

Généralement, le génotype contient des informations qui sont ensuite interprétées par une procédure pour construire un phénotype associé. Le phénotype résultant dépend donc fortement de la procédure utilisée. Celle-ci peut être très simple, comme une heuristique, ou plus sophistiquée, comme une métaheuristique ou une méthode exacte. Très souvent, elle inclut un mécanisme pour prendre en compte une ou plusieurs contraintes et éviter de générer des solutions qui ne les satisferaient pas. Malgré cette procédure supplémentaire de traduction du génotype en phénotype, le codage indirect n'est pas forcément plus coûteux que le codage direct. En effet, le codage direct génère, souvent, des solutions infaisables (c'est-à-dire ne respectant pas toutes les contraintes du problème), et une procédure de réparation ou un mécanisme de pénalisation doivent ensuite être appliqués. La suite de cette section présente quatre types de codage, en détaillant plus particulièrement le codage par permutations qui est le plus utilisé pour les problèmes de tournées. Les différences entre codage direct et codage indirect seront plus précisément illustrés dans ce cas.

Pour représenter les solutions dans la population, on distingue quatre types de représentations (génotypes) différentes :

- la représentation binaire, lorsque la solution peut être représentée par une chaîne de bits (nombres binaires), où chaque gène peut prendre seulement les valeurs 0 ou 1. (Vianna *et al.*, 1999), par exemple, utilise une chaine binaire pour représenter les jours de visite d'un client au cours d'un horizon de planification donné. 0 signifie que ce client ne sera pas desservi ce jour-là, 1 qu'il sera visité.
- la représentation entière ou réelle : cette représentation a été introduite initialement pour les Stratégies d'Evolution (Schwefel, 1977), mais son utilisation s'est étendue rapidement aux autres types d'algorithmes évolutionnaires. Là, l'espace de recherche est l'espace des entiers ou celui des réels, dans lesquels les variables ont des intervalles de définition en fonction des contraintes du problème. Chaque individu (solution) se compose d'un (ou plusieurs) vecteur de valeurs entières ou réelles. Les valeurs entières, par exemple, peuvent être utilisées pour représenter le numéro de véhicule auquel un client a été affecté. Ce nombre est positif et inférieur au nombre de véhicules disponibles dans la flotte si ce dernier est limité (Nve dans le sous-champ α2 de la notation).
- la représentation sous forme de structure arborescente : c'est une manière de représenter la nature hiérarchique d'une structure sous une forme graphique. Le terme d' « arbre » est alors employé, parce que la structure ressemble à un arbre, mais dont la racine (un nœud unique) est généralement en haut, et les feuilles (ensemble de nœuds qui représentent en quelque sorte les «enfants» de la racine) en bas. Tout nœud sans parent s'appelle une racine (*root*), et peut être considéré comme un nœud de début. Les nœuds qui n'ont pas d'enfants s'appellent des feuilles (*leaves*) et les lignes reliant les nœuds s'appellent des branches. Un exemple donné par (Golden *et al.*, 2008), pour un problème de tournées avec contrainte de capacité, est montré dans la figure (3.2). Il représente chaque tournée de véhicules par un arbre.

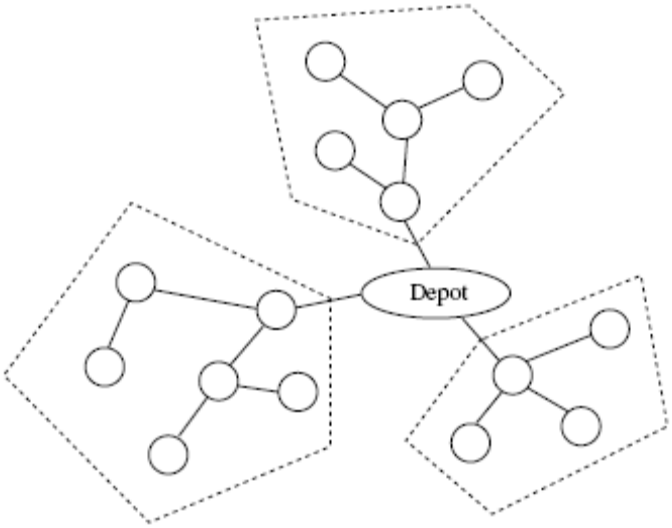


Fig. 3.2 – Présentation de structure arborescente pour le VRP

- la représentation de permutation : ici, les chromosomes contiennent typiquement une séquence de gènes, dans laquelle l'ordre est significatif, et où les gènes sont des nombres entiers ou des lettres. Un exemple de ce type de codage pour le problème de VRP est la chaîne (2 5 1 4 7 8 6 3 9). Dans ce cas, la solution (chromosome) est représentée par une chaîne entière de longueur 9, où 9 est le nombre de clients. Chaque gène dans ce chromosome indique l'identification d'un nœud (nombre ou lettre) assigné à l'origine à un client, tandis que l'ordre des gènes dans la chaîne du chromosome donne l'ordre de visite des clients. Mais aucun séparateur n'est utilisé pour indiquer le début ou la fin d'une tournée dans cette solution. Alors, il est impossible de distinguer les tournées de véhicules avant que le chromosome ne passe par une étape complémentaire de découpage, qui va construire les tournées et construire ainsi la solution finale complètement définie. Ce type de codage s'appelle codage indirect. L'avantage le plus important de ce codage est la simplicité avec laquelle les opérateurs génétiques peuvent être appliqués sur un chromosome représenté par ce codage. Le découpage en tournée prend généralement en compte la contrainte de capacité, et les contraintes de fenêtres de temps si le problème en inclut. (Prins, 2004) par exemple, proposent une procédure de découpage (nommé *split*) qui fournit une solution optimale respectant les contraintes de capacité en résolvant un problème de plus court chemin dans un graphe associé au chromosome. (Labadi *et al.*, 2008) adaptent cette procédure au cas du VRP incluant des fenêtres temporelles.

  Dans le cas d'un codage direct, toute l'information de la solution est présente dans le chromosome. Un exemple de ce codage pour le VRP est donné dans (Alba et Dorronsoro, 2004) : chaque permutation contient à la fois les clients et des séparateurs de tournées, voir figure (3.3). L'exemple cité montre une solution de VRP avec 10 clients et 4 véhicules, représentée sous forme de permutation. Les valeurs [0,…, 9] représentent les clients tandis que [10,…, 12]

sont les séparateurs de tournées. La tournée (1) commence au dépôt, visite les clients 4-5-2 (dans cet ordre), et retourne au dépôt. La tournée (2) va du dépôt aux clients 0-3-1 et retourne. Le véhicule de la tournée (3) démarre au dépôt et visite les clients 7-8-9. En fin, dans la tournée (4), seul le client 6 est visité. Cette représentation permet de représenter des véhicules auxquels aucune tournée n'est affectée, lorsque le nombre de véhicules du problème est fixe et supérieur à 1. Ce cas est représenté par l'apparition d'une tournée vide dans le chromosome, en mettant deux séparateurs des tournées sans clients entre eux.

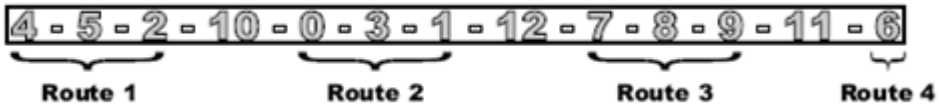


Fig. 3.3 – Individu représentant une solution de VRP

Les séparateurs de tournées peuvent aussi prendre d'autres valeurs comme par exemple dans (Ho *et al.*, 2008) qui ont utilisé une représentation sous la forme (0 2 4 1 0 3 6 5 0) de leur solution de VRP. Ici, le dépôt est noté (0) et le problème comporte 6 clients (nombres de 1 à 5). En utilisant la valeur de dépôt à chaque début et fin de tournée, on peut distinguer les tournées de véhicules. Généralement, l'avantage du codage direct est d'économiser le temps de traitement utilisé par l'étape de découpage pour construire les tournées dans le cas du codage indirect. Ses inconvénients principaux sont les difficultés que posent les séparateurs pour l'application d'opérateurs évolutionnaires, même classiques. Pour un croisement par points par exemple (décrit dans les premiers paragraphes de la section 3.2.5.1), il faut déterminer si les séparateurs sont traités comme les autres gènes lors de la recopie des gènes des parents dans les enfants. Si c'est le cas, même lorsque les parents respectent les contraintes de capacité des véhicules, les enfants construits ne le feront pas forcément. Il faut alors appliquer une procédure de réparation de manière à déplacer les séparateurs pour résoudre ce problème, dont le temps de traitement peut être équivalent à celui du temps de découpage en codage indirect. Une autre solution consiste à ajouter une pénalisation dans la fonction objectif pour les solutions qui ne respectent pas les contraintes. Mais ceci augmente la taille de l'espace de recherche considéré, puisque des solutions non faisables sont conservées. Il est difficile de trouver la forme de pénalisation adaptée (voir section 3.3.3). En conséquence, l'algorithme risque de trouver plus difficilement le sous-ensemble de solutions faisables intéressantes.

Ainsi, le simple choix de la représentation des solutions par l'algorithme évolutionnaire repose déjà sur un nombre important de décisions. De plus, il peut avoir une influence non négligeable sur les choix à faire pour le reste de la conception de l'algorithme. En particulier, il aura un impact sur la manière d'évaluer la qualité de chaque solution (vis à vis du ou des objectifs du problème de VRP) ainsi représentée. La section suivante présente plusieurs formes d'évaluation possibles.

3.2.2 Evaluation de chaque solution

Chaque fois qu'un nouvel individu a été créé, il faut lui associé une valeur (appelée fitness, force, adaptation ou encore fonction d'évaluation) qui mesure la qualité de la solution. Cette valeur sera utilisée par les processus de sélection (sections 3.2.3 et 3.2.6) pour favoriser les individus les mieux adaptés, autrement dit les meilleures solutions au problème. L'évaluation représente donc la performance de l'individu vis-à-vis du problème à résoudre. De ce fait, cette fonction est en rapport avec la fonction objectif de ce dernier.

Dans le cas mono-objectif, la fonction d'évaluation est généralement

- l'objectif du problème lui-même,
- ou une fonction de cet objectif.

Mais il est aussi possible d'incorporer dans la fonction d'évaluation d'autres composantes en rapport avec le processus d'évolution. Par exemple, une pénalisation sur les contraintes non satisfaites, peut être ajoutée. Dans ce cas, cet « objectif » supplémentaire à minimiser peut-être traité sous l'une ou l'autre des formes de fonction d'évaluation utilisées dans le cas multi-objectif, décrit dans le paragraphe suivant.

Lorsque la fonction d'évaluation doit intégrer plusieurs objectifs, éventuellement contradictoires, il existe trois manières de la construire :

- dans le cas le plus simple, c'est une fonction d'agrégation. Elle peut être plus ou moins complexe selon que les auteurs additionnent les critères entre eux sans précaution supplémentaire ou non. Certains auteurs utilisent des fonctions particulières soit pour « normaliser » les critères avant de les additionner (Boussedjra *et al.*, 2006) soit pour inclure des coefficients correspondant au niveau d'importance accordé à chaque critère. Les auteurs définissent ainsi en quelque sorte une priorité entre les critères.
- les auteurs peuvent également considérer les critères successivement, dans un certain ordre. Typiquement, l'objectif de (Solomon, 1987), utilisé pour les problèmes avec fenêtres de temps, consiste à minimiser le nombre de véhicules, puis la distance totale parcourue. Comme les *benchmarks* de (Solomon, 1987) figurent parmi les plus utilisés, ce cas est très fréquemment rencontré dans la littérature.
- enfin, certaines approches utilisent la notion de dominance au sens de Pareto (Pareto, 1896), sur laquelle sont basés de nombreux algorithmes évolutionnaires multi-objectifs. (Deb, 2002) est un excellent état de l'art des algorithmes génétiques multi-objectifs, qui sont l'objet d'une très forte activité de recherche. Le lecteur pourra s'y référer pour plus de détails. En utilisant le principe de dominance de Pareto, la comparaison entre deux solutions s_1 et s_2 peut mener à trois conclusions selon les cas :
  - soit s_1 présente des valeurs des critères toutes égales ou meilleures que celles de s_2, avec au moins une valeur strictement meilleure. On dit alors que s_1 domine s_2.
  - soit on a la relation inverse et s_2 domine s_1.
  - soit on ne peut établir aucune relation de dominance. s_1 et s_2 sont alors considérées comme équivalentes en termes de qualité.

Une solution est dite Pareto-optimale s'il n'existe aucune autre solution qui soit meilleure qu'elle pour tous les critères. Différentes méthodes peuvent alors être utilisées pour attribuer une force à chaque solution, par exemple :

- en comptabilisant le nombre de solutions qu'elle domine,
- en comptabilisant le nombre de solutions qui la dominent,
- ou encore en partageant la population par groupes de solutions équivalentes au sens de la dominance, nommés front de Pareto, puis en affectant des « forces » différentes aux solutions en fonction du front auxquelles elles appartiennent (méthodes dites de *ranking*) (Deb, 2002) (Coello *et al*., 2007).

Comme le choix du mode de représentation des solutions, celui du mode d'évaluation a un impact fort sur la manière dont l'algorithme va parcourir l'espace de recherche, et donc sur les solutions qu'il retournera finalement. Ce choix a également des conséquences sur le choix des autres mécanismes utilisés dans l'algorithme. Une évaluation multi-objectifs dans laquelle l'un des objectifs représente une violation de contrainte à minimiser, par exemple, peut permettre de se passer de procédure de réparation des solutions non faisables générées (que ce soit au moment de la création de la population initiale, ou par utilisation des opérateurs de recombinaison, amélioration et diversification). Après ces deux choix cruciaux (codage et évaluation), l'étape suivante consiste à définir comment l'algorithme génère de nouveaux individus. La section suivante distingue trois approches pour produire une population initiale.

3.2.3 Création de la population initiale

Dès la création de la population initiale, le fonctionnement de l'algorithme est conditionné par l'équilibre atteint entre qualité et diversité des solutions dans la population. Un réflexe naturel consiste à penser que l'algorithme trouvera d'autant plus rapidement des optima si la population initiale contient déjà des solutions quasi-optimales. Toutefois, des individus dont la qualité est très supérieure au reste des individus de la population peuvent à terme piéger l'algorithme dans des optima locaux. Beaucoup plus forts, ils sont beaucoup plus utilisés pour générer des enfants, et de génération en génération la population s'appauvrit en contenant des individus de plus en plus similaires, tous descendants de ces individus dominants. Maintenir une diversité suffisante dans la population permet d'éviter de rester piégé dans les optima locaux. Mais à l'inverse, une excellente diversité d'individus qui seraient tous de forces quasi-équivalentes conduirait à une recherche très aléatoire. Pour la génération de la population initiale, les auteurs sont donc partagés entre

- une construction totalement aléatoire de solutions,
- la construction des individus à l'aide d'heuristiques ayant déjà fait leurs preuves pour résoudre les problèmes de tournées,
- une procédure réunissant les deux principes, dans laquelle une part de la population initiale est tout d'abord générée aléatoirement, puis une ou plusieurs heuristiques sont utilisées pour compléter la population jusqu'à une taille fixée.

L'utilisation d'heuristiques peut également être utile pour générer des solutions faisables lorsque les contraintes du problème sont telles que la génération aléatoire construit des solutions ne respectant pas les contraintes dans la majorité des cas.

La taille de la population, quant à elle, est un paramètre de réglage de l'algorithme, souvent déterminé de façon empirique. Elle peut également être variable au cours de l'exécution. Elle doit être en concordance avec la longueur des chromosomes et les objectifs de la recherche. Certains auteurs soutiennent qu'une grande taille est importante au début de la recherche (De Jong, 2007), car tout au début on traite typiquement un espace complexe, plein de "pièges". Il est possible aussi de grouper les solutions dans des sous-populations, ce qui peut aider à maintenir la diversité et améliorer la performance (Cantù-Paz, 1997) (Tsutsui *et al.*, 1997) (Wineberg et Chen, 2004).

Si la procédure de construction de la population initiale a permis d'obtenir un équilibre adapté entre qualité et diversité des individus, le processus de sélection de candidats (parents) pour générer de nouvelles solutions (enfants) à l'aide d'opérateurs évolutionnaires, doit fonctionner correctement et permettre d'améliorer la qualité des solutions tout en préservant la diversité de générations en générations. Différents processus de sélection sont présentés dans la section 3.2.4.

3.2.4 Sélection pour reproduction (ou sélection de parents)

La sélection détermine combien de fois un individu participe à la reproduction ou sera reproduit en une génération. Les individus ayant les meilleures performances sont sélectionnés plus souvent que les autres. Les individus sélectionnés, appelés parents, sont ensuite disponibles pour la phase suivante, dite de reproduction (variation). Celle-ci consiste à appliquer des opérateurs de variation sur des copies des individus sélectionnés pour en engendrer de nouveaux. Dans ce qui suit, différents types de sélection sont expliqués.

3.2.4.1. Sélection proportionnelle

Plusieurs stratégies de sélection des individus sont basées sur le principe de sélection proportionnelle. Autrement dit, la chance qu'a un individu d'être sélectionné est proportionnelle à sa valeur de fitness. Ainsi,

- dans la sélection par roulette (***Roulette Wheel Selection*** ou RWS), introduite par (Goldberg, 1989), la population est représentée comme une roue de roulette, où chacun individu est représenté par une portion qui correspond proportionnellement à sa valeur de fitness. La sélection d'un individu se fait en tournant la roue en face d'un pointeur fixe. Cette procédure est répétée jusqu'à constitution complète de la population des parents. L'un des inconvénients de ce type de sélection est de choisir presque toujours le même individu s'il en existe un bien meilleur que les autres, ce qui cause une perte de diversité dans la population.
- l'échantillonnage universel stochastique (***Stochastic Universal Sampling*** ou SUS) proposé par (Baker, 1987) utilise aussi une roulette partagée en autant de portions (individus), proportionnellement aux valeurs de fitness. Mais cette fois, les individus sélectionnés sont désignés par un ensemble d'indicateurs

équidistants où une rotation de la roue de roulette sélectionne tous les parents simultanément. La figure (3.4) illustre la différence entre RWS et SUS.

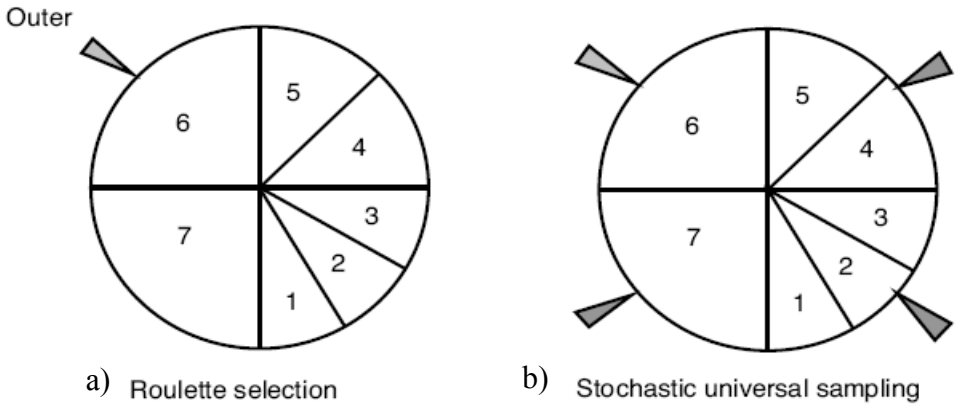


Fig. 3.4 – a) sélection par roulette, b) exemple de sélection SUS pour choisir 4 individus

- la sélection, nommé (***Remainder Stochastic Sampling*** ou RSS), utilise une procédure de traitement sur la valeur de fitness des individus, qui a été illustrée par (Whitley, 1994). En supposant que la valeur de fitness d'une solution est un nombre décimal (par exemple 1,36), il a calculé le nombre de copies de cette solution dans la population de parents comme suit. La partie entière de la valeur de fitness indique combien de copies sont directement placées dans la population de parents (donc 1 copie, dans le cas de 1,36). Puis, il faut déterminer si d'autres copies de cette solution doivent être encore ajoutées. Dans un premier temps, l'auteur considère le reste de la valeur de fitness (0,36) comme une possibilité de placer une deuxième copie de cette solution. Puis, il génère une valeur aléatoire (entre 0 et 1) (par exemple 0,8) et fait une comparaison. Si cette valeur est plus grande que (1- le reste), c'est-à-dire (1-0,36= 0,64), alors il ajoute une autre copie de cette solution à la population de parents.
- la sélection par rang **(*Ranking selection*),** se compose de deux étapes (Whitley, 1989). D'abord tous les individus de la population sont rangés selon leurs valeurs de fitness. Le rang est fait dans l'ordre décroissant (ou croissant), selon si on veut minimiser (ou maximiser) la fonction de fitness. Normalement, les individus de moins bonne qualité obtiennent un rang faible (à partir de 1). Et ainsi en itérant sur chaque individu on finit par attribuer le rang N au meilleur individu (où N est la taille de la population). Ensuite on effectue une sélection par roulette basée sur les rangs des individus où l'angle de chaque secteur de la roue sera proportionnel au rang de l'individu.

3.2.4.2 Sélection par tournoi (*Tournament selection* ou TS)

La sélection par tournoi est l'une des sélections les plus utilisées dans les algorithmes évolutionnaires. Le principe consiste à choisir un sous-ensemble d'individus (S individus) aléatoirement dans la population, puis à sélectionner le meilleur individu

dans ce groupe en fonction de sa fitness. Ce processus est répété jusqu'à l'obtention du nombre d'individus requis (Miller et Goldberg, 1995). Le nombre de participants à un tournoi (S), appelé la taille de tournoi, est utilisé pour faire varier la pression de cette sélection. Si ce nombre est grand, alors la pression sera forte et les faibles individus auront une petite chance d'être choisis. En général, un seul gagnant est choisi parmi les participants à un tournoi. Ce gagnant peut être choisi d'une façon déterministe ou probabiliste. Dans le cas déterministe, qui est pratiquement le plus utilisé, le gagnant est l'individu de meilleure qualité (meilleure fitness). Dans le cas probabiliste, chacun des participants peut être choisi en tant que gagnant avec une probabilité proportionnelle à sa fitness. Cela consiste à utiliser une procédure de sélection proportionnelle (comme la « roulette » citée ci-dessus) se référant seulement aux individus du tournoi. La méthode de tournoi la plus commune est le tournoi binaire, où on choisit deux individus aléatoirement (S=2) puis on sélectionne le meilleur (qui a la valeur de fitness la plus élevée). Plus d'informations sur la sélection par tournoi peuvent être trouvées dans (Bäck *et al*., 2000) (Freitas, 2002).

Les individus sélectionnés, proportionnellement à leur fitness ou par tournoi, sont ensuite recombinés pour générer de nouveaux individus (enfants), selon divers processus décrits dans la section 3.2.5.

3.2.5 Reproduction (variation)

Pour que l'algorithme puisse trouver des solutions meilleures que celles représentées dans la population courante, il est indispensable qu'elles soient transformées par l'application d'opérateurs de variation sur des copies des individus sélectionnés pour en engendrer de nouveaux. En général, on peut trouver deux classes principales :

- les opérateurs de recombinaison (croisement) qui produisent un ou plusieurs enfants à partir de deux (ou plusieurs) parents,
- et ceux de mutation qui produisent un nouvel individu à partir d'un seul individu.

Parfois, d'autres procédures d'amélioration de la qualité de la solution ou de diversification (amélioration de la diversité de la population) sont également utilisées. Mais elles reposent sur des procédures beaucoup plus complexes, qui, pour la plupart, ont été présentées dans le chapitre 1. Il peut s'agir de recherches locales ou d'algorithmes de colonies de fourmi, comme dans (Berger *et al.*, 2001) ou de règles basées sur des indicateurs de similarité entre individus, comme la règle « *cost spacing rule* » utilisée par (Labadi *et al.*, 2008), voir section 2.4.4.3.

Par conséquent, ils ont été exclus de cette section, par ailleurs déjà très volumineuse. Les paragraphes suivants présentent essentiellement des opérateurs de croisement et de mutation en se concentrant plus particulièrement sur les opérateurs utilisés pour le codage de permutation.

3.2.5.1 Croisement (*Crossover* ou Recombinaison)

D'une manière générale, le croisement consiste à appliquer des procédures avec une certaine probabilité (qui s'appelle le taux de croisement [$Pc \in (0, 1)$] sur les individus sélectionnés pour donner naissance à un ou plusieurs (généralement deux) enfants. Ces derniers doivent hériter, par croisement, de certaines caractéristiques des parents. Le taux de croisement représente la proportion de la population de parents qui sera utilisée

par un opérateur de croisement. Il est lié à d'autres paramètres comme la taille de population, le taux de mutation, et la procédure de sélection. Généralement, les taux les plus utilisés sont dans l'intervalle [0.45, 0.95].

Au début de la recherche, quand les solutions sont plutôt différentes, le croisement a un rôle d'exploration, car la combinaison des solutions produit des solutions placées aux endroits non encore visités pendant la recherche. Par contre, à la fin de la recherche, les parents sont similaires, ce qui produit des enfants similaires, accordant au croisement un rôle plutôt d'intensification (ou d'exploitation).Pratiquement, en recombinant des individus, l'enfant peut être potentiellement infaisable.

Comme la présentation des codages direct et indirect l'a déjà mentionné, il existe alors trois manières différentes d'échapper à ce problème. Le plus simple est de rejeter les enfants infaisables. Ils peuvent également être pénalisés ou un objectif supplémentaire peut être ajouté pour minimiser leur « degré d'infaisabilité». Enfin, la troisième possibilité consiste à tenter de réparer ces solutions infaisables, voir (Eiben et Ruttkay, 1997). La suite passe en revue vingt-deux types de croisement différents, identifiés par leurs noms en anglais, qui sont souvent utilisés pour la représentation de permutation (notamment pour le VRP).

Order Crossover (OX)

L'opérateur de croisement OX choisit deux points aléatoires de croisement. L'enfant hérite des éléments situés entre les deux points de croisement, inclus, du premier parent. Ces éléments occupent les mêmes positions, et donc apparaissent dans le même ordre que dans ce dernier. Les éléments restants sont hérités du deuxième parent dans l'ordre dans lequel ils apparaissent dans ce parent, en commençant par la première position suivant le deuxième point de croisement et en omettant tous les éléments déjà présents dans l'enfant.

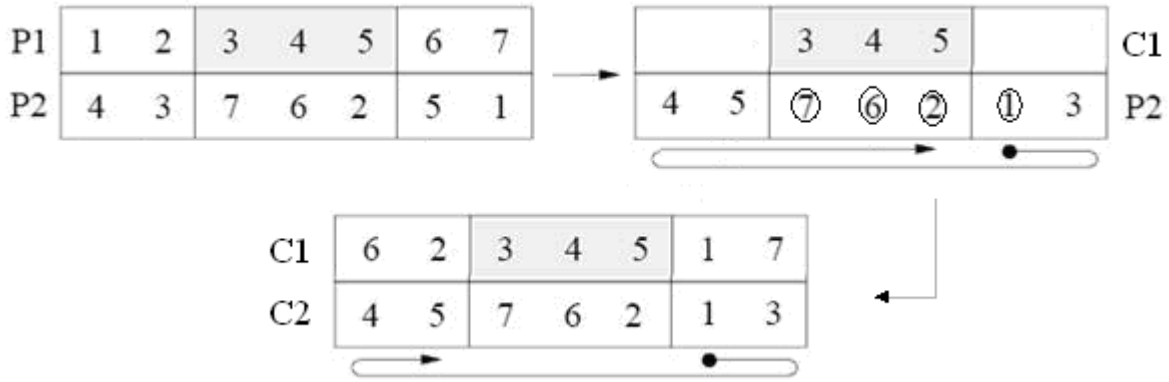


Fig. 3.5 – Illustration du croisement OX

L'exemple de la figure (3.5), qui illustre ce type de croisement, est issu de (Rodriguez-Tello *et al.*, 2005). Ces auteurs ont utilisé OX pour traiter un problème appelé *Minimum Linear Arrangement Problem* (ou MinLA), modélisé sous forme de permutations. Dans cet exemple, l'enfant C1 hérite du premier parent P1 les éléments [3, 4 et 5] qui sont situés entre les deux points de croisement. Ensuite, après le deuxième

point de croisement nous le complétons avec les éléments qui manquent [1, 2, 6, 7] mais selon leur ordre d'apparition dans le deuxième parent P2, à partir de la position de [1] qui est le premier élément après le deuxième point de croisement dans le deuxième parent P2. Donc l'ordre sera [1, 7, 6, 2].

Linear Order Crossover (LOX)

LOX présente une seule différence avec OX. Les éléments restants sont aussi hérités du deuxième parent dans l'ordre dans lequel ils apparaissent dans ce parent, mais en commençant par la première position de gauche à droite, et en omettant tous les éléments déjà présents dans l'enfant. (Sevaux et Dauzere-Peres, 2000), qui a traité un problème d'ordonnancement à une machine (single-machine scheduling problem), donne un exemple de LOX, figure (3.6).

Pour l'enfant 1, la séquence (3,4,5) du parent 1 est tout d'abord recopiée, puis les gènes restants (1,2,6 et 7) sont replacés dans leur ordre d'apparition dans le parent 2 en partant de la gauche et en omettant les gènes 3, 4et 5. Cela résulte en deux séquences : (7, 6) et (2,1) qui viennent compléter le chromosome avant et après la zone délimitée par les deux points de coupure. Le rôle des parents est inversé pour construire de la même manière le second enfant.

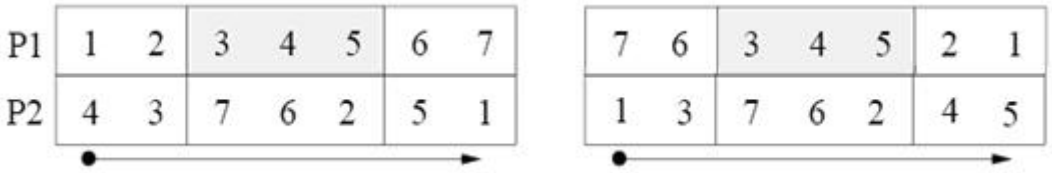


Fig. 3.6 – Illustration du croisement LOX (Sevaux et Dauzere-Peres, 2000)

One point Crossover (1X)

L'opérateur de croisement le plus simple est le croisement en un point. Initialement, cet opérateur a été proposé pour la représentation binaire. La première étape consiste à choisir aléatoirement un point de coupure pour partager chaque parent en deux parties. Puis le premier enfant est construit en utilisant la première partie du premier parent et la deuxième partie du deuxième parent. A l'inverse, le deuxième enfant est une concaténation de la seconde partie du premier parent et de la première partie du second parent. Mais cet opérateur s'il est appliqué aux problèmes représentés sous forme de permutations, comme le TSP ou le VRP, a de fortes chances de produire des enfants invalides, par duplication et/ou omission de certains éléments de la permutation. Pour résoudre ce problème, cet opérateur est adapté pour le codage de permutation. La figure (3.7) donne un exemple pour illustrer cette modification.

Pour construire le premier enfant (C1), la première partie (8 2 3 6) du premier parent (P1) est d'abord recopiée, puis ensuite les éléments (5 4 7 1 9) de la deuxième partie de ce parent sont ré-ordonnés dans l'ordre d'apparition qu'ils ont dans le deuxième parent (P2). Le second enfant est généré de manière symétrique, en échangeant le rôle de P1 et P2. Notez que le croisement 1X est une simplification de OX ou LOX si un seul point de croisement est utilisé au lieu de deux points (Portmann, 1996).

P1 = 8 2 3 6 | 5 4 7 1 9 → C1 = 8 2 3 6 | 4 5 1 7 9
P2 = 4 5 2 1| 8 7 6 9 3 → C2 = 4 5 2 1 | 8 3 6 7 9

Fig. 3.7 – Illustration du croisement 1X

N Point Crossover (nX)

Cet opérateur est une généralisation du croisement 1X avec n points de croisement. Ces points choisis aléatoirement coupent chaque parent en n+1 parties. Dans le processus qui construit le premier enfant [illustré dans la figure (3.8), dans le cas où n vaut 2], les éléments (8, 2 et 7, 1 et 9) des parties impaires du premier parent (P1) sont héritées du parent à l'enfant (C1), et les éléments des parties paires (3, 6, 5 et 4) sont réordonnés dans le même ordre que dans l'autre parent. Pour le second enfant, les rôles des parents sont inversés.

P1 = 8 2 | 3 6 5 4 | 7 1 9 → C1 = 8 2 | 4 5 6 3 | 7 1 9
P2 = 4 5 | 2 1 8 7 | 6 9 3 → C2 = 4 5 | 8 2 7 1 | 6 9 3

Fig. 3.8 – Illustration du croisement 2X

Partially Mapped Crossover (PMX)

Dans PMX, qui a été proposé par (Goldberg et Lingle, 1985), deux parents sont considérés et deux positions de découpage sont sélectionnées uniformément au hasard. Les deux points de croisement (P1 et P2) définissent une section compatible qui est utilisée pour effectuer des opérations d'échange de gènes, position par position. Pour l'illustration, considérons l'exemple de (Baudet *et al.*, 1996) pour le problème d'ordonnancement dans les ateliers de type Job Shop (*Job Shop Scheduling Problem* ou JSSP) et supposons les deux parents suivants, figure (3.9) :

			P1			**P2**				
Parent 1	9	8	4	5	6	7	1	3	2	10
			P1			**P2**				
Parent 2	8	7	1	2	3	10	9	5	4	6

Fig. 3.9 – Illustration du croisement PMX (Baudet *et al.*, 1996), étape 1

PMX procède par des échanges de position. Au départ, les enfants 1 et 2 sont des copies conformes des parents 1 et 2, respectivement. Puis les gènes se trouvant entre les points de coupure sont échangés deux à deux, position par position (5 avec 2, 3 avec 6, et 10 avec 7).

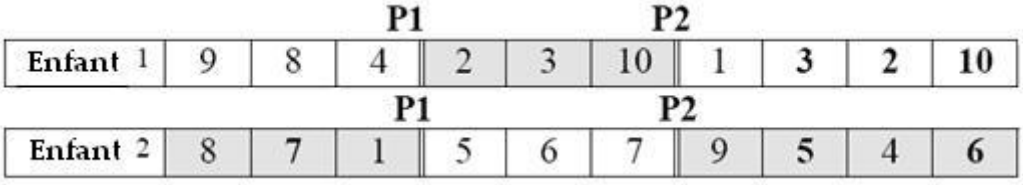


Fig. 3.10 – Illustration du croisement PMX (Baudet *et al.*, 1996), étape 2

Ensuite, pour les gènes placés avant et après les points de coupure, il reste à traiter le cas des gènes répétés (en double). Dans l'enfant 1, c'est le cas pour les gènes qui étaient placés dans la zone délimitée par les points de coupure dans le parent 2, mais en dehors de cette zone dans le parent 1. Dans ce cas, le gène placé en dehors de la zone prend la valeur du gène du parent 1 placé à la même position à l'intérieur de la zone, et par conséquent à la même position dans l'enfant 2. Par exemple, dans la figure 3.10, le gène 3 apparaît à deux endroits dans l'enfant 1 : une première fois en cinquième position (entre les points de coupure) et une seconde fois en huitième position, après le second point de coupure. La valeur 6, placée en cinquième position dans le parent 1 et le second enfant (entre les points de coupure P1 et P2) est donc utilisée pour remplacer le gène 3 en huitième position dans l'enfant 1. De même, les gènes 5 et 7 placés en quatrième et sixième position dans l'enfant 2 sont utilisés pour remplacer les gènes 2 et 10 aux neuvième et dixième positions de l'enfant 1. La même procédure est répétée jusqu'à éliminer tous les gènes apparaissant deux fois dans l'enfant 2. Les enfants finalement obtenus sont donc :

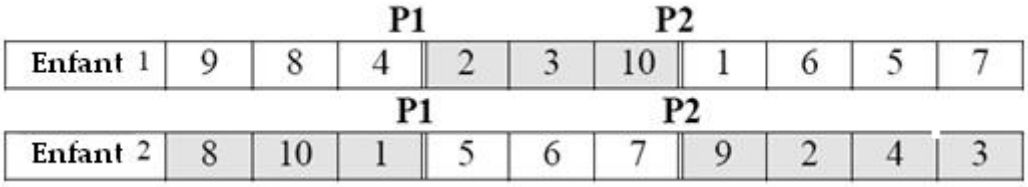


Fig. 3.11 – Illustration du croisement PMX (Baudet *et al.*, 1996), étape 3

Uniform Crossover (UX)

Le croisement uniforme de permutations tend à faire hériter un enfant d'une combinaison des ordres existants dans deux parents. Le croisement se déroule en trois étapes :

- un masque binaire est engendré aléatoirement, figure (3.12, a);
- deux parents sont appariés. Les "0" du masque binaire définissent les positions préservées dans la séquence du parent 1, et les "1" du masque binaire définissent les positions préservées dans la séquence du parent 2, figure (3.12, b);
- pour obtenir l'enfant 1, les éléments non préservés du parent 1 sont permutés de façon à respecter l'ordre qu'ils ont dans le parent 2, figure (3.12, c).

Un autre exemple peut être trouvé dans (Siarry *et al*,. 2003).

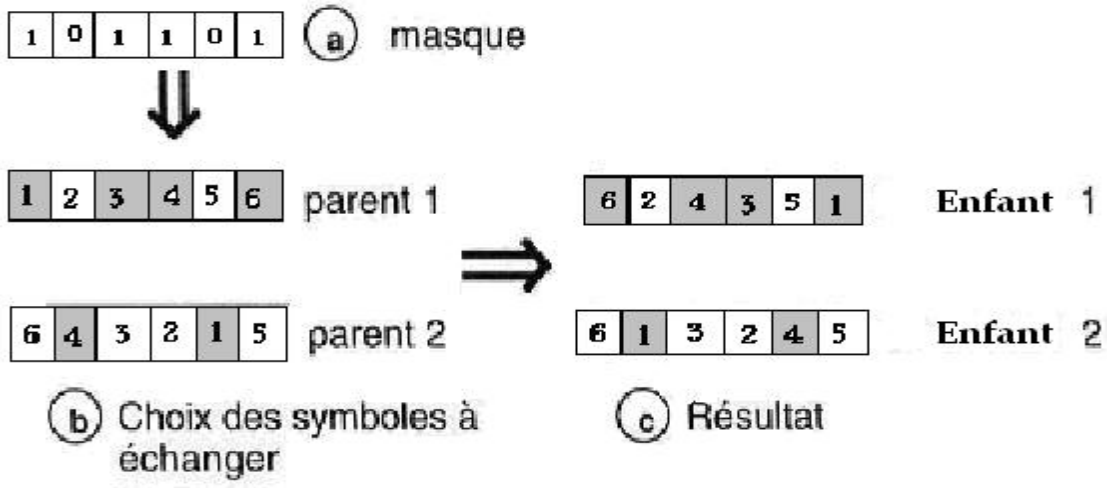


Fig. 3.12 – Illustration du croisement UX

Position-based Crossover (POS)

(Syswerda, 1989) a proposé ce croisement qui est essentiellement un croisement uniforme. La figure (3.13) est un exemple de ce croisement, issu de (Barbulescu *et al*., 2004). D'abord, on choisit au hasard un ensemble de positions (*) à partir d'un parent (P1 par exemple) et on les copie dans les mêmes positions dans l'enfant correspondant (C1). Ensuite, on supprime les gènes qui sont déjà choisis dans l'autre parent (P2). Ainsi, le dernier parent (P2) contient seulement les gènes dont l'enfant (C1) a besoin. Enfin, on met ces gènes dans les positions à compléter dans l'enfant C1, de gauche à droite, dans l'ordre dans lequel ils apparaissent dans ce parent (P2).

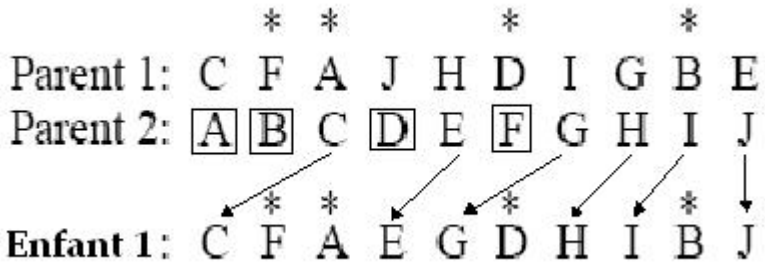


Fig. 3.13 – Illustration du croisement POS

Order-Based Crossover (OBX)

Ce croisement a également été proposé par (Syswerda, 1989), il est illustré par la figure (3.14). C'est une version légèrement différente du croisement (*Position-based Crossover* ou POS) dans laquelle l'ordre des symboles sélectionnés (*) dans un parent (P1), sont placés dans le fils aux positions exactes qu'ils occupent dans l'autre parent (P2).

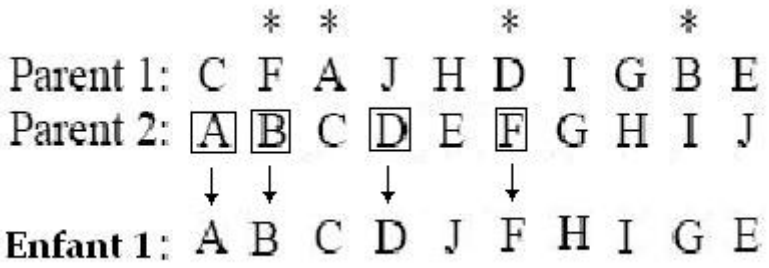


Fig. 3.14 – Illustration du croisement OBX

Route Crossover (RC)

RC est une amélioration du croisement UX (*uniform crossover*) (Davis, 1991). Comme dans le croisement UX standard, un masque binaire est produit aléatoirement. Cependant, dans UX, la longueur d'un masque est égale à la longueur des chromosomes des parents. Autrement dit, dans UX chaque bit du masque correspond à chaque gène (c'est-à-dire à chaque client) des parents. Par contre, dans le croisement RC, un bit du masque correspond à une tournée dans le chromosome. Ainsi, dans RC, la longueur du masque est égale au nombre de tournées représentées dans un parent. Le nombre de tournées n'est pas nécessairement identique dans les deux parents, alors deux masques sont produits (un pour chaque parent).

L'exemple de la figure (3.15), donné par (Ombuki *et al.*, 2002), illustre comment RC produit deux enfants. Cet exemple considère deux parents (P1 et P2) contenant chacun 8 clients. D'abord, un masque binaire (composé de 0 et de 1) est construit aléatoirement pour chaque parent, et sa longueur égale le nombre de tournées du parent auquel il correspond. Dans l'exemple, un masque binaire de longueur trois est construit pour le parent P1, puisqu'il contient trois tournées [1 2 3], [4 5] et [6 7 8] et de la même manière, un masque de longueur quatre est construit pour le parent P2. Puis, le contenu des tournées correspondant à « 1 » dans les masques est copié directement dans les enfants correspondants. Par contre, les clients (gènes) des tournées associées à « 0 » dans le masque sont enlevés pour former une liste respectant leur ordre d'apparition dans les parents. Par exemple, la liste (L1) qui contient [1 2 3] est générée en respectant l'ordre dans P1, et la liste (L2) qui contient [8 6 7] respecte l'ordre dans P2. Ensuite, ces listes sont ré-ordonnées selon l'ordre de l'autre parent. (L1) devient [2 1 3] et (L2) devient [6 7 8]. Enfin, on utilise ces listes pour réinsérer les gènes dans les chromosomes. Les clients de la liste (L1) sont successivement considérés dans l'ordre de cette liste. Chacun d'eux est inséré à la première position faisable dans les tournées de l'enfant (C1). Si une insertion d'un client ne satisfait pas les contraintes du problème, alors ce client est considéré comme non servi. Il n'est pas affecté à une tournée mais à la fin de la procédure, il est ajouté à l'enfant dans une tournée supplémentaire (comme le client (7) dans C2, par exemple).

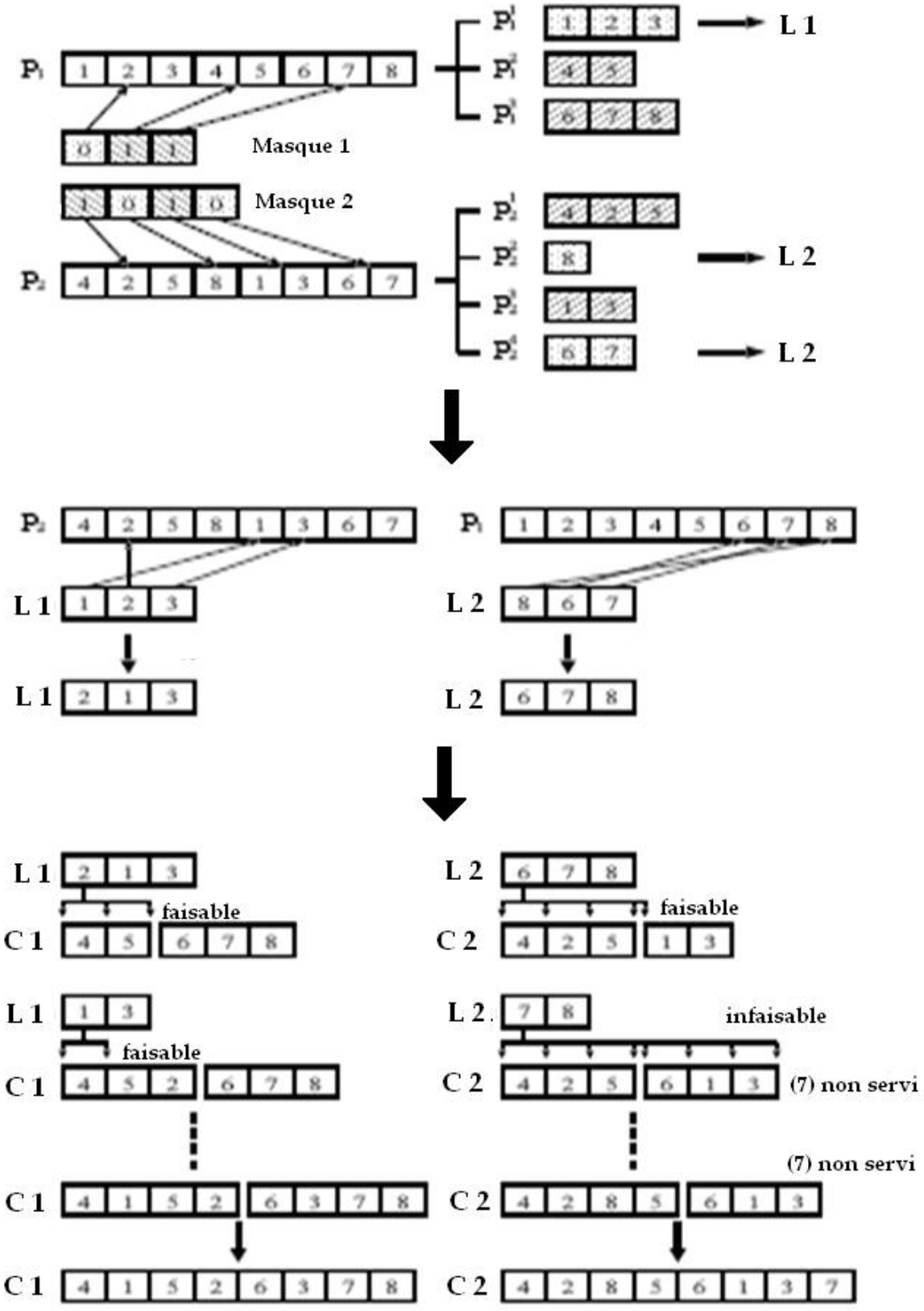


Fig. 3.15 – Illustration du croisement RC (Ombuki *et al.*, 2002)

Cycle Crossover (CX)

Le croisement CX est un opérateur proposé par (Oliver *et al.*, 1987) . Il préserve l'information contenue dans les deux parents. Pour expliquer ce type de croisement, considérons l'exemple représenté à la figure (3.16) qui est pris de (Merz, 2000) qui a traité un problème d'affectation quadratique (*Quadratic Assignment Problem* ou QAP).

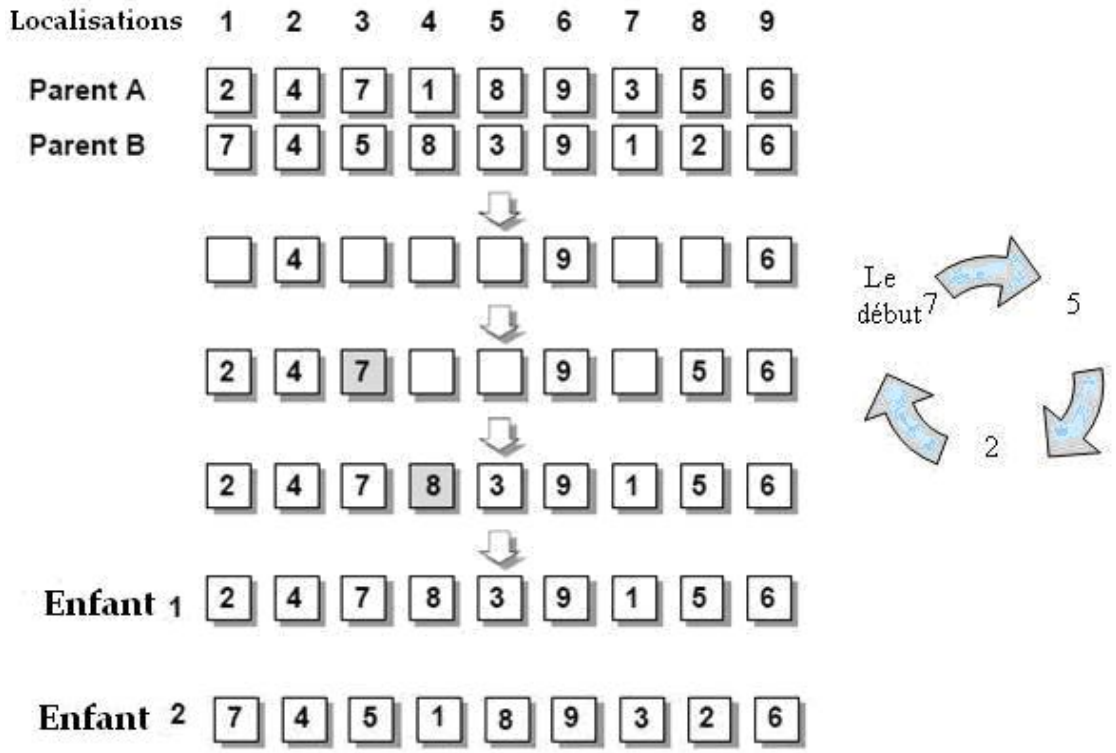


Fig. 3.16 – Illustration du croisement CX par (Merz, 2000)

D'abord, tous les gènes qui sont assignés à la même place dans les deux parents sont hérités par les enfants. Ce sont les gènes 4, 9, et 6. Puis, commençant par une position choisie aléatoirement (dans cet exemple, localisation 3), un gène est choisi aléatoirement parmi un des parents et est assigné à la même place dans l'enfant. Dans l'exemple, pour construire l'enfant 1, c'est le gène 7 du parent A qui est copié dans l'enfant 1. Puis l'algorithme cherche quel gène est en face de 7 (ou dans la même position) dans le parent B : c'est le gène 5. Ensuite, le gène 5 est placé dans l'enfant mais dans la même position que celle qu'il occupe dans le parent A (position 8). De la même manière, le gène 2 du parent A est placé dans l'enfant 1 parce que le gène 2 du parent B est en face du gène 5 de parent A. Ceci ferme le cycle car le gène suivant est le gène 7, qui est déjà inclus dans l'enfant 1. Finalement, les gènes encore à compléter dans l'enfant 1 sont trouvés en examinant les positions correspondantes du parent B. C'est-à-dire, les gènes 8, 3 et 1 sont placés dans l'enfant 1 aux positions 4,5 et 7 respectivement. Enfin, cette procédure est répétée en inversant le rôle des parents pour obtenir l'enfant 2.

Trajectory Crossover (TX)

Ce croisement produit les enfants en explorant la trajectoire qui relie les deux parents. La procédure commence par l'un des parents, appelé solution initiale. Ensuite, une position aléatoire est choisie pour les deux parents [par exemple la position 3 pour les parents A et B dans la figure (3.17, a)]. Les éléments situés à cette position sont permutés ce qui construit deux autres solutions. Dans l'exemple, la position 3 est occupée par le gène 7 dans le parent A (appelé solution initiale), et par le gène 5 dans le parent B. Permuter ces deux éléments (5 et 7) permet d'obtenir les solutions W et X. Les valeurs de *fitness* de ces deux solutions sont calculées. Supposons que la solution X est meilleure que W. Alors la solution W est éliminée, pour ne considérer que la solution initiale (parent A) et la solution gagnante (solution X). La procédure précédente est alors répétée sur l'élément suivant (position 4), figure (3.17, b). Les gènes 1 et 8 sont permutés pour obtenir les solutions Y et Z, dont les *fitness* sont calculées pour éliminer la pire solution, et ainsi de suite. Si les éléments situés à la position sélectionnée sont identiques dans les deux solutions, la position suivante est considérée. Cette procédure est répétée jusqu'à ce que toutes les positions aient été considérées pour obtenir l'enfant 1.

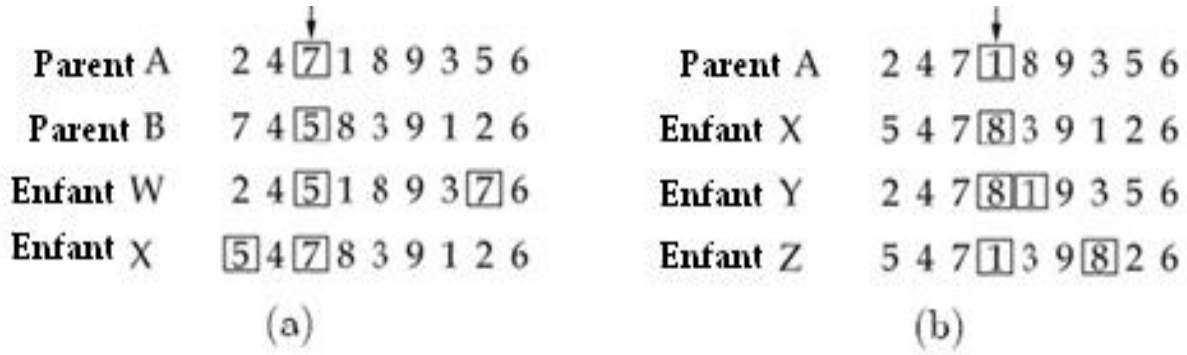


Fig. 3.17 – Illustration du croisement TX (Rodriguez-Tello *et al.*, 2005)

Distance Preserving Crossover (DPX)

DPX fonctionne comme dans la figure (3.18) qui est issue de (Freisleben et Merz, 1996) pour le TSP. Cet exemple considère deux parents contenant chacun 10 villes. Pour produire le premier enfant, le premier parent est découpé en plusieurs fragments. Le découpage repose sur un principe de voisinage entre les villes. Chaque série de villes existant dans les deux parents simultanément, est considérée comme un fragment.

Dans l'exemple, le parent 1 est découpé selon les fragments (5 3 9), (1 2), 8, (0 6), 7 et 4. Ensuite, une ville est choisie aléatoirement comme point de départ (la ville (6) dans l'exemple). La solution est complétée en ajoutant le reste du fragment contenant la ville (6), c'est-à-dire (0). Puis la solution est construite en ajoutant les villes progressivement. Les villes candidates pour occuper la position suivante dans le chromosome sont les bords des fragments qui n'ont pas encore été utilisés, en excluant les bords qui sont déjà voisins de la ville (0) dans les parents. Dans l'exemple, l'ensemble de villes candidates

est {5, 9, 1, 2,4}. Les villes (8 et 7) n'appartiennent pas à cet ensemble, parce qu'il n'est pas souhaitable de réinsérer (0 8) ou (0 7) qui existent déjà dans les parents.

Parmi ces solutions candidates, la ville choisie est celle qui est la plus proche de (0). Dans l'exemple, il s'agit de la ville (5), et elle est ajoutée avec le reste du fragment qui la contient (5 3 9) à l'enfant 1. De la même manière, la ville la plus proche de la dernière ville apparaissant dans l'enfant (ici la ville(9)) est sélectionnée parmi le nouvel ensemble de villes candidates {2, 8,7}. C'est la ville (8) qui est supposée être la plus proche, et elle est ajoutée. Le procédé est répété jusqu'à ce que tous les fragments aient été réunis.

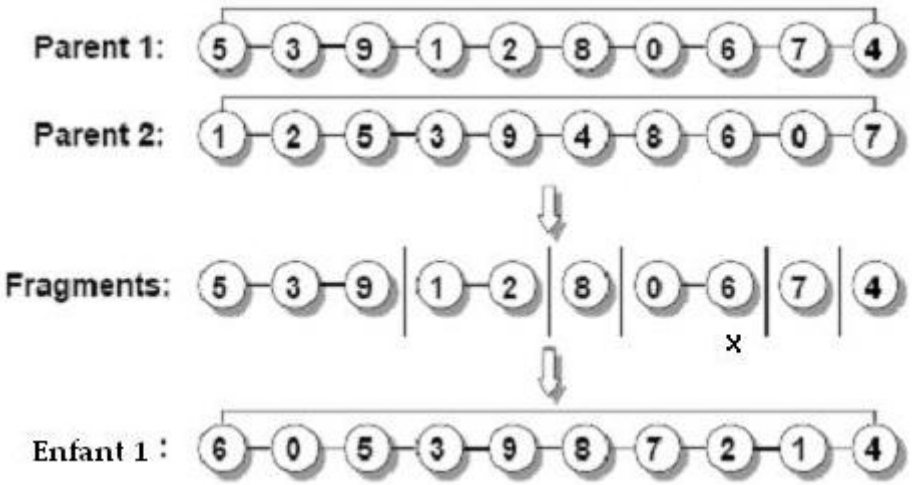


Fig. 3.18 – Illustration du croisement DPX

Brady's Crossover

Cet opérateur de croisement cherche à améliorer des sous-séquences présentes dans un parent en les remplaçant par des sous-séquences contenant les mêmes villes dans l'autre parent mais correspondant à une distance parcourue plus faible. L'algorithme recherche donc des sous-séquences contenant des villes identiques, compare leurs longueurs et effectue le remplacement s'il produit une amélioration. Pour illustrer ce croisement, la figure (3.19), extraite de (Merz, 2000) qui a traité le TSP, présente un exemple dans lequel les sous-séquences contenant les mêmes villes sont (8, 9, 5, 3,7) dans le parent A et (9, 8, 3, 7,5) dans le parent B. La distance d'une ville i à une ville j est notée dij. Si la somme (d89+d95+d53+d37) est plus grande que (d98+d83+d37+d75), la sous-séquence (8, 9, 5, 3,7) est remplacée par la sous-séquence (9, 8, 3, 7,5) dans le parent A. Le chemin dans le parent B reste inchangé. Brady signale dans ses recherches que pour un problème de 64 villes, il est conseillé de rechercher des sous-séquences de longueur variant entre 8 et 32 villes.

Parent A	4	2	0	|	8	9	5	3	7	|	6	1
Parent B	2	|	9	8	3	7	5	|	0	1	6	4
Enfant À	4	2	0	|	9	8	3	7	5	|	6	1

Fig. 3.19 – Illustration du croisement de Brady par (Merz, 2000)

Subsequence Exchange Crossover (SXX)

Le croisement SXX ressemble au croisement de Brady car il cherche aussi des sous-séquences communes aux deux parents. Ces sous-séquences contiennent les mêmes gènes mais dans un ordre différent dans les deux chromosomes des parents. Ensuite, le SXX produit un enfant en remplaçant des sous-séquences du parent P1 par des sous-séquences du parent P2. Il est également possible de créer un deuxième enfant en remplaçant des sous-séquences du parent P2 par des sous-séquences du parent P1. L'exemple de la figure (3.20) est donné par (Braune *et al.*, 2005) pour résoudre *le Job Shop Scheduling Problem* (JSSP). Il illustre le fonctionnement de SXX.

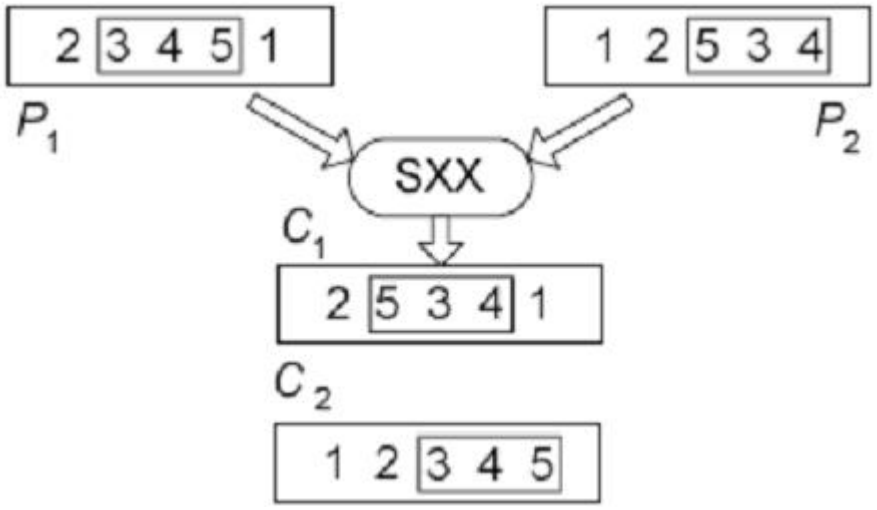


Fig. 3.20 – Illustration du croisement SXX fourni par (Braune *et al.*, 2005)

Sequence-Based Crossover (SBX)

Cet opérateur de croisement est illustré dans la figure (3.21) par (Potvin et Bengio, 1996) qui ont traité le VRP avec fenêtres de temps. Premièrement, un lien est choisi aléatoirement et enlevé dans chaque parent. Puis, les clients servis avant le point de coupure dans le parent 1 sont liés aux clients servis après le point de coupure dans le parent 2 (représentés par les nœuds noirs dans la figure (3.21)). Enfin, dans le parent 1, la nouvelle route remplace l'ancienne. Le deuxième enfant peut être créé en inversant le rôle des parents.

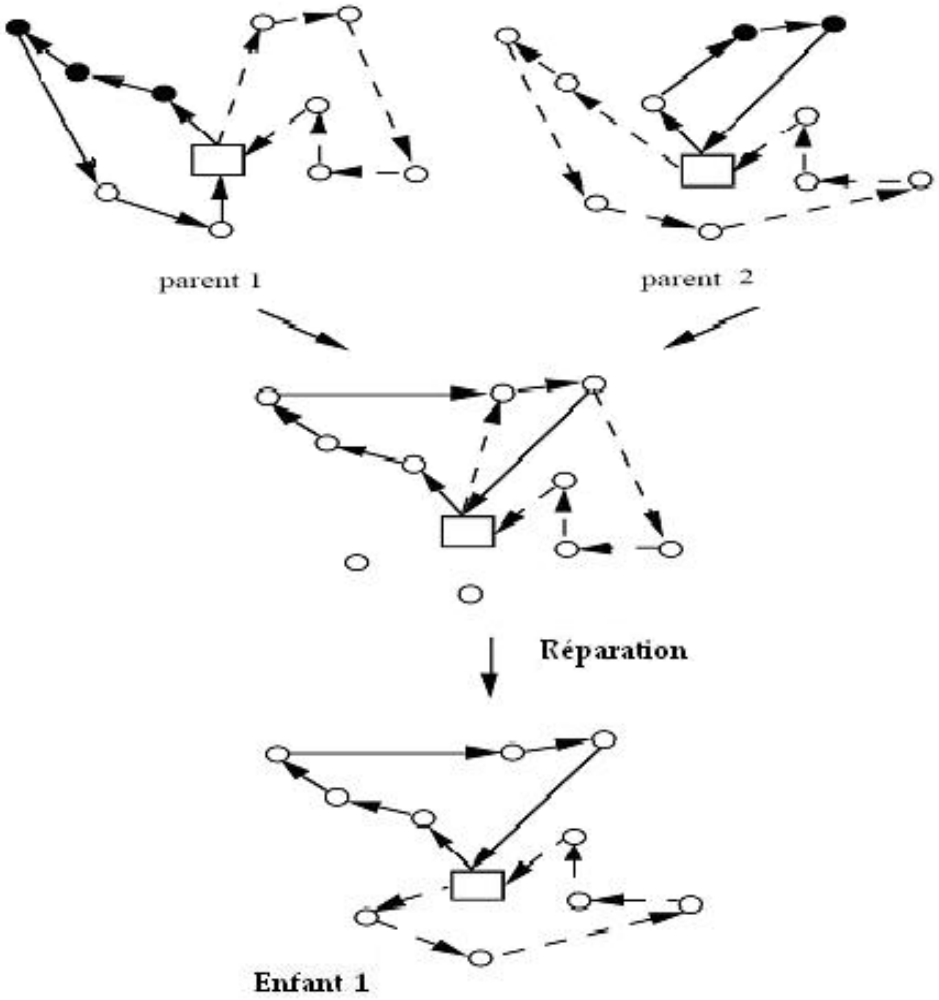


Fig. 3.21 – Illustration de SBX par (Potvin et Bengio, 1996)

Malheureusement, la nouvelle solution est rarement valide, car certains clients sont dupliqués ou au contraire non servis dans les nouvelles tournées construites. Par exemple, pour l'enfant de la figure (3.21), deux clients sont situés dans deux routes différentes, et par ailleurs deux autres clients sont non servis. En conséquence, il est nécessaire d'appliquer un opérateur de réparation pour produire des solutions faisables. Cet opérateur est appliqué sur un enfant infaisable de la façon suivante :

- Si un client apparaît deux fois dans une nouvelle route, une des deux copies est enlevée. Si un client apparaît une fois dans une nouvelle route et une fois dans une ancienne route, il est enlevé de l'ancienne.
- Si un client est non servi, alors ce client est inséré à une place faisable qui peut minimiser la distance parcourue. Dans le VRPTW, il n'y a aucune garantie qu'il y ait une place faisable d'insertion pour tous les clients non servis, donc si la procédure échoue, l'enfant est rejeté et deux autres parents sont sélectionnés pour recommencer le croisement.

Route-Based Crossover (RBX)

Cet opérateur crée un enfant 1 en remplaçant un itinéraire du parent 2 par un itinéraire du parent 1 (Potvin et Bengio, 1996). Comme dans SBX, RBX produit probablement des enfants avec clients non servis ou clients dupliqués. Par exemple, dans la figure (3.22), l'itinéraire avec les clients noirs dans le parent 1 remplace l'itinéraire correspondant dans le parent 2. On remarque qu'un client est situé maintenant dans deux itinéraires différents, alors qu'un autre client est devenu non servi. Par conséquent, l'opérateur de réparation présenté dans la section précédente (SBX) est appliqué sur les solutions infaisables pour les transformer si possible en solutions faisables.

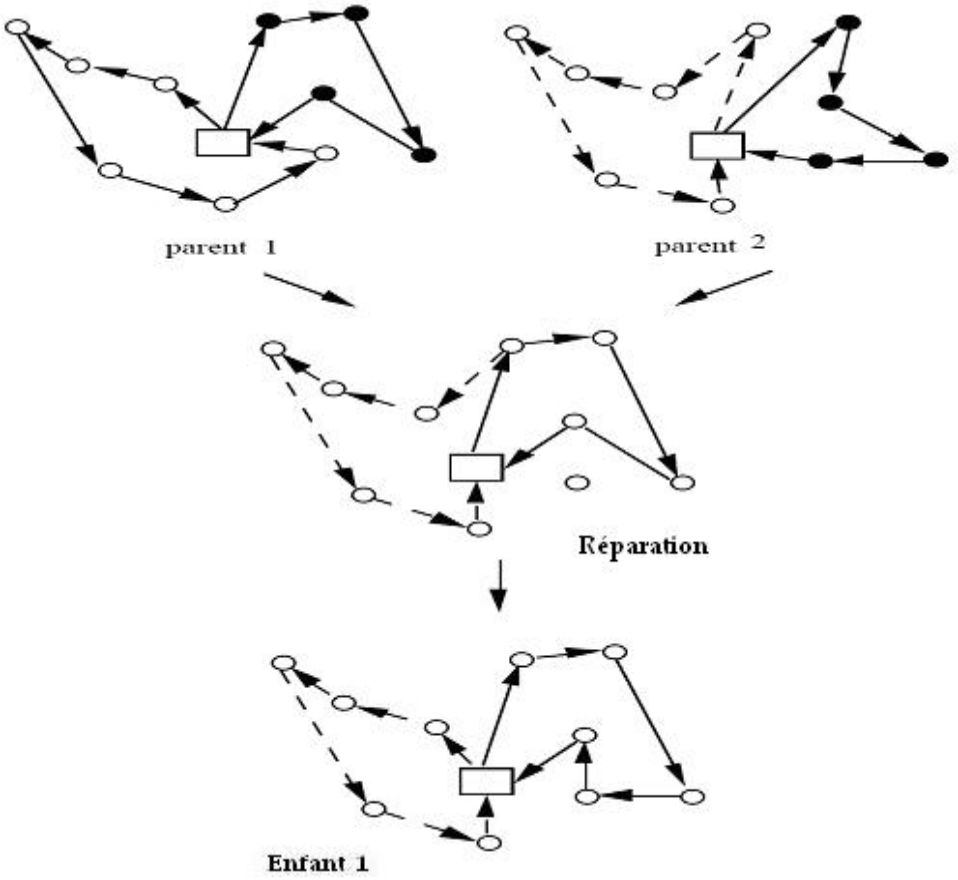


Fig. 3.22 – Illustration de RBX (Potvin et Bengio, 1996)

Merge Crossover

(Blanton et Wainwright, 1991) ont proposé le croisement *Merge Crossover* pour le problème de tournées de véhicules avec des fenêtres de temps. Dans ce problème, chaque client a une fenêtre de temps à respecter pour effectuer le service. Par conséquent, il existe une relation de précédence entre les clients, basée sur leurs fenêtres de temps. Généralement, le principe de ce croisement s'appuie sur cette relation pour comparer deux gènes issus des deux parents. Le gène dont la fenêtre de temps est plus tôt gagne le duel et est inséré dans l'enfant, alors que l'autre gène est permuté ou

supprimé dans l'autre parent. Cet opérateur de croisement est nommé *Merge Crossover* 1 (ou M1) et génère un seul enfant. Il est détaillé dans l'exemple suivant, figures (3.23) à (3.25), extrait de (Louis *et al.*, 1999). A et B sont deux parents et AB est leur enfant. Le vecteur de précédence est construit en rangeant les clients par ordre croissant de fenêtres de temps. Dans l'exemple fourni, il est supposé égal à [0 1 2 3 4 5 6 7 8 9].

Parent A: 2 5 6 1 0 7 3 8 4 9
Parent B: 4 1 6 9 3 8 2 0 5 7

Fig. 3.23 – Illustration du croisement *Merge Crossover* : les parents (Louis *et al.*, 1999)

Ainsi, par exemple, la limite inférieure de la fenêtre de temps du client 1 (à partir de laquelle le service chez ce client est possible) est plus petite que celle de la fenêtre de temps du client 2, et ainsi de suite. Les clients (gènes) qui sont en première position dans les parents, soulignés dans la figure3.23, sont tout d'abord comparés. Selon le vecteur de précédence, le gène du parent A (2) a une date possible de début de service plus petite que celle du gène du parent B (4). Donc, le client (2) est inséré dans l'enfant AB et il échange sa position avec le client (4) dans le parent B, figure (3.24).

Parent A: 2 5 6 1 0 7 3 8 4 9
Parent B: **2** 1 6 9 3 8 2 0 5 7

Enfant AB: **2** x x x x x x x x x

Fig. 3.24 – Illustration du croisement *Merge Crossover* : premier gène (Louis *et al.*, 1999)

Ensuite, la comparaison a lieu entre les gènes placés en deuxième position dans les parents (les gènes soulignés dans la figure (3.24)). En continuant de la même manière, l'enfant AB présenté dans la figure (3.25) est obtenu.

Enfant AB: 2 1 6 5 0 7 3 4 8 9

Fig. 3.25 – Illustration du croisement *Merge Crossover* : enfant (Louis *et al.*, 1999)

Le principe de croisement de l'opérateur *Merge Crossover* 2 (ou M2) est semblable, sauf que le client qui est ajouté à l'enfant, est supprimé dans les deux parents. Ainsi, ce croisement fonctionne comme suit pour les mêmes parents A et B, et le même vecteur de précédence. D'abord, le client 2 est ajouté à l'enfant et supprimé dans les deux parents, figure (3.26).

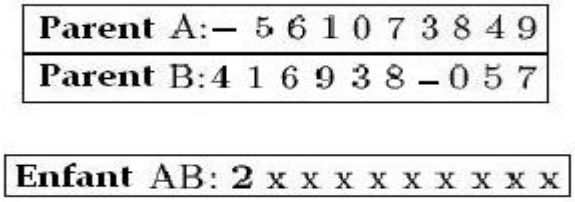


Fig. 3.26 – Illustration du croisement M2 : premier gène (Louis *et al.*, 1999)

Ensuite, alors que la comparaison se faisait dans M1 au niveau de la position suivante, elle est faite ici en considérant successivement les gènes qui n'ont pas encore gagné de duel dans les deux parents. Ainsi, l'exemple fourni compare les gènes (5) et (4), (4) est choisi et supprimé et ainsi de suite jusqu'à obtenir l'enfant suivant, figure (3.27).

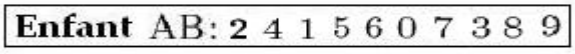


Fig. 3.27 – Illustration du croisement M2 : enfant (Louis *et al.,* 1999)

Heuristic crossover (HX)

Le croisement HX ressemble au croisement *Merge Crossover*, mais il utilise les distances entre les nœuds pour la comparaison entre les gènes des parents. Pour bien expliquer l'idée, la figure (3.28) utilise un exemple issu de (Vacic et Sobh, 2002). Tout d'abord, un gène est choisi aléatoirement dans les parents pour commencer la construction du chromosome de l'enfant [le gène B, en position 7 du parent 1, par exemple, souligné et recopié dans l'enfant dans la figure (3.28)].

Parent 1: H K C E F D **B** L A I G J

Parent 2: A B C D E F G H I J K L

Enfant : **B** _ _ _ _ _ _ _ _ _ _ _

Fig. 3.28 – Illustration du croisement HX, première étape (Vacic et Sobh, 2002)

Dans l'autre parent, le gène identique à celui qui vient d'être copié dans l'enfant est permuté avec celui qui occupe la même position que le gène copié (comme dans *Merge Crossover* 1). Ainsi, dans la figure (3.29), le gène B (en position 2) est permuté avec le gène G (placé en position 7) dans le parent 2.

Parent 1: H K C E F D **B** L A I G J

Parent 2: A G C D E F **B** H I J K L

Enfant : **B** _ _ _ _ _ _ _ _ _ _ _

Fig. 3.29 – Illustration du croisement HX, seconde étape (Vacic et Sobh, 2002)

Puis l'algorithme calcule la distance entre B et chacun des deux gènes qui lui succèdent dans les parents (L et H). Le gène qui est géographiquement plus près de B est sélectionnez. Dans l'exemple, les auteurs supposent qu'il s'agit de H. Alors, à nouveau, H est copié dans l'enfant et permuté avec L dans le parent 1, pour éviter la duplication, figure (3.30).

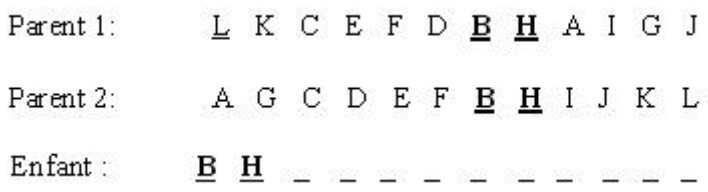


Fig. 3.30 – Illustration du croisement HX, second gène

Cette procédure est répétée jusqu'à ce qu'un nouveau chromosome (enfant) de la même longueur que les parents soit formé. Comme M1, cet opérateur génère juste un enfant. De plus, comme pour M2, un croisement heuristique semblable peut être créé en supprimant les gènes des parents au lieu de les permuter. Les deux opérateurs de croisement ainsi obtenus sont nommés *Heuristic Crossover* 1 (ou HX1) et *Heuristic Crossover* 2 (ou HX2).

Best Cost Route Crossover (BCRC)

La figure (3.31) illustre la création de deux enfants (C1 et C2) en utilisant deux parents (P1 et P2) contenant chacun 9 clients, par le croisement BCRC.

Cet exemple vient de (Ombuki *et al.*, 2006) qui a traité le VRP avec fenêtres de temps. RP1 et RP2 décrivent les tournées (ou routes) des parents P1 et P2, respectivement : P1 définit trois tournées [3 1 7], [5 6] et [4 2 8 9] et P2 également [1 9 8], [2 4 6 5] et [7 3]. La première étape du croisement choisit aléatoirement une route dans chaque parent (par exemple deuxième route de parent P1et troisième route de P2). Ensuite, les clients présents dans l'une de ces routes sont enlevés de l'autre parent. Dans l'exemple, les clients 5 et 6 sont enlevés de P2 et les clients 7 et 3 sont enlevés de P1 pour produire les enfants (C2 et C1), respectivement, comme indiqué dans la figure (3.31, a). Puisque chaque chromosome (enfant) devrait contenir tous les clients, la prochaine étape est de replacer les clients enlevés aux meilleures places possibles dans les enfants correspondants. Donc, comme le montre l'étape (b) de la figure (3.31), l'algorithme réinsère les clients 7 et 3 dans l'enfant C1 et les clients 5 et 6 dans l'enfant C2. Le choix du premier client à insérer est fait aléatoirement, par exemple, dans C1, l'ordre d'insertion des clients (7 et 3) est arbitraire. Le client 3 a été inséré dans la meilleure place trouvée dans C1 le premier, et ensuite le client 7 a été inséré, comme le montre l'étape (c) de la figure (3.31).

L'insertion des clients doit respecter les contraintes du problème. La meilleure place d'insertion est donc la place qui minimise le cout total tout en respectant les contraintes. Par exemple, BCRC vise à réduire le coût total et le nombre de véhicules simultanément, tout en vérifiant les contraintes de capacité et les fenêtres de temps pour le VRP avec fenêtres de temps. Si aucun point d'insertion faisable n'est trouvé pour un

client, une nouvelle route est définie pour ce client, comme par exemple le client 6 qui ne pourrait pas être inséré dans les routes de l'enfant 2.

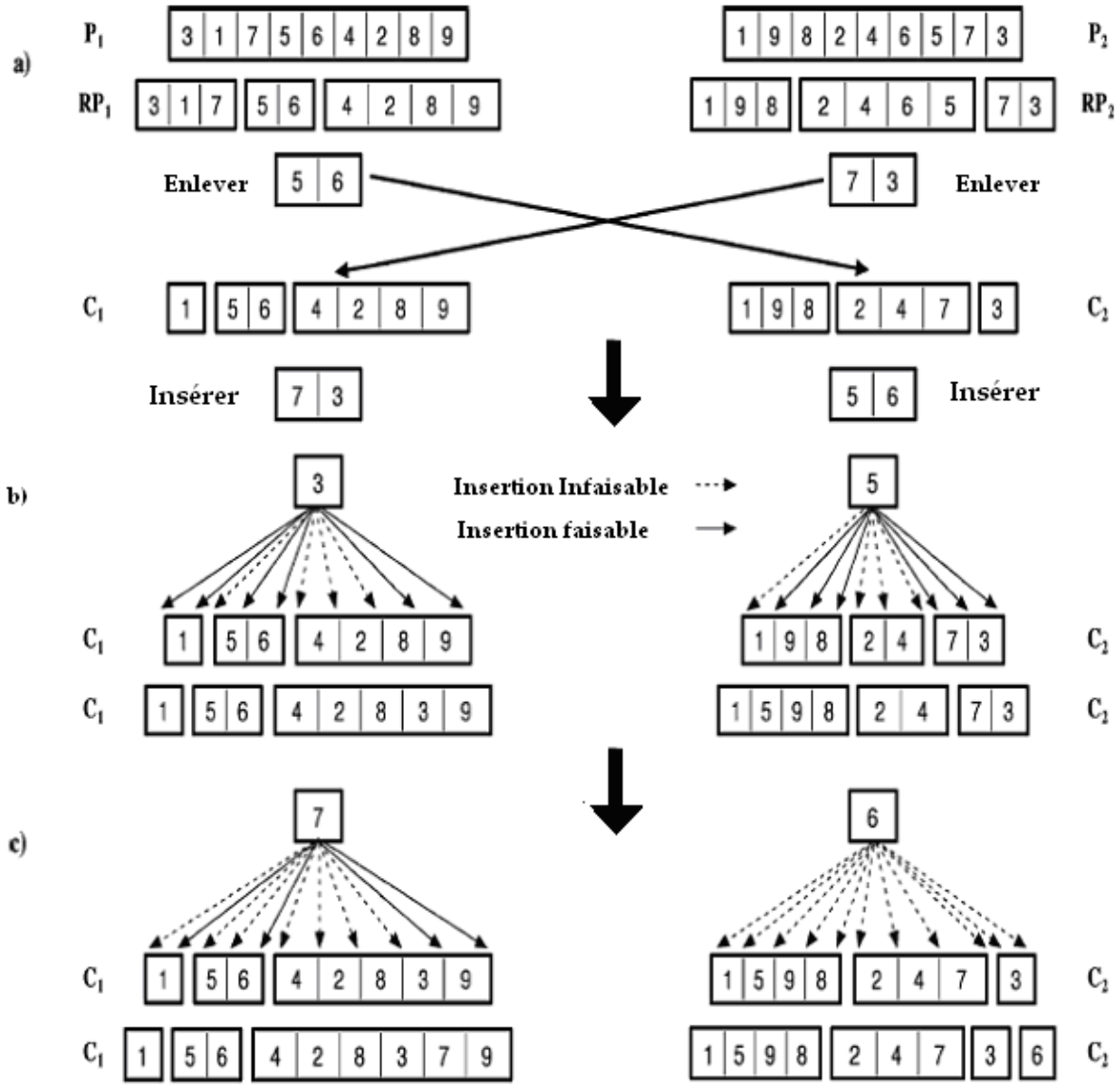


Fig. 3.31 – Illustration du croisement BCRC (Ombuki *et al*., 2006)

Edge Recombination Crossover (ERX)

Aussi nommé parfois *Genetic Edge Recombination* (ou GER) (Valenzuela, 1995), l'opérateur ERX utilise une liste d'adjacences pour construire des enfants qui héritent des informations (caractères) des parents. Cette liste stocke tous les liens (relations) entre nœuds contenus dans les deux parents, qu'il s'agisse d'arcs entrants ou sortants dans une ville. Donc, chaque ville aura deux relations d'adjacence au minimum et quatre au maximum (deux par parent). L'exemple suivant est donné par (Whitley *et al.*, 1991)

pour expliquer la construction de ces listes, dans le cadre de la résolution du TSP. Les deux parents considérés sont : [A B C D E F] et [B D C A E F]. Les liens extraits des tournées sont (ab bc cd de ef af) pour le premier parent et (bd dc ca ae ef bf) pour le second. Les listes obtenues sont donc :

- B F C E pour A,
- A C D F pour B,
- B D A pour C,
- C E B pour D,
- D F A pour E,
- A E B pour F.

ERX procède alors comme suit :

1. choisir la première ville de l'enfant parmi les deux premières villes des parents. Elle peut être choisie aléatoirement ou selon des critères décrits dans l'étape 4, et est appelée ville courante.
2. Enlever la ville courante des listes.
3. Si la ville courante a des liens (avec des villes qui n'ont pas encore été supprimées) dans sa liste, alors aller à l'étape 4 ; sinon, passer à l'étape 5.
4. Dans la liste de la ville courante, déterminer quelle ville a le moins de liens (villes) dans sa propre liste. La sélectionner pour devenir la ville courante et passer à l'étape 2. Dans le cas où deux (voire plusieurs) villes ont les mêmes nombres minimum de liens (villes) dans leurs listes, choisir la ville courante aléatoirement entre ces deux candidates.
5. S'il n'y a plus aucune ville non visitée dans toutes les listes, alors arrêter. Sinon, choisir une ville non visitée aléatoirement et passer à l'étape 2.

Par application de ces étapes dans l'exemple proposé, la première ville de l'enfant peut être choisie aléatoirement parmi les villes A et B. B est supposée choisie pour être la ville courante, et est supprimée de toutes les listes de villes. La liste de B indique que les candidats pour la prochaine ville sont A, C, D et F. Les villes C, D et F ont deux éléments dans leurs listes respectives. A n'est pas considéré car elle a trois éléments dans sa liste. C est aléatoirement choisi et supprimée de toutes les listes. Cette procédure est réitérée jusqu'à l'obtention de l'enfant final [B C D E A F] qui se compose complètement de liens issus des deux parents.

Recombinaison d'adjacences

Ce croisement, semblable à l'opérateur ERX, est repris de (Mathias et Whitley, 1992) qui lui donnent le nom de *Edge-2 recombination*. Ici, les adjacences communes aux deux parents sont marquées par (*) dans le tableau des adjacences. Les étapes sont similaires à celles de ERX sauf la quatrième étape qui détermine la ville suivante. (Mathias et Whitley, 1992) ont donné un exemple pour expliquer ce croisement, dans lequel deux individus sont considérés: (g, d, m, h, b, j, f, i, a, k, e, c) et (c, e, k, a, g, b, h, i, j, f, m, d). La première étape construit un tableau des adjacences (tableau 2.1) tel qu'à chaque nœud correspond une liste de sommets adjacents dans les deux parents. Les adjacences communes aux deux parents sont marquées d'une (*) dans le tableau (3.1).

Lors de la seconde étape, un nœud actif initial est choisi au hasard et toutes les références à ce nœud sont supprimées du tableau (dans l'exemple, le nœud (a) est choisi par hasard). L'étape 3 consiste à choisir l'arête qui, à partir du nœud actif, conduit à un

nœud adjacent marqué par une (*). A défaut, et s'il y a plusieurs villes (nœuds), la ville qui a le plus petit nombre de liens (adjacences) est choisie comme prochain nœud. En cas d'équivalence, le choix devient aléatoire. De plus, si la liste d'adjacences est vide pour une ville courante, la prochaine ville courante est choisie aléatoirement. Le nœud adjacent choisi devient le nouveau nœud actif ajouté dans l'enfant. Toutes les références à ce nœud sont supprimées des listes d'adjacences du tableau.

nœuds	listes des adjacences	noeuds	listes des adjacences
a	*k, g, i	g	a, b, c, d
b	*h, g, j	h	*b, i, m
c	*e, d, g	i	h, j, a, f
d	*m, g, c	j	*f, i, b
e	*k,*c	k	*e,*a
f	*j, m, i	m	*d, f, h

Tab. 3.1 –Liste des adjacences de l'exemple de (Mathias et Whitley, 1992)

Le tableau (3.2) montre le déroulement de l'algorithme. La progression dans la construction du cycle hamiltonien est présentée dans la dernière ligne. Les nœuds actifs sont soulignés. Lorsqu'un nœud actif est marqué (1), cela signifie qu'il a dû être choisi aléatoirement en raison de l'existence de plusieurs possibilités équivalentes. Lorsqu'il est marqué (2), il s'agit du cas d'une ville avec liste d'adjacences vide, et la prochaine ville est choisie aléatoirement selon l'étape (4) de ce croisement. Finalement, l'enfant obtenu est (a, k, e, c, d, m, h, b, g, i, f, j).

étape	1	2	3	4	5	6	7	8	9	10	11	12
a	*k,g,i	g, i	g, i	g, i	g, i	g, i	g, i	g, i	i			
b	*h,g,j	*h,g,j	*h,g,j	*h,g,j	*h,g,j	*h,g,	g,j	g, j	j	j	j	
c	*e,d,g	*e,d,g	d,g	d,g	g	g	g	g				
d	*m,g,c	*m,g,c	*m,g,c	*m,g	*m,g	g	g	g				
e	*k,*c	*c	*c									
f	*j,m,i	*j,m,i	*j,m,i	*j,m,i	*j,m,i	*j,i	*j,i	*j, i	*j, i	*j	*j	
g	b,c,d	b,c,d	b,c,d	b,d	b	b	b					
h	*b,i,m	*b,i,m	*b,i,	*b,i,m	*b,i,m	*b,i	*b, i	i	i			
i	h,j,f	h,j,f	h,j,f	h,j,f	h,j,f	h,j,f	j,f	j, f	j, f	j, f	j	
j	*f,i,b	*f,i,b	*f,i,b	*f,i,b	*f,i,b	*f,i,b	*f,i,b	*f,i	*f, i	*f		
k	*e	*e										
m	*d,f,h	*d,f,h	*d,f,h	*d,f,h	f,h	f,h	f	f	f	f		
tour	a(1)	k	e	c	d(1)	m	h(1)	b	g(2)	i	f(1)	j

Tab. 3.2 –Déroulement de *Edge-2 recombination* (Mathias et Whitley, 1992)

Maximal Preservative Crossover (MPX)

(Mülhenbein, 1993) et (Mühlenbein et Schlierkamp-Voosen, 1993) ont proposé le croisement MPX pour le TSP. L'idée de base de cet opérateur est d'insérer un segment d'un parent dans le chromosome de l'autre parent pour obtenir un enfant très proche de ses parents. L'exemple suivant, figure (3.32), illustre le fonctionnement de ce croisement, en considérant deux parents : P1 (a b c d e f g h i j k l) et P2 (c b a g h i j l k f d e). Tout d'abord, deux positions de coupure sont aléatoirement choisies dans les parents, comme dans X2. Ensuite, les clients entre les points de coupure sont copiés dans les enfants respectifs.

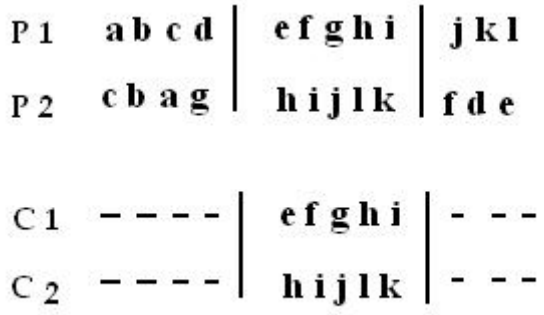


Fig. 3.32 – Première étape du croisement MPX

Puis, les gènes restants (hors de la zone délimitée par les points de coupure) sont traités de la façon suivante :

a) le $i^{ème}$ gène du parent 2 est copié en $i^{ème}$ position dans l'enfant 1, si cette copie respecte les contraintes (par exemple ne duplique pas un client dans l'enfant 1). Sinon, le $i^{ème}$ gène du parent1 est copié en $i^{ème}$ position de l'enfant1.

b) Si les deux cas précédents ne peuvent pas s'appliquer, le $i^{ème}$ gène de l'enfant 1 reçoit un gène de la zone située entre les points de coupure dans le parent 2, qui respecte les contraintes et ne fait pas une duplication (comme les gènes (e) et (g) dans l'enfant 2 dans la figure (3.33)).

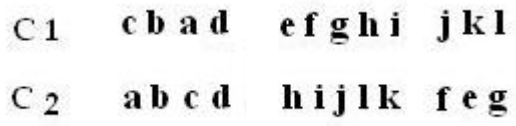


Fig. 3.33 – Exemples d'enfants obtenus par application de PMX

Ainsi, cette section a mis en évidence la grande variété d'opérateurs qui sont utilisés pour la première phase de la reproduction (croisement) destinée à combiner les caractéristiques de solutions existantes (parents) pour engendrer de nouvelles solutions (enfants) qui héritent de leurs caractéristiques. La section suivante s'intéresse à la seconde phase de la reproduction, la mutation, destinée à maintenir une diversité suffisante dans les générations successives.

3.2.5.2 Mutation

Après le croisement, un opérateur de mutation est appliqué au sein de la population des enfants avec une probabilité très faible, nommée taux de mutation ou Pm. Cet opérateur joue le rôle d'un "élément perturbateur". Il introduit du "bruit". Il permet ainsi de maintenir la diversité de la population des enfants et d'explorer l'espace de recherche en évitant à l'algorithme de converger trop rapidement vers un optimum local. Les petits taux de mutation sont recommandés car un grand taux peut occasionner une destruction de l'information utile contenue dans les solutions et être considéré probablement comme une recherche aléatoire.

Quelques taux de mutation communément employés en algorithmique évolutionnaire peuvent être trouvés dans (Bäck, 1992), comme Pm = 0.001 (DeJong, 1975), Pm = 0.01 (Grefenstette, 1986) et Pm$\in$ [0.005, 0.01] (Schaffer *et al.*,1989) ou ils peuvent aussi prendre une valeur égale à (1 / lg) où lg est la longueur de la chaîne de bits codant le chromosome.

Généralement, L'opérateur de mutation modifie de manière complètement aléatoire les caractéristiques d'un ou plusieurs individus de la population des enfants. Il modifie un ou plusieurs gènes pour passer d'une solution à une autre solution de forme similaire mais qui peut avoir une évaluation totalement différente. L'individu muté n'est pas une solution "miracle". Il ne sera pas forcément meilleur ou moins bon, mais il apportera des possibilités supplémentaires qui pourraient être utiles pour la création de bonnes solutions. Plusieurs opérateurs de mutation existent pour le codage de permutation qui est utilisé dans la résolution du VRP. Généralement, ces opérateurs sont très semblables à la création d'un voisin dans des méthodes de recherche locale ou à certaines heuristiques utilisées par ailleurs pour résoudre le VRP. Ces méthodes ayant déjà été décrites dans le chapitre 1, les paragraphes suivants ne présentent que six types de mutation plus communément utilisés.

Insertion

L'opérateur d'insertion choisit un gène aléatoirement et l'insère dans une autre position du chromosome.

Echange (1-Opt ou Swap mutation)

L'opérateur d'échange consiste à prendre au hasard deux gènes (2 clients) du chromosome et à les permuter.

Inversion

L'inversion inverse l'ordre de visite des clients entre deux points de coupure choisis aléatoirement.

La figure (3.34), issue de (Alba et Dorronsoro, 2004), donne des exemples des trois mutations précédentes.

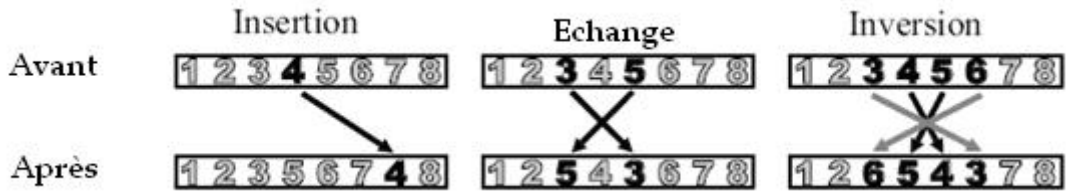


Fig. 3.34 – Opérateurs d'insertion, échange et inversion (Alba et Dorronsoro, 2004)

λ-Opt

L'opérateur λ-Opt enlève λ arcs (avec λ = 2, 3, 4,…. n) pour remettre les chaînes associées dans la meilleure combinaison possible. Pour plus d'information, voir section 1.5.3.1 du chapitre 1.

Or-Opt

L'opérateur Or-Opt opère de la même manière que la mutation 1-opt, mais les tailles des suites choisies peuvent varier entre 0 et 3 (figure 3.35). Cet opérateur échange deux suites ou il déplace une suite vers une autre position si la taille de l'une des suites est égale à zéro.

Fig. 3.35 – Or-opt

Mélange (Scramble Mutation ou SM)

Cette mutation fut proposée par (Syswerda, 1991). Elle permute aléatoirement les clients entre deux points de coupure choisis au hasard comme dans la figure (3.36)

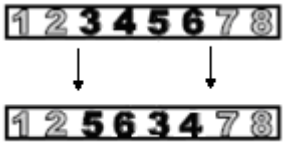


Fig. 3.36 – Exemple de *Scramble Mutation*

Noter que les mutations (1-Opt, λ-Opt et Or-opt) peuvent être trouvées dans la littérature sous le nom de mutation *échange* (*Exchange Mutation* ou EM).

3.2.6 L'évaluation et la sélection pour remplacement

Après l'application des opérateurs de reproduction (variation), chaque individu dans la population des enfants est évalué. Puis, la sélection pour le remplacement ou plus simplement le remplacement détermine quels individus devront disparaître de la population à chaque génération et quels individus forts survivent dans la population de la prochaine génération. Les paragraphes suivants présentent quatre types de stratégies de remplacement, en considérant que

- μ est la population courante (de génération courante),
- λ est la population des enfants
- et μ^* est la population de prochaine génération.

3.2.6.1 Remplacement générationnel

Cette stratégie est proposée par (Holland, 1975). Ici, la population des enfants remplace systématiquement la population courante. Ainsi, la population de prochaine génération est égale à la population des enfants ($\mu^*=\lambda$).

3.2.6.2 Remplacement élitiste

L'élitisme est une façon de protéger la rémanence de bonnes solutions et d'assurer leur survie tout au long de la recherche. Ici, la population de la prochaine génération μ^* est choisie à partir de la population des enfants et de la population courante ensemble ($\mu+\lambda$). Cette sélection a l'avantage de permettre une convergence (plus) rapide des solutions, mais au détriment de la diversité des individus.

3.2.6.3 Remplacement des stratégies d'évolution

Dans les stratégies d'évolution, la taille de la population de la prochaine génération est plus petite que la taille de la population des enfants. Soit on choisit les meilleurs μ individus de la population des enfants [(μ, λ) −ES], soit on les choisit de la population des enfants et la population courante conjointes [$(\mu+\lambda)$ −ES].

3.2.6.4 Remplacement de type Steady-state

À chaque génération, un (ou deux) enfant(s) seulement est (sont) généré(s). Ils remplacent le plus mauvais individu de la population courante. En d'autres termes, la population de la prochaine génération est la population courante, excepté le plus mauvais individu qui est remplacé par une ou deux solutions de la population des enfants [$\mu^* = \mu$ – la plus mauvaise solution de μ + (1 ou 2) solutions de λ]. Ainsi la population de la prochaine génération est peut-être plus grande que la population courante $\mu^* \geq \mu$).

Cependant, la plupart des algorithmes évolutionnaires sont appliqués sur des populations avec des tailles fixes en gardant au moins le meilleur individu dans la population courante.

3.2.7 Critère d'arrêt

Généralement, le cycle de génération et remplacement est répété jusqu'à ce qu'un critère d'arrêt soit satisfait. Ce critère peut être notamment un nombre fixe d'itérations (générations), un temps maximal de calcul, ou/et une solution satisfaisante. L'algorithme évolutionnaire retourne alors la (ou les) meilleure(s) solution(s) qu'il a identifiée(s) de génération en génération.

3.2.8 Conclusion

Le recensement des mécanismes évolutionnaires de la littérature présenté dans cette section représente un fond non négligeable pour débuter la conception et la mise en œuvre de la base d'éléments évolutionnaires de la figure 3.1. Il n'est pas exhaustif. En particulier, il est limité, pour de nombreux aspects, à une représentation du problème sous forme de permutation, car ce type de représentation est majoritairement utilisé pour résoudre le VRP. Il a montré

- que la conception d'un algorithme évolutionnaire pour résoudre une instance de VRP nécessite de prendre de multiples décisions, qui correspondent aux étapes clés de la conception,
- que ces décisions ne sont pas indépendantes, chacune d'entre elles peut avoir un impact fort sur les performances de l'algorithme, et certaines conditionnent les choix ultérieurs,
- que la richesse de la littérature sur ce sujet est telle qu'il n'est pas simple de déterminer quels sont les bons choix à faire face à une instance de problème à résoudre.

Le chapitre précédent tentait de guider ces choix en s'appuyant sur l'étude bibliographique du domaine, mais il a montré les limites de cette démarche. La suite de ce chapitre propose par conséquent une approche expérimentale. Cette dernière vise à

- mieux cerner les interdépendances qui existent entre les différents choix de conception,
- identifier si possible des combinaisons d'opérateurs plus performantes que les autres,
- en déduire des règles à réunir dans la base de règles de la figure 3.1, qui permettrait de guider la conception de l'algorithme pour la résolution d'une instance de VRP.

Cette démarche n'est pas totalement illustrée, en particulier elle ne porte pour l'instant que sur une variante simple de VRP, notée RP/ NVe, Tr:Dis/ Cap:Ve/Tr:Dis, mais cela représente en quelque sorte une étude de faisabilité. Le principe est celui d'une étude paramétrique. Certains éléments de l'algorithme évolutionnaire (création de la population initiale, évaluation, sélection, stratégie de remplacement, critère d'arrêt) sont figés arbitrairement et identiques pour tous les tests. Pour les éléments restants (codage, sélection des parents, croisement, mutation), il existe plusieurs choix, comme si le concepteur allait puiser dans la base d'éléments évolutionnaires. . En fait, à titre

d'exemple pour illustrer la démarche, trois opérateurs de croisement et trois opérateurs de mutation sont considérés. Les performances des 9 combinaisons croisement+mutation correspondantes sont expérimentalement comparées. Cela ne signifie pas que le croisement et la mutation sont considérés comme les éléments les plus importants d'un algorithme évolutionnaire. Tous les éléments décrits précédemment pourraient (devraient) faire l'objet d'une étude identique. Simplement, deux types d'éléments étaient suffisants pour illustrer celle-ci, et limiter leur nombre permettait de le faire sous une forme la plus simple possible. 3 types de codage et 2 types de sélection ont également été retenus. Dans le premier cas, il s'agit d'un codage direct et d'un codage indirect combiné avec deux méthodes de découpage distinctes, et le but est d'estimer la sensibilité des croisements et mutations au choix du codage (souvent réalisé avant le choix des opérateurs). Dans le second cas, il s'agit d'estimer leur sensibilité au niveau de pression sélective de l'algorithme. Dès lors, la première n'applique aucune pression sélective : elle est totalement aléatoire (elle correspond à l'extrême qui revient, en pratique, à supprimer la sélection). La seconde est très sélective (elle représente l'extrême inverse) : la solution sélectionnée est systématiquement celle qui minimise le mieux l'objectif du problème. De nombreux tests sont effectués pour résoudre l'instance de VRP considérée en utilisant diverses combinaisons de codage, sélection des parents, croisement, mutation, et les performances sont statistiquement analysées, en s'appuyant sur des plans d'expérience. La fin de ce chapitre détaille plus précisément les éléments fixes et variables de l'algorithme utilisé pour cette expérimentation. Leur choix s'est appuyé sur le travail de recensement précédent. Le chapitre suivant, quant à lui, présentera le reste de la démarche (notamment l'utilisation des plans d'expérience) et les résultats obtenus.

3.3 Détail de l'algorithme utilisé pour la campagne de tests

Cette section explique les composants des différentes versions d'algorithmes évolutionnaires appliqués à la variante RP/ NVe, Tr:Dis/ Cap:Ve/Tr:Dis du VRP dans la phase d'expérimentation. Cette variante est l'une des plus simples, elle inclut seulement un nombre limité de véhicules tous identiques et la contrainte de capacité, l'objectif étant de minimiser la distance totale parcourue. La figure (3.37) décrit de manière générale le cycle de vie des différents mécanismes qui font évoluer la population dans les algorithmes comparés pour sa résolution.

Dans un premier temps, les individus de la population sont initialisés, puis selon le codage utilisé, avant ou pendant l'évaluation, une procédure est appliquée sur les solutions soit pour les réparer soit pour les découper en tournées. Ensuite, une partie d'entre eux est sélectionnée pour être recombinée (par croisement). L'opérateur de mutation est appliqué sur certains d'entre eux et les solutions de la population suivante sont sélectionnées à partir des populations des parents et des enfants après l'évaluation des enfants. L'ensemble des opérateurs est décrit dans la suite.

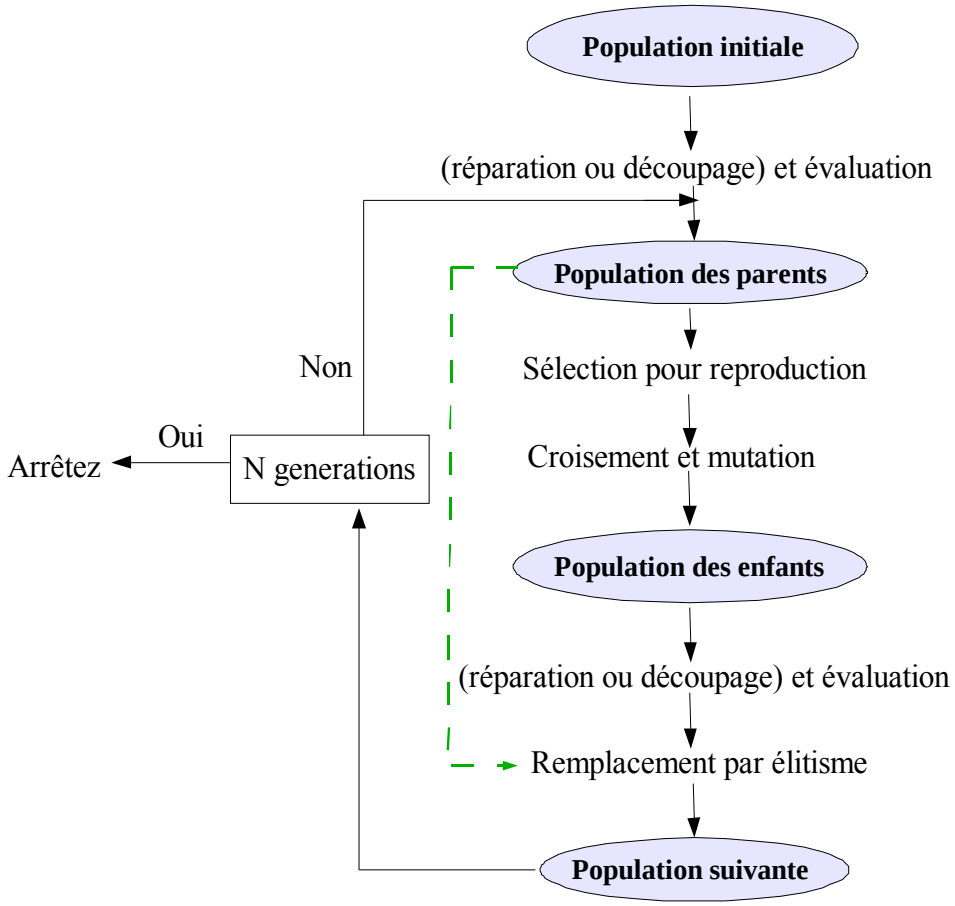


Fig. 3.37 – Schéma général commun aux différentes versions d'algorithmes utilisées

3.3.1 Codages utilisés et création de la population initiale

Le type de codage des chromosomes fait partie des paramètres pour lesquels différents choix seront possibles au cours des plans d'expérience du chapitre 4. En conséquence, deux types de codage ont été considérés : codage direct et codage indirect de permutation. Dans le codage direct, chaque solution contient à la fois les clients à servir et des séparateurs de tournées. Les valeurs [1,…, N] représentent les clients, tandis que [N+1,…, N+NVe] représentent les tournées où NVe est le nombre de véhicules. Ce type de codage a besoin, à chaque nouvelle création d'individu (à l'étape de création de la population initiale ou après application des opérateurs de croisement et mutation), d'une procédure qui vérifie la contrainte de capacité et répare les solutions qui violent cette contrainte (voir la section 3.3.1.1). Dans le codage indirect, la solution représente seulement l'ordre de visite des clients. Pour donner des solutions complètement définies, il faut construire les tournées en découpant chaque chromosome en plusieurs tournées tout en respectant l'ordre des gènes. Deux méthodes de découpage sont proposées dans (3.3.1.2).

Quel que soit le type de codage et la procédure de réparation ou de découpage utilisés, la population initiale est générée aléatoirement. Cette initialisation est une bonne approche pour favoriser la diversité des solutions (Le Bouthillier, 2000). Plus les solutions de la population de départ seront différentes les unes des autres, plus il

existera de chances d'y trouver des bonnes solutions. Un des désavantages de cette initialisation est que la recherche prend beaucoup de temps pour atteindre la convergence (Zhu, 2000). Ce paramètre, et la taille de la population qui est égale à 100, ne font pas partie des variables des plans d'expérience. C'est pourquoi leur choix a été radicalement réducteur. D'une part, ajouter des heuristiques dans la procédure de création de la population initiale aurait pu biaiser les observations faites pour les différentes variables du plan d'expérience, en particulier le cas où une heuristique s'avèrerait particulièrement efficace pour résoudre une variante donnée de VRP plutôt qu'une autre.

D'autre part, étant donné le nombre important de tests à exécuter, choisir une taille de population faible permet de limiter le temps d'expérimentation pour chaque test. Cela, comme l'absence d'heuristiques dans la création de la population initiale, peut nuire à la qualité des solutions finalement retournées. Mais le but ici n'est pas d'être aussi performant que d'autres approches de la littérature, mais plutôt de comparer l'efficacité des différents paramètres variant dans les plans d'expérience, en gardant le reste de l'algorithme totalement identique par ailleurs.

La section suivante complète l'explication donnée ci-dessus pour le type de codage direct, en présentant la procédure de réparation utilisée dans ce cas.

3.3.1.1 Validation de la capacité pour le codage direct

Dans le cas du codage direct, une heuristique, appelée *capacity validation*, répare les solutions infaisables qui ne respectent pas la contrainte de capacité. Cette heuristique fonctionne comme suit pour chaque solution :

1. d'abord, elle calcule la somme des demandes de clients pour chaque véhicule (route) dans le chromosome (solution) et détecte les véhicules pour lesquels la contrainte de capacité n'est pas satisfaite,

2. puis pour chaque tournée ne respectant pas la contrainte, elle enlève la chaîne de gènes (clients) qui provoque la violation et essaie de l'ajouter à la route d'un autre véhicule en tenant compte de la contrainte de capacité mais également de la distance entre la fin des différentes tournées existantes et le premier gène de la chaine à replacer. Ainsi, cette procédure suit les étapes suivantes :

   a. ajouter le premier gène (client) (i) de la chaîne à insérer dans la tournée du véhicule (v) tel que

   - la demande de ce client peut être ajoutée à son chargement sans violer la contrainte de capacité,
   - et dont la tournée se termine par le gène (client) le plus proche de (i) parmi les gènes finaux de toutes les tournées de véhicules dans lesquelles (i) peut être ajouté sans violer la contrainte de capacité.

   b. essayer d'ajouter le gène suivant de la chaîne à insérer dans la tournée du même véhicule, et ainsi de suite jusqu'à la fin de la chaine si cela est possible tout en respectant la contrainte de capacité. Sinon considérer que le gène courant à insérer est le premier gène de la chaîne à placer et retourner en (a).

3. S'il reste au moins un véhicule qui ne respecte pas la contrainte de capacité, l'heuristique retourne en (2), sinon elle s'arrête.

Pour illustrer ce fonctionnement, considérons un exemple contenant quatre routes : A qui vaut [6 2 5 4 9], B qui vaut [1 3], C qui vaut [8 7] et D qui ne contient que [10]. Supposons que la chaîne (5 4 9) de la route A ne satisfait pas la contrainte de capacité. Supposons de plus que toutes les routes (B, C et D) sont capables d'intégrer le client 5 sans violation de la capacité et que le client 7 est plus proche du client 5 que 3 et 10. Alors, l'heuristique ajoute 5 à la route C et essaie d'ajouter 4 et 9 à la même route C. Si seul le client 4 peut être ajouté sans dépasser la capacité, que le client 9 peut être ajouté aux routes (A, B et D) et que le client 3 est plus proche du client 9 que 2 et 10, les routes obtenues seront : A qui vaut [6 2], B qui vaut [1 3 9], C qui vaut [8 7 5 4] et D qui vaut [10]. Cette heuristique est répétée jusqu'à l'obtention d'une population de 100 individus faisables.

3.3.1.2 Méthodes de découpage pour le codage indirect

Dans le cas du codage indirect, deux méthodes de découpage sont proposées. Elles sont appelées découpage (A) et découpage (B), et servent à découper chaque chromosome (solution) en plusieurs tournées en respectant l'ordre de ses gènes :

- la méthode de découpage (A) est une méthode très simple, qui repose seulement sur la contrainte de capacité. Autrement dit, elle construit progressivement les tournées. Elle commence par le premier gène dans le chromosome et ajoute successivement les clients du chromosome dans la tournée du premier véhicule tant que l'ajout respecte la contrainte de capacité. Lorsque l'ajout d'un client viole cette contrainte, la tournée est supposée être complète et ce client est utilisé pour commencer la construction de la tournée d'un autre véhicule et ainsi de suite.
- la méthode de découpage (B) prend en compte les distances entre les clients et le dépôt et la contrainte de capacité. Le calcul des distances entre les clients et le dépôt est utilisé pour sélectionner les meilleurs endroits pour placer des séparateurs dans le chromosome, de manière à choisir en priorité les coupures au niveau des clients les plus proches du dépôt. Le nombre de séparateurs ainsi placés est égal à (2 x NVe), où NVe est le nombre de véhicules. Ensuite, la somme des demandes pour chaque tournée ainsi construite est calculée. Puis l'heuristique tente de concaténer chaque tournée avec la tournée suivante si cela est possible sans violer la contrainte de capacité. Dans le cas contraire, la tournée est validée et la procédure de calcul de distance parcourue fait la somme des distances entre le dépôt et le premier client, puis entre les clients, et enfin entre le dernier client de la tournée et le dépôt. Toutes les tournées sont traitées de cette manière jusqu'à atteindre la fin du chromosome.

3.3.2 Fonction d'évaluation

La fonction de fitness ou d'évaluation permet de mesurer l'aptitude d'une solution à survivre. Dans le problème considéré pour les tests, l'objectif est de minimiser la

distance totale parcourue par les véhicules (D).Dans l'algorithme, plus la fonction de fitness d'un individu est élevée plus il aura de chance de survivre dans la génération suivante. Aussi pour minimiser l'objectif, la fonction de fitness F donnée par l'équation (3.1) est utilisée :

$$F = \text{Integer Max\_Value} / 2 - D \qquad (3.1)$$

Où :

Integer Max_Value : est un grand nombre entier existant dans la bibliothèque utilisée pour l'implantation (voir chapitre 4).

Ainsi, le choix de la fonction d'évaluation a privilégié une forme très simple, pour les mêmes raisons que celles évoquées dans la section 3.3.1 consacrée à la génération de la population initiale.

3.3.3 Sélection pour reproduction

La taille des populations reste égale à 100 solutions tout au long de l'algorithme (population initiale, des parents, des enfants, et de la génération suivante). La population courante est supposée être la population des parents : avant chaque croisement, deux individus sont sélectionnés dans la population courante, en utilisant l'un des deux types de sélection suivants :

- sélection aléatoire: les deux parents sont successivement choisis au hasard dans la population courante, selon une loi de probabilité uniforme;
- sélection par tournoi binaire : pour sélectionner chaque parent, deux solutions sont tirées au hasard dans la population courante, et la meilleur d'entre elles (celle qui a la valeur la plus élevée de fitness) est retenue comme parent.

A nouveau, la simplicité a été privilégiée. L'objectif est de comparer une sélection purement aléatoire (qui sert en quelque sorte de référence) et une sélection qui prend en compte la valeur de fitness.

3.3.4 Croisement

Trois types de croisement sont utilisés pour recombiner chaque paire de parents sélectionnée dans la population courante. Tous ces opérateurs de croisement produisent deux enfants qui héritent certaines caractéristiques de ces parents. Il s'agit de *Partially Mapped Crossover* (PMX), *Order Crossover* (OX) et *Uniform Crossover* (UX). Ces opérateurs ont été choisis, à nouveau pour leur simplicité et parce qu'ils figurent parmi les plus utilisés pour résoudre le VRP. Dans le cas du codage direct, les séparateurs des tournées sont considérés par ces trois opérateurs comme des gènes qui peuvent être copiés d'un parent à un enfant.

La taille de la population des enfants vaut 100. Dans cette population, le taux de croisement (par exemple Pc= 80%) indique combien d'enfants sont produit par l'opérateur de croisement. Le reste des enfants (20%) sont tout simplement des copies des parents. Les taux de croisement utilisés ont été déterminés empiriquement pour chaque type d'opérateur de croisement (voir le chapitre 4).

3.3.5 Mutation

Pour préserver la diversité, un opérateur de mutation est appliqué après le croisement au sein de la population des enfants avec un taux de mutation faible. Trois cas ont été étudiés : mutation d'inversion, mutation d'échange ou 1-Opt (swapping) et mutation combinant les deux en même temps (SwapInvers). Les taux de mutation utilisés ont été fixés et égaux dans tous les tests (0,01) (voir le chapitre 4). Après application de la mutation, chaque solution est traitée par une procédure, soit de réparation (section 3.3.1.1) soit de découpage (section 3.3.1.2), selon le codage utilisé, et est évaluée.

3.3.6 Stratégie de remplacement

Comme l'initialisation de la population initiale est totalement aléatoire et qu'il existe un opérateur de mutation, une stratégie de remplacement élitiste a été choisie, afin de respecter l'équilibre entre diversité et qualité des solutions au cours de l'évolution. Cette sélection choisit les 100 meilleures solutions parmi la population des parents et celle des enfants pour créer la population de la génération suivante. Elle permet de ne pas perdre les meilleures solutions trouvées au cours de l'évolution (Michalewicz, 1994) sans gérer d'archive de solutions.

3.3.7 Critère d'arrêt

Toutes les étapes précédentes (sauf l'initialisation et l'évaluation de la population initiale) sont répétées jusqu'à ce qu'un nombre fixe de générations (itérations) soit atteint. Ce paramètre a été défini empiriquement (voir chapitre 4).

3.3.8 Conclusion

Cette section a détaillé la définition des versions d'algorithmes évolutionnaires utilisées pour la phase d'expérimentation du chapitre 4. Elle permet de distinguer un schéma général, commun à toutes les versions d'algorithmes, composé d'éléments fixes qui ont été choisis de manière arbitraire. Il s'agit en particulier

- de la taille et de la procédure de création de la population initiale,
- de la fonction d'évaluation,
- de la stratégie de remplacement
- du critère d'arrêt.

Tous ces éléments ont été choisis parce qu'ils sont communément utilisés, et aussi simples que possible pour faciliter leur implantation et minimiser la durée de la phase d'expérimentation présentée dans le chapitre 4. Quelques règles de base ont également guidé ces choix, comme par exemple, le respect d'un certain équilibre entre mécanismes favorisant la qualité des solutions et mécanismes favorisant leur diversité. De plus, l'ajout d'heuristiques spécifiques a également été évité à ce niveau pour ne pas nuire à la lisibilité des résultats. En effet, l'objectif des tests est de déterminer l'influence de divers paramètres ou combinaisons de paramètres de l'algorithme à travers un plan d'expériences.

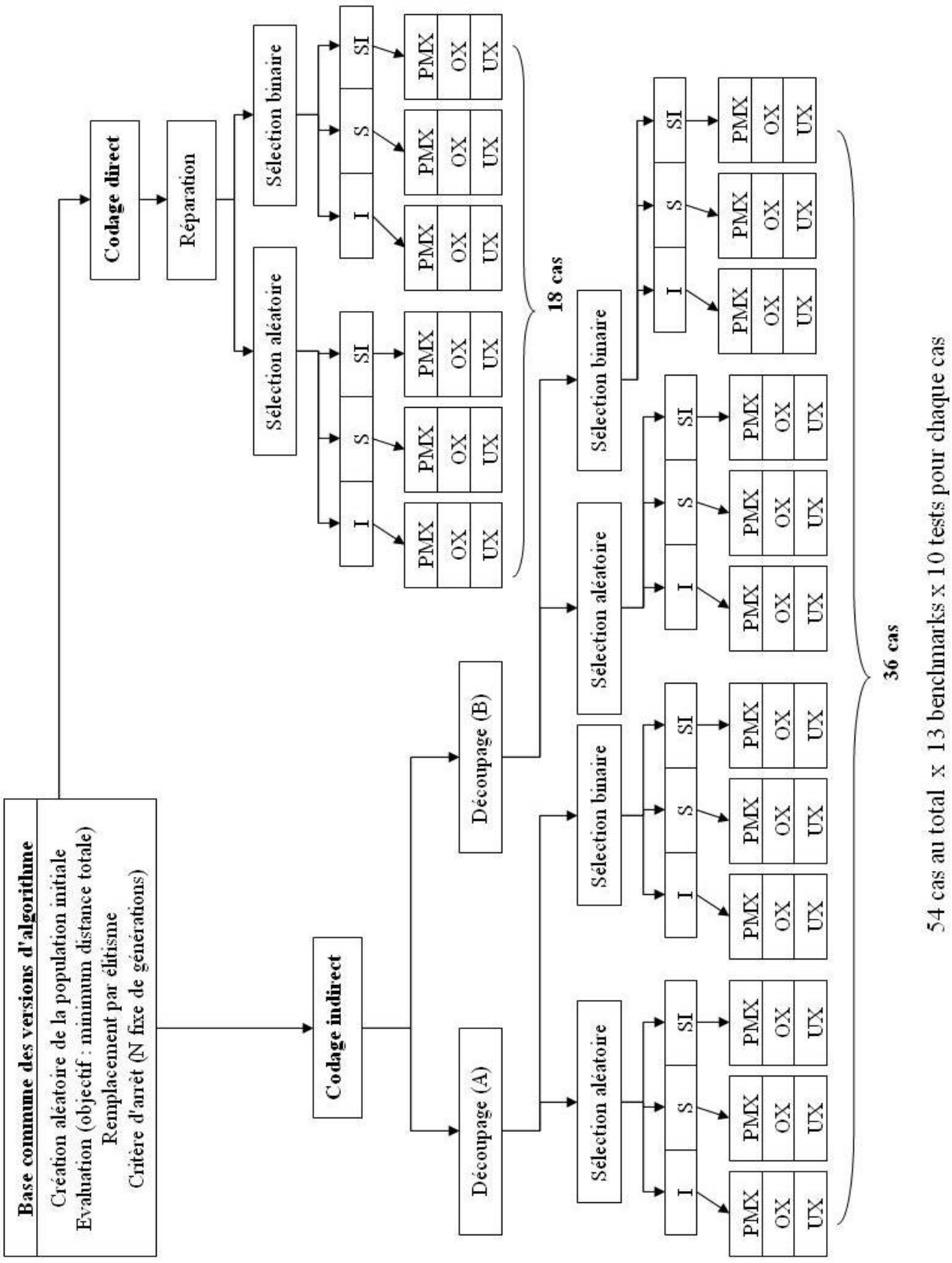


Fig. 3.38 – Schéma des 54 cas étudiés

Ainsi, quatre des éléments nécessaires à sa conception sont variables d'une version d'algorithme à l'autre. Il s'agit

- du codage et de la procédure (de réparation ou de découpage selon les cas) associée,
- de la sélection des parents,
- du type de croisement,
- du type de mutation.

Ainsi, les expériences du chapitre 4 s'appuieront sur 54 versions d'algorithme différentes, représentées dans la figure (3.38). Chaque version sera testée 10 fois (test ou *run*) sur 13 *benchmarks* de (Christofides et Eilon, 1969), et leurs résultats seront analysés et comparés sous plusieurs angles. Le chapitre 4 donne plus de détails sur les conditions et la démarche d'expérimentation.

3.4 Conclusion

Ce chapitre a exposé les principes essentiels d'un projet destiné à guider la conception d'algorithmes évolutionnaires pour la résolution d'une variante donnée de VRP.

L'introduction a montré comment une base de règles liant les notations des problèmes de tournées aux éléments évolutionnaires de la littérature pourrait permettre de conseiller une version d'algorithme plutôt qu'une autre en présence d'une variante particulière de problème.

La suite du chapitre s'est employée à présenter, tout d'abord, quels seraient les éléments à réunir dans la base d'éléments évolutionnaires. Le recensement effectué à cet effet a été limité à un sous-ensemble se focalisant plus particulièrement sur les codages de permutation et les opérateurs associés. Il a permis, au cours de la présentation, de rappeler également de manière un peu plus détaillée le fonctionnement des algorithmes évolutionnaires et leurs principaux fondements en termes d'exploration de l'espace de recherche et de diversification de la recherche. Cette partie a mis en évidence les nombreux choix à effectuer pour concevoir un algorithme évolutionnaire, mais également et la richesse de la littérature en termes d'éléments évolutionnaires disponibles, notamment pour les opérateurs de croisement. L'ensemble de ces choix a un impact sur les performances de l'algorithme, ces choix sont interdépendants et il n'est pas simple de déterminer quels éléments utiliser dans l'algorithme pour résoudre une variante de VRP donnée.

Le projet présenté consiste donc à déterminer empiriquement quels éléments, ou quelles combinaisons d'éléments, sont à conseiller selon la variante de VRP considérée. Le chapitre 4 illustre cette approche pour une variante de VRP simple, notée RP/ NVe, Tr:Dis/ Cap:Ve/Tr:Dis, caractérisée par un nombre limité de véhicules, tous identiques, de capacité limitée, et dont le but est de minimiser la distance totale parcourue. La démarche proposée pour identifier les éléments à conseiller pour l'algorithme repose sur la notion de plan d'expérience. Différentes versions d'algorithme sont utilisées pour résoudre plusieurs instances de la variante du problème étudiée, puis leurs performances sont comparées.

La fin du chapitre 3 a détaillé les caractéristiques de ces algorithmes, qui reposent tous sur une base commune dans laquelle quatre éléments ont été fixés arbitrairement, il s'agit de

- la procédure de création de la population initiale,
- la fonction d'évaluation,
- la stratégie de remplacement
- et du critère d'arrêt.

Trois objectifs ont guidé le choix de ces éléments, ils devaient :

- être simples et couramment utilisés,
- être complémentaires pour maintenir l'équilibre qualité/diversité des solutions nécessaire au bon fonctionnement de l'évolution,
- limiter les risques de développer une base commune très sensible au type de variante de VRP étudiée, pour ne pas nuire à la lisibilité des résultats.

Les autres composants varient d'une version à l'autre d'algorithme, afin d'examiner l'impact de leur choix sur les performances de résolution. Ainsi, deux types de codage de permutation, deux types de sélection des parents, trois types de croisement et trois types de mutation peuvent être utilisés. Ceci définit 54 versions d'algorithmes, qui seront toutes utilisées dans le chapitre 4 pour résoudre 13 benchmarks de la littérature, en effectuant pour chacun dix exécutions successives.

Le chapitre 4 détaille cette démarche d'expérimentation, en particulier il présente les conditions de test, puis les résultats et leur analyse.

Chapitre 4

Estimation expérimentale de l'efficacité des opérateurs évolutionnaires, résultats

Sommaire

4.1 Introduction

Ce chapitre illustre le projet de création d'une base de règles pour guider la conception d'un algorithme évolutionnaire en présence d'une variante de VRP donnée. Les règles à établir doivent permettre de sélectionner dans une base de composants évolutionnaires les éléments les plus appropriés pour la résolution. La démarche est expliquée ici pour une variante de base du VRP, et un sous-ensemble de composants évolutionnaires basés sur une représentation de permutation. Ces derniers ont été détaillés dans le chapitre précédent, qui a montré que les composants sélectionnés permettent de générer 54 versions différentes d'algorithmes évolutionnaires, utilisant tous les mêmes procédures de création initiale, fonction d'évaluation, stratégie de remplacement et critère d'arrêt.

L'efficacité de ces différentes versions d'algorithme pour résoudre la variante de VRP considérée va ici être comparée en utilisant 13 *benchmarks* de la littérature. Les tests s'appuient sur un logiciel de statistiques, nommé JMP (*John's MacIntosh Product*), qui facilite la mise en œuvre de plans d'expérience. Il permet d'organiser, de planifier et d'optimiser la réalisation d'expériences en tenant compte des interactions possibles entre les paramètres étudiés. Un document introductif est consultable sur le site JMP[6].

Le chapitre présente tout d'abord plus précisément la variante de problème étudiée (section 4.2). Puis il décrit le jeu de *benchmarks* utilisé (section 4.3). La section 4.4 décrit ensuite la plate-forme et le travail d'implantation des 54 versions d'algorithmes construites. Le logiciel JMP et la méthodologie d'utilisation de ce logiciel pour diriger les tests sont présentés dans la section 4.5. Finalement le chapitre présente les résultats et leur analyse (section 4.6) pour conclure quant à l'identification de composants évolutionnaires à conseiller pour résoudre cette instance de VRP (section 4.7).

4.2 Variante de VRP traitée

Le problème considéré est un problème de tournées de véhicules avec contraintes de capacité (souvent désigné par l'appellation *Capacitated Vehicle Routing Problem* dans la littérature). Il peut être classifié selon la notation proposée dans cette thèse par : RP/ NVe, Tr:Dis/ Cap:Ve/Tr:Dis. Cette notation indique que ce problème de tournées est un problème sur nœuds, avec une flotte homogène contenant un nombre limité de véhicules, de capacité finie. L'objectif est de minimiser la distance totale parcourue par ces véhicules.

Cette section présente la formulation de ce problème proposée par (Bjarnadóttir, 2004). Un graphe G = (N, A) représente le problème. N = {0, . . . , n} est l'ensemble des nœuds qui représente les clients C={1, . . . , n} et le dépôt {0}. A = {(i, j): i,j$\in$ N} est l'ensemble des arcs reliant les nœuds i,j $\in$ N entre eux, qui représente les trajets. A chaque nœud est associée une demande positive de produit (di) sauf pour le dépôt (d0 =0). Chaque arc a $\in$ A est associé à la distance cij qui sépare les nœuds i et j, indépendamment du sens de parcours. Autrement dit le problème est symétrique : cij= cji. Une flotte V de NVe véhicules identiques (flotte homogène) est disponible au dépôt. Chaque véhicule a une capacité Q, suffisante pour satisfaire la demande de n'importe quel client i : Q est telle que Q$\geq$ max (di), $\forall$ i$\in$ N. ce problème détermine l'ensemble

[6] http://www.jmp.com/support/downloads/pdf/jmp_introductory_guide.pdf

des tournées qui minimisent la distance totale parcourue en respectant les points suivants :

- chaque client est visité une seule fois par un seul véhicule;
- chaque véhicule exécute une seule tournée;
- chaque tournée commence et finit au dépôt;
- la somme des demandes des clients servis par un véhicule au cours de sa tournée ne dépasse pas la capacité Q de ce véhicule.

L'objectif est donné par l'équation suivante :

$$\text{Minimiser} \quad \sum_v \sum_{i,j} c_{ij} X_{ij}^{v} \qquad i,j \in N,\ v \in V \qquad (4.1)$$

Où X_{ij}^{v} est une variable binaire qui vaut 1 si le véhicule v visite le client j après le client i, et 0 dans le cas contraire.

$$X_{ij}^{v} \in \{0,1\} \qquad \forall\ i,j \in A,\ \forall\ v \in V \qquad (4.2)$$

L'équation (4.3) assure que chaque client est visité une seule fois et par un seul véhicule.

$$\sum_v \sum_j X_{ij}^{v} = 1 \qquad \forall\ i \in C,\ j \in N,\ v \in V \qquad (4.3)$$

L'équation (4.4) définit la contrainte de capacité

$$\sum_i d_i \sum_j X_{ij}^{v} \le Q \qquad \forall\ v \in V,\ i \in C,\ j \in N \qquad (4.4)$$

L'équation (4.5) assure que chaque véhicule ne sort qu'une seule fois du dépôt :

$$\sum_j X_{0j}^{v} = 1 \qquad \forall\ v \in V,\ j \in N \qquad (4.5)$$

L'équation (4.6) assure que le nombre de véhicules qui arrivent à un nœud est le même que le nombre de véhicules qui partent de ce nœud.

$$\sum_i X_{ik}^{v} - \sum_j X_{kj}^{v} = 0 \qquad i,\ j \in N,\ \forall\ k \in N,\ \forall\ v \in V \qquad (4.6)$$

Cette modélisation sera utilisée lors de l'évaluation de chaque solution construite par les algorithmes, notamment pour vérifier que les contraintes de capacité (4.4) sont satisfaites pour tous les véhicules. De nombreuses approches de la littérature traitent cette variante de problème. En particulier Christofides et Eilon (1969) fournissent plusieurs benchmarks pour ce cas. C'est ce jeu de benchmarks, décrit dans la section suivante, qui a été retenu pour les tests présentés ici.

4.3 Jeu de *benchmarks* utilisé

Le jeu de problèmes de référence fourni par Christofides et Eilon (1969) comprend 16 problèmes au total pour le problème décrit dans la section 4.2. Chacune de ces 16 instances correspond à un nombre de clients (variant entre 6 et 100) et à une capacité donnée pour les véhicules. Plus de détails peuvent être trouvés sur le site de VRP[7]. Il existe deux catégories de benchmarks, en fonction de la manière dont les distances entre clients sont définies. La première catégorie, nommée EUC_2D contient 13 problèmes, dans lesquels les distances s'obtiennent par calcul de distance en coordonnées euclidiennes. La seconde, nommée EXPLICIT contient 3 problèmes pour lesquels les distances sont des valeurs explicitement fournies. Dans cette thèse, les 13 problèmes de type EUC_2D sont utilisés. Chacun des problèmes est associé à un fichier fourni qui contient les données suivantes :

- le nom de fichier, par exemple (E-n101-k8)
- un commentaire précisant les auteurs de l'instance, le nombre minimum de véhicules, et la valeur de l'objectif (distance) pour la meilleure solution trouvée, par exemple, pour le fichier (E-n101-k8), le commentaire est : Christophides and Eilon, Min no of trucks: 8, Best value: 817.
- le type de problème (TYPE : CVRP),
- le nombre de nœuds NB Clients (en comptant les clients et le dépôt), par exemple (DIMENSION : 101) pour une instance à 100 clients,
- la catégorie à laquelle le benchmark appartient (EDGE_WEIGHT_TYPE : EUC_2D),
- la capacité du véhicule, par exemple (CAPACITY : 200),
- les coordonnes euclidiennes (x, y) des nœuds dans le plan (NODE_COORD_SECTION),
- les demandes associées aux nœuds (DEMAND_SECTION)

Pour l'expérimentation, ces 13 benchmarks sont successivement considérés par chacune des 54 versions d'algorithmes évolutionnaires. Chaque fichier correspondant constitue les données du problème à résoudre. Il est fourni en entrée de la version d'algorithme à exécuter, dont l'implantation en langage Java est décrite dans la section suivante.

4.4 Plate-forme et travail d'implantation

Pour évaluer et comparer les résultats des 54 versions d'algorithmes partageant une base commune, une plate-forme en Java a été développée. Il existe plusieurs librairies (Packages) en java utilisables pour développer des applications d'algorithmique évolutionnaire. Chacune d'elle regroupe les concepts principaux des algorithmes évolutionnaires (comme la création de la population initiale, le choix les paramètres d'évaluation, le choix ou la création des operateurs de croisement et de mutation). Parmi

[7] http://neo.lcc.uma.es/radi-aeb/WebVRP/index.html?VRP-Intro.html

ces outils on peut citer : Jaga[8] (Java Genetic Algorithm Package), Galapagos[9] (Genetic Algorithm framework), n-genes[10] (Genetic Algorithms and Programming toolkit), JAGA[11] (java API for genetic algorithms) et enfin JGAP[12] (Java Genetic Algorithm package).

C'est ce dernier (JGAP), qui a été choisi pour les raisons suivantes. Premièrement il fait partie du domaine des logiciels libres, il est téléchargeable gratuitement, et l'accès aux codes des programmes et leur modification est librement autorisée (Open source), comme dans un package de développement générique. De plus, cette librairie est munie d'une bonne documentation, et inclut quelques exemples de problèmes réels. Elle offre la possibilité d'utiliser les opérateurs déjà programmés mais également de créer de nouveaux opérateurs. Enfin, il existe un groupe d'utilisateurs de JGAP qui offre un support technique et crée de nouvelles versions en continu.

Parmi les problèmes réels regroupés dans le package « exemples », JGAP offre une implémentation d'un problème de voyageur de commerce (nommé TSP): la classe «Salesman ». C'est à partir de cette base qu'a été construite l'application des algorithmes évolutionnaires à la variante de VRP étudiée ici. L'adaptation à ce problème légèrement différent du problème de voyageur de commerce a nécessité les modifications suivantes :

1- Construire la classe (VRP) qui lit les données des *benchmark* (comme par exemple le nombre de clients, le nombre de véhicules, la capacité, les coordonnées) décrites dans la section précédente dans un fichier texte (qui a une extension .txt). Cette classe crée également un chromosome échantillon, lance l'évolution, et retourne finalement la meilleure solution.

2- Créer les méthodes de construction des chromosomes sous forme de séquences de clients avec ou sans séparateurs selon le type de codage de permutation utilisé. Pour le codage indirect, il a suffi de modifier le nombre de gènes dans l'implémentation de TSP. Pour le codage direct, l'implémentation de TSP a été modifiée pour représenter un chromosome sous la forme suivante : les gènes sont des nombres entiers variant dans l'intervalle [1… NB Clients + NB Véhicules]. Les gènes qui représentent les clients appartiennent à [1...NB clients], et les gènes qui représentent les véhicules appartiennent à [NB Clients + 1 ... NB Clients +NB Véhicules]. Le premier gène représente toujours un véhicule.

3- Construire la classe VRP_FitnessFunction qui étend la classe (FitnessFunction) et implémente la méthode (evaluate) qui calcule la fitness de chaque chromosome dans la population. Cette fonction contient aussi l'opérateur de découpage dans le cas du codage indirect.

4- Créer une population initiale aléatoire en utilisant une méthode de JGAP nommée Genotype.randomInitialGenotype (conf). Cette méthode utilise un objet de configuration pour retourner un génotype contenant le nombre de chromosomes fixé par cet objet, chaque chromosome contenant des gènes dont les valeurs ont été aléatoirement choisies parmi les valeurs possibles.

5- Construire la classe (VRP_Config) qui crée un objet de configuration qui spécifie les opérateurs qui seront utilisés. En fait, JGAP utilise un objet de

[8] http://cs.felk.cvut.cz/~koutnij/studium/jaga/jaga.html
[9] http://www.andymeneely.com/prog/ga/galapagos/index.htm
[10] http://spc.unige.ch/tools/n-genes/index.html
[11] http://www.jaga.org
[12] http://jgap.sourceforge.net

configuration par défaut (Default Configuration) qui indique tous les paramètres à utiliser. (Default Configuration) est représenté par une classe qui contient déjà les paramètres les plus communs. De ce fait, la classe (VRP_Config) spécifiquement développée a juste besoin de fournir trois éléments d'information supplémentaires : la fonction de fitness utilisée, la configuration souhaitée pour les chromosomse et le nombre de chromosomes dans la population. De plus, (VRP_Config) contient également la méthode (CapacityValidation) qui est utilisée dans le cas du codage direct.

6- Construire les classes (VRP_PMXCrossover), (VRP_OXCrossover), (VRP_UXCrossover) et la classe (VRP_InversionMutation), qui étendent la classe (BaseGeneticOperator) pour implémenter respectivement les opérateurs de croisement PMX, OX et UX, et l'opérateur de mutation par inversion. La mutation par échange (Swapping) existe déjà dans JGAP.

7- Modifier les paramètres de réglage des algorithmes comme par exemple la taille de population, le nombre de générations, les taux de croisement et de mutation.

8- Faire évoluer la population en utilisant la méthode population.evolve () qui retourne la meilleure solution pour chaque génération.

4.5 Démarche générale d'expérimentation basée sur les plans d'expérience

Cette section présente la méthodologie qui a été choisie pour déterminer les versions les plus efficaces d'algorithmes évolutionnaires (AE). Il n'existe pas dans la littérature de résultats généraux et admis par tous pour choisir les composants d'un AE et régler ses différents paramètres, tels que la taille de population ou le nombre de générations.

Le plus souvent, les chercheurs définissent ces éléments de manière empirique. Ils exécutent l'algorithme plusieurs fois en utilisant différentes combinaisons pour finalement retenir celle qui donne les meilleurs résultats (Jason *et al.*, 2002).

(Hippolyte *et al.*, 2007) ont proposé une approche de conception originale d'algorithme évolutionnaire, destinée à spécialiser l'algorithme pour le problème réel résolu et à augmenter sa robustesse. Le domaine d'application est le dimensionnement optimal de machines électriques. L'algorithme considéré est un algorithme atypique, auto-adaptatif basé sur un système multi-agent. Les interactions entre agents font émerger le comportement évolutionnaire. En conséquence, la taille de population est variable et auto-régulée et l'algorithme ne nécessite le réglage que de deux paramètres. Les auteurs ont utilisé une démarche basée sur la méthode de Taguchi et les plans d'expériences pour concevoir l'algorithme.

Dans cette thèse nous utilisons une démarche similaire mais qui se limite à l'utilisation de plans d'expériences. L'organisation des tests s'appuie sur un logiciel, nommé JMP. Ce logiciel de statistiques contient une partie (DOE) consacrée aux plans d'expériences. Ceci permet d'organiser, de planifier et d'optimiser les tests pour déterminer les paramètres en tenant compte de leurs interactions éventuelles.

Pour décrire cette démarche d'expérimentation, la suite du chapitre présente d'abord les principes généraux de JMP, la terminologie qu'il emploie et les différents outils

d'analyse qu'il offre. Puis son utilisation pour déterminer les meilleures valeurs des paramètres et leur influence dans le cas des algorithmes évolutionnaires est ensuite décrite, avant de présenter les résultats et leur analyse.

4.5.1 Plans d'expérience et logiciel JMP : principes, terminologie et outils

Selon (Goupy et Creighton, 2009), JMP est un logiciel de statistique élaboré par la société SAS Institute Inc. qui est particulièrement bien adapté pour la construction et l'analyse des plans d'expériences. Son ensemble très complet d'outils statistiques permet la mise en forme, l'exploration et l'analyse des données expérimentales. La figure (4.1) résume la démarche générale de JMP, en six étapes :

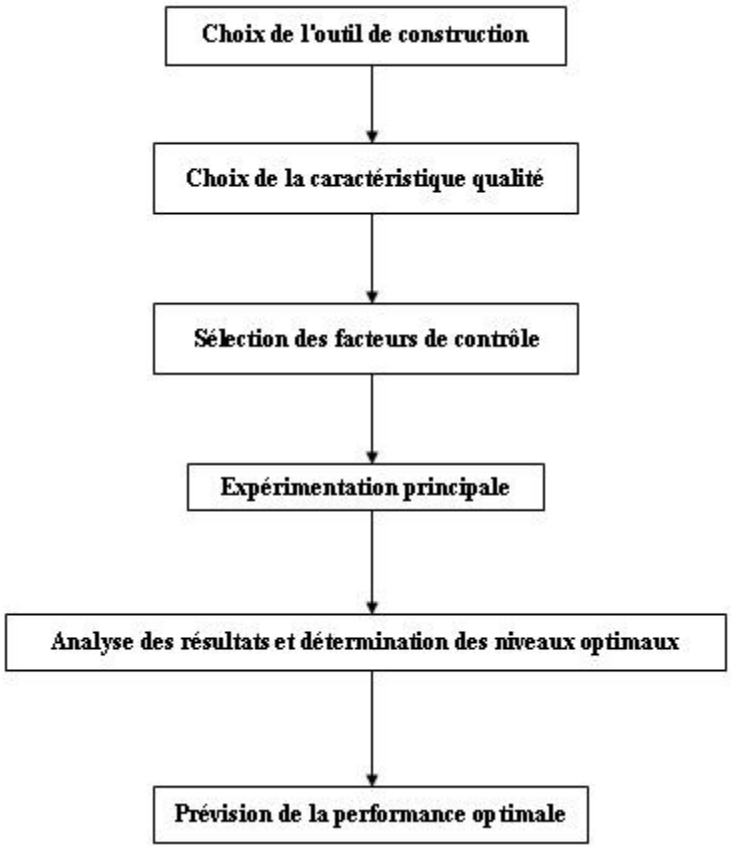


Fig. 4.1 – Démarche de détermination des paramètres sur JMP

1) la première étape est le choix du type de plans à construire, à choisir parmi dix rubriques que comporte la partie « plans d'expériences » (ou Design Of Experiment, DOE) du logiciel :
   - Custom Design (Plans sur mesures)
   - Screening Design (Plans factoriels fractionnaires)
   - Response Surface Design (Plans pour surfaces de réponse)
   - Nonlinear Design (Plans pour modèles non linéaires)

- Space Filling Design (Plans pour simulations numériques)
- Full Factorial Design (Plans factoriels complets)
- Taguchi Arrays (Tables de Taguchi)
- Mixture Design (Plans de mélanges)
- Augment Design (extension d'un plan d'expériences)
- Sample Size and Power (taille de l'échantillon et puissance)

2) puis la caractéristique qualité ainsi que les facteurs de contrôle et leurs niveaux doivent être définis. Cette phase nécessite au préalable une identification claire du problème et des objectifs à atteindre. Cette étape est essentielle pour obtenir des résultats corrects.

3) la caractéristique qualité est la mesure utilisée par le logiciel pour comparer les résultats des différents tests, en quelque sorte la fonction qu'il cherche à optimiser. Dans le logiciel (voir figure 4.2.), elle est représentée par (Y) (ou *Response*). Pour la définir, il faut préciser son nom (response Name) et si elle doit être minimisée ou maximisée (Goal). Il est également possible de préciser les limites entre lesquelles elle varie (Lower Limit et Upper Limit), et dans le cas de plusieurs réponses, de définir un niveau d'importance (Importance).

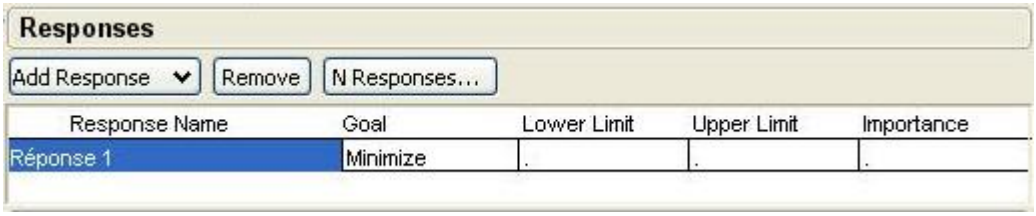


Fig. 4.2 – Informations nécessaires à la définition d'une réponse dans JMP

4) les facteurs de contrôle (X) (ou *Factors*) sont les paramètres variables d'un test à l'autre. Il faut préciser (voir figure 4.3) s'ils sont continus (*Continuous*) ou discrets (*Categorical*). Le type de facteur est très important car il est pris en compte par le logiciel pour choisir le type d'analyses et de graphiques. Les niveaux associés à chaque facteur sont les valeurs possibles pour chacun de ces facteurs au cours des tests. Ces valeurs peuvent être quantitatives ou qualitatives.

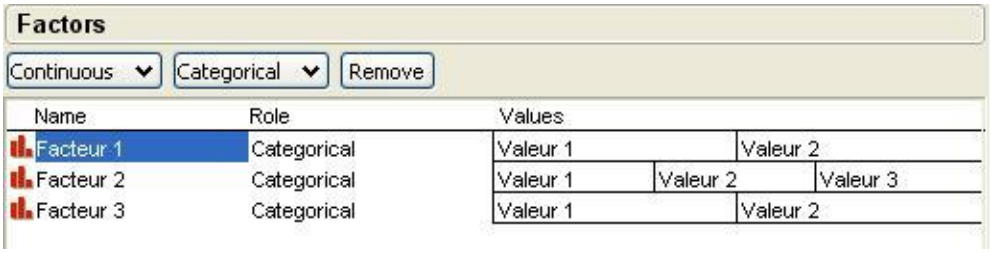
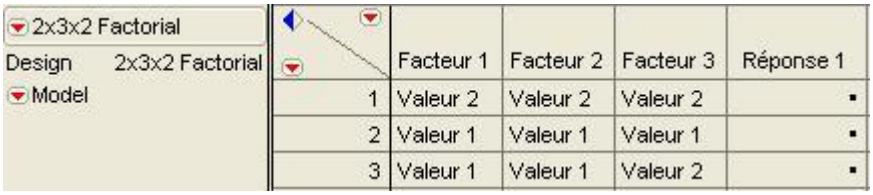


Fig. 4.3 – Définition des facteurs et niveaux dans JMP

5) à partir de ces éléments, le programme construit un plan des essais (expérimentation principale). Le plan est affiché dans un tableur (voir figure 4.4), où les facteurs sont en colonnes et les essais en lignes. Chaque ligne (mis à part la première ligne qui contient les intitulés de colonnes) représente un essai expérimental à effectuer. La colonne « Réponse » doit être complétée après l'exécution des expériences.



	Facteur 1	Facteur 2	Facteur 3	Réponse 1
1	Valeur 2	Valeur 2	Valeur 2	.
2	Valeur 1	Valeur 1	Valeur 1	.
3	Valeur 1	Valeur 1	Valeur 2	.

Fig. 4.4 – Exemple de plan sous forme de tableur dans JMP

Le plan généré peut être complet ou fractionnaire. Dans le premier cas, toutes les combinaisons possibles, en considérant tous les niveaux pour tous les paramètres, figurent dans le plan des tests. A l'inverse, un plan fractionnaire limite le nombre d'essais à un sous-ensemble de combinaisons. Celles-ci sont choisies par le logiciel de manière à pouvoir estimer statistiquement les résultats qui auraient été obtenus pour les combinaisons omises dans le plan. Cette méthode est utile en particulier lorsque l'étude comporte de nombreux facteurs, présentant eux-mêmes de nombreux niveaux, ce qui représente au total un nombre important d'essais. Elle peut également être employée lorsque les temps d'exécution ou de simulation requis pour chaque test est élevé.

6) lorsque les expériences ont été réalisées selon le plan établi, JMP analyse les résultats en utilisant plusieurs moyens (comme la construction de graphes des effets, le calcul des coefficients du modèle, la mesure de la variabilité des réponses) pour déterminer finalement les niveaux des facteurs qui donnent les résultats optimaux. Un rapport d'analyse affiche plusieurs rubriques, parmi lesquelles figurent notamment :

– *Parameter Estimates* qui indique notamment la p-value, notée Prob > |t|. Si cette probabilité est faible, le coefficient est statistiquement influent (autrement dit, il s'agit d'un facteur significatif). Si cette probabilité est forte, le coefficient est statistiquement négligeable pour expliquer les variations de la réponse. Le tableau 4.1. par exemple présente deux niveaux de facteurs différents qui ont été jugés significatifs, ainsi qu'une combinaison de deux facteurs.

Term	Prob>|t|
Niveau 1 du facteur 1	< 0,0001
Niveau 3 du facteur 2	< 0,0001
Niveau 2 du facteur 1* Niveau 2 du facteur 2	< 0,0001

Tab. 4.1 – Observation de facteurs ou combinaisons de facteurs significatifs

- *Factor Profiling* qui donne de nombreuses informations à travers plusieurs graphiques :
  - *Profiler* est un graphique interactif (représenté figure 4.5) permettant de voir l'influence des variations des facteurs sur la réponse. Les informations au bas de la figure sont les facteurs et leurs niveaux. Les courbes de la moitié supérieure de la figure 4.5 indiquent les prévisions des valeurs de la réponse en fonction des niveaux. Une ligne verticale (pointillés rouge dans la figure) peut être déplacée pour faire varier le niveau d'un facteur et l'affichage de la réponse correspondante est mis à jour pour chaque jeu de niveaux ainsi défini. Le premier nombre sur la gauche (en rouge) est la valeur prévue de la réponse pour la combinaison définie par les lignes verticales en pointillé rouge. Tandis que le second nombre (en bleu) est l'intervalle de confiance à 95% de cette valeur. De même, l'intervalle de confiance est indiqué sur les courbes, pour chaque niveau (segments bleu pour les facteurs discrets, lignes continues bleu pour le facteur génération). De plus, la rubrique *Prediction Profiler* contient la fonction de désirabilité (*Desirability Function*) qui permet d'afficher les réponses en fonction du niveau de désirabilité grâce à la commande (*Maximize Desirability*). La courbe de désirabilité baisse avec la qualité de la réponse estimée. Dans l'idéal, la désirabilité égale 0 pour la plus mauvaise valeur de réponse, et elle est égale à 1 pour la réponse optimale. Les courbes situées dans la partie inférieure de la figure 4.5 donnent les valeurs prévues pour la désirabilité en fonction des niveaux des facteurs. La valeur donnée sur la gauche est celle associée la combinaison définie par les lignes verticales en pointillé rouge.

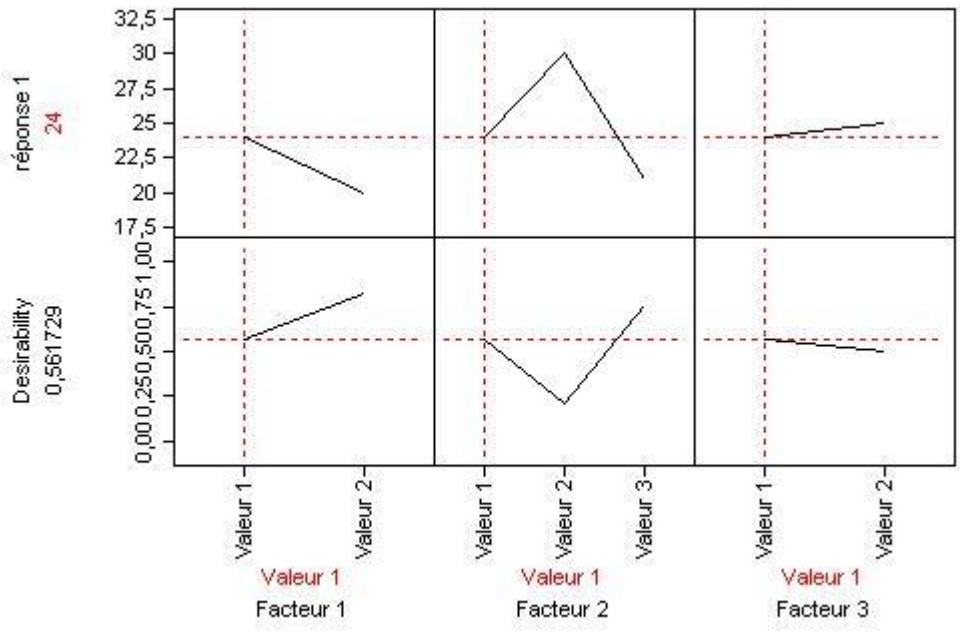


Fig. 4.5 – Exemple d'utilisation de l'outil Profiler de JMP

- *Surface Profiler* fait apparaître un graphique permettant d'étudier des surfaces de réponse en trois dimensions.

Ainsi, généralement, l'utilisation de cette démarche et du logiciel JMP a plusieurs avantages, en particulier :

- permettre une diminution considérable du nombre d'essais et une interprétation rapide,
- permettre d'étudier un plus grand nombre de facteurs et de déterminer parmi ceux-ci les facteurs influents, notamment à travers une analyse d'effet (l'effet d'un facteur sur une réponse est la variation de cette réponse en fonction des niveaux du facteur),
- pouvoir détecter des interactions éventuelles entre les facteurs (deux facteurs sont en interaction si l'effet du premier facteur dépend du niveau du deuxième),
- fournir des résultats faciles à présenter à des non spécialistes,
- pouvoir déterminer des résultats accompagnés d'une estimation de leur précision, notamment à l'aide d'intervalles de confiance.

C'est pourquoi cette démarche a été choisie pour déterminer les paramètres évolutionnaires les plus adaptés pour résoudre la variante de VRP considérée. La section suivante explique sa mise en œuvre dans ce cadre.

4.5.2 Détermination des paramètres évolutionnaires en utilisant JMP

Le chapitre 3 a présenté les 54 versions d'algorithmes évolutionnaires à comparer dans cette étude pour minimiser la distance totale parcourue dans la variante de VRP étudiée. Ces versions utilisaient toutes le même schéma de base commun, reposant sur les mêmes types de

- procédure de création de la population initiale (aléatoire);
- fonction d'évaluation;
- stratégie de remplacement (élitiste);
- critère d'arrêt (nombre fixe de générations).

D'autre part, la taille des populations (100 individus) et le taux de mutation (0,01) ont été fixés arbitrairement. En conséquence, dans la phase d'expérimentation décrite ici, les paramètres variables d'une version d'algorithme à l'autre forment deux catégories, qui font l'objet de l'étude :

- d'une part, les composants évolutionnaires pour lesquels la dernière section du chapitre 3 a listé plusieurs choix possibles : le type de codage (et la procédure de découpage associée dans le cas du codage indirect), ainsi que le type de sélection, de croisement et de mutation;
- les paramètres de réglages qui n'ont pas encore été fixés : taux de croisement et nombre de générations (qui est utilisé comme critère d'arrêt).

Ces éléments constituent par conséquent une seule réponse (la distance à minimiser) et six facteurs pour lesquels différents niveaux doivent être définis. Le tableau 4.2 présente les niveaux qui ont été choisis. Pour le taux de croisement, les valeurs données sont exprimées en %. Par exemple, le niveau 1 égal à 60 représente un taux de croisement égal à 0,6.

Facteurs	**Niveaux**							
Type de codage	Direct	Indirect avec découpage A				Indirect avec découpage B		
Type de sélection	aléatoire				binaire			
Type de croisement	PMX			OX			UX	
Taux de croisement	60	65	70	75	80	85	90	95
Type de mutation	Inversion			Swapping			SwapInvers	
Nombre de générations	5000		10000		15000		20000	

Tab. 4.2 – Facteurs à faire varier et niveaux associés

Considérer toutes les combinaisons possibles de l'ensemble de ces paramètres et niveaux génèrerait un plan factoriel complet comptant 1728 exécutions de l'algorithme évolutionnaire. De plus chaque test doit être répété plusieurs fois étant donné que ce type d'algorithme est stochastique. Enfin, ceci doit être effectué pour chacun des 13 benchmarks retenus pour les expériences. Dès lors, une approche globale de ce type aurait été difficilement réalisable, tant en termes de délai, que de lisibilité des résultats (par exemple sur les graphiques du type *Profiler*). Même sous forme de plan fractionnaire, il paraissait difficile de procéder de la sorte. Par conséquent, les expériences ont été séparées en trois phases :

- la phase **Préalable** (section 4.5.2.1 et annexe C) détermine un sous-ensemble de combinaisons intéressantes en ne considérant que les facteurs taux de croisement, type de croisement, type de mutation et nombre de générations. Seule la sélection aléatoire est utilisée et les tests sont réalisés pour le codage direct et pour le codage indirect avec procédure de découpage B (qui prend en compte la distance et la contrainte de capacité). De plus, une seule instance est utilisée parmi les *benchmarks* de (Christophides et Eilon, 1969) : E-n101-k8, qui comporte 100 clients et 8 véhicules. La planification des tests est un plan factoriel complet dans lequel chacun des 288 tests planifiés est exécuté une seule fois. 2 sous-ensembles (un par type de codage) de 16 combinaisons intéressantes chacun sont ainsi déterminés.
- les sous-ensembles résultant de la phase préliminaire sont conservés pour réaliser la seconde phase de test (nommée **Réglage**, section 4.5.2.2) destinée à régler le taux de croisement et le nombre de générations. Dix exécutions supplémentaires de l'algorithme sont réalisées pour chacun des sous-ensembles,

sur le même benchmark. Ceci permet d'analyser l'effet des paramètres en utilisant l'outil Parameter Estimates et d'observer plus en détail les performances des différentes combinaisons à l'aide du Profiler. Le résultat de cette phase est la constitution de deux sous-ensembles (un par type de codage) restreints chacun à 9 combinaisons des 4 facteurs.

– Finalement, ces 18 combinaisons sont utilisées pour appliquer l'algorithme évolutionnaire 10 fois sur chacun des 13 benchmarks considérés dans un nouveau plan d'expérience (nommé **Expérience**, section 4.5.2.3), pour les deux types de sélection (aléatoire ou binaire) et les trois types de codage (direct, indirect type A, indirect type B). Dans ce cadre, étant donné le nombre important de résultats à analyser, la présentation ne s'appuie pas sur les outils Parameter Estimates ou Profiler, mais sur des tableaux de synthèse réunissant les principales observations intéressantes dégagées par cette étude.

Phase	Instance(s)	Facteurs	Paramètres fixes	NCo	NEC
Préalable	E-n101-k8	type de croisement taux de croisement type de mutation nombre de générations	sélection aléatoire, codage direct ou indirect type B	288	1
Réglage	E-n101-k8	type de croisement taux de croisement type de mutation nombre de générations	sélection aléatoire, codage direct ou indirect type B	16 + 16	10
Expérience	13 benchmarks	sélection codage type de croisement type de mutation	Taux de croisement et nombre de générations fixés pour chaque combinaison	9 + 9	10

Tab. 4.3 – Caractéristiques des trois phases de test successives

Le tableau 4.3 résume les caractéristiques des phases d'expérimentation successives. Il précise, pour chaque phase, la ou les instances de VRP résolues, les facteurs variables ou au contraire considérés comme des paramètres fixes, le nombre de combinaisons de facteurs étudiées (Nco) et le nombre d'exécutions de l'algorithme pour chacune d'elles (NEC).

4.5.2.1 Phase de test « Préalable »

La figure (4.6) présente la définition de la réponse et des facteurs du plan factoriel complet (*Full Factorial Design*) sur lequel est basée la phase de test Préalable. Cette figure rappelle que

– la réponse (Y) est la distance (*Response name*) totale parcourue par les véhicules à minimiser (*goal*) pour chaque solution du problème de VRP considéré;

– les facteurs à faire varier au cours de l'expérience sont le taux et le type de croisement (intitulé simplement Croisement), le type de mutation (intitulé Mutation) et le nombre de générations (intitulé génération), pour lesquels les niveaux ont été définis dans le tableau 4.2. Les trois premiers sont discrets alors que le nombre de générations est continu. En effet, le taux et le type de croisement, ainsi que le type de mutation, prennent pour chaque essai la valeur du niveau spécifié au début de l'exécution de l'algorithme et cette valeur reste fixe pour toute la durée de l'essai. Par contre, le nombre de générations varie dans chaque essai de 0 jusqu'au niveau spécifié au cours de l'exécution de l'algorithme.

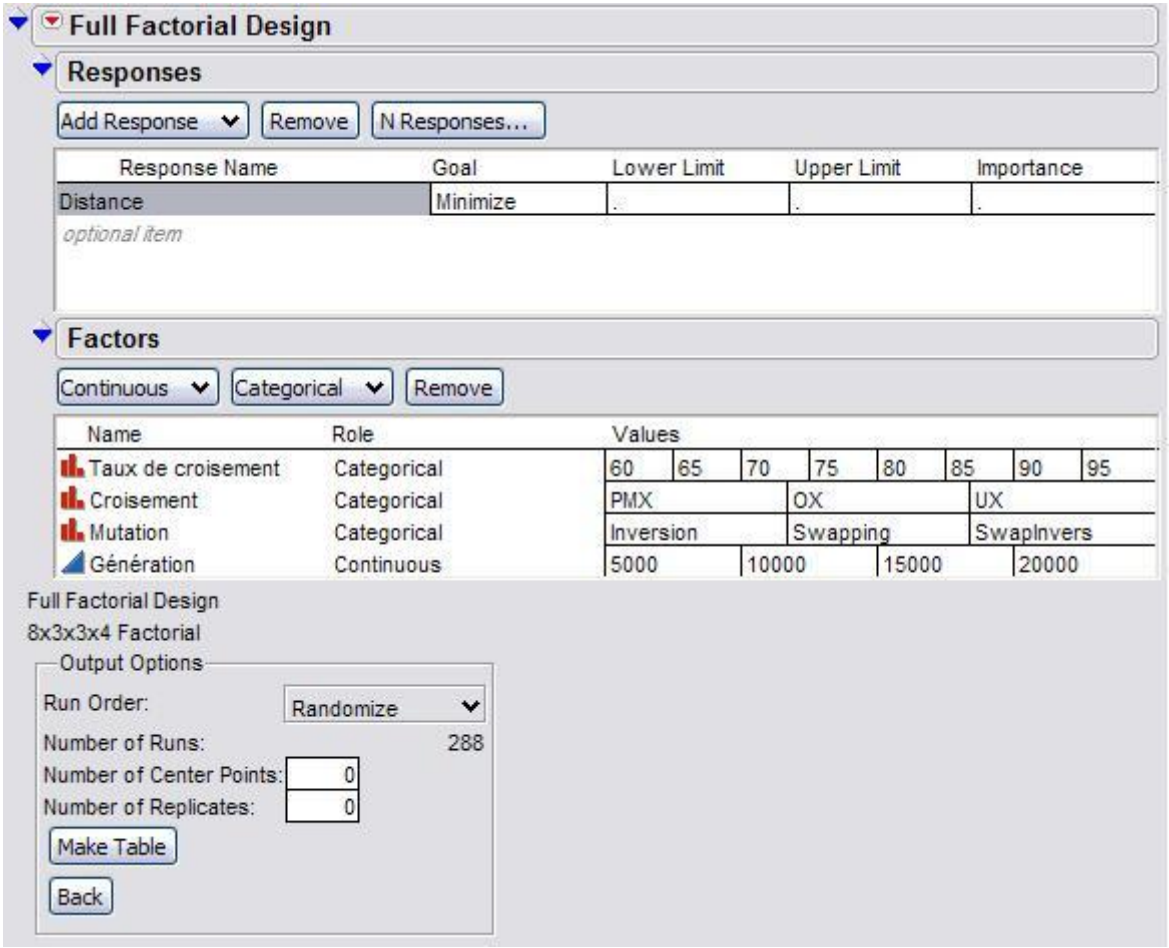


Fig. 4.6 – Définition de la réponse et des facteurs du plan factoriel à générer

La sélection et le type de codage n'apparaissent pas puisque la première sera aléatoire pour l'ensemble des tests de cette phase et pour le second deux campagnes de tests séparées seront réalisées : l'une pour le codage direct et l'autre pour le codage indirect de type B (qui prend en compte la distance et la contrainte de capacité). Enfin, l'instance de VRP résolue (E-n101-k8, issue des *benchmarks* de (Christofides et Eilon, 1969) n'est pas mentionnée non plus car le logiciel utilise seulement la valeur de la réponse obtenue

lors de chaque exécution de l'algorithme évolutionnaire pour juger de la qualité de la combinaison de facteurs correspondante.

Le logiciel présente également différentes propositions d'ordre pour l'exécution des essais (*Run order*). Par défaut, ils sont réalisés dans un ordre aléatoire (*Randomize*). Enfin, il précise le nombre d'essais que contiendra le plan factoriel (288 essais dans la rubrique *Number of runs*). Le plan d'expériences est ensuite généré à partir de ces informations (commande *Make Table*). Il est affiché dans un tableur, où les facteurs sont en colonnes et les essais en lignes.

L'annexe C fournit dans les sections C1.1.a et C1.2.a les deux plans correspondant au codage indirect type B et au codage direct. Chaque ligne représente l'essai expérimental à réaliser pour une combinaison donnée des facteurs, sauf la première ligne qui contient les intitulés de colonnes. La colonne de réponse (distance) doit être remplie par l'utilisateur après exécution des expériences avant d'effectuer l'analyse des résultats.

L'utilisateur doit au préalable définir le modèle d'analyse à utiliser (dans la fenêtre *Fit Model* représentée dans la figure 4.7, ouverte par la commande *Model > Run Script*). Ici, le modèle Construct Model Effects est choisi en précisant l'analyse statique et le type d'analyse à effectuer : l'analyse statique proposée par défaut (*Personality: Standard Least Squares*) est sélectionnée. C'est une analyse des moindres carrés, pour laquelle il faut préciser également le type d'analyse (*Emphasis: Effect Screening*). Dans le cadre de cette étude, ce modèle a été choisi car il est conçu pour les plans ayant beaucoup de facteurs et pour lesquels l'intérêt réside dans la découverte des facteurs influents. Le choix du type d'analyse ne modifie pas les calculs mais seulement la présentation et le type de sorties. En particulier, il est nécessaire de re-préciser la réponse considérée et les facteurs ou combinaisons de facteurs sur lesquels la recherche d'effet doit se focaliser.

Les tests nécessaires pour déterminer les meilleurs choix de paramètres pour l'algorithme évolutionnaire ont ensuite été exécutés en suivant les plans fournis par l'annexe C (d'une part pour le codage indirect type B, et d'autre part pour le codage direct). Les résultats obtenus en fin d'exécution de l'algorithme ont été reportés dans la colonne Distance. Finalement, la commande *Run Model* a été utilisée pour lancer l'analyse. L'annexe C fournit des résultats plus détaillés, en s'appuyant sur l'outil *Parameter estimates* et les graphiques interactifs de l'outil *Factor profiling*.

Ces résultats ont permis d'extraire les 16 résultats les plus intéressants pour chaque type de codage. Cette sélection a été faite

- en tenant compte de l'analyse des effets des sections C.1.1.b et C1.2.b de l'annexe C,
- à partir des observations des meilleures estimations données par le logiciel pour les neuf combinaisons possibles pour le couple de facteurs (croisement, mutation) dans l'outil *Prediction Profiler* (annexe C, sections C1.1.c et C1.2.c),
- de manière à conserver les 9 combinaisons ainsi déterminées mais aussi quelques autres présentant des taux de croisement ou nombre de générations légèrement différents, mais une estimation de la distance relativement proche.

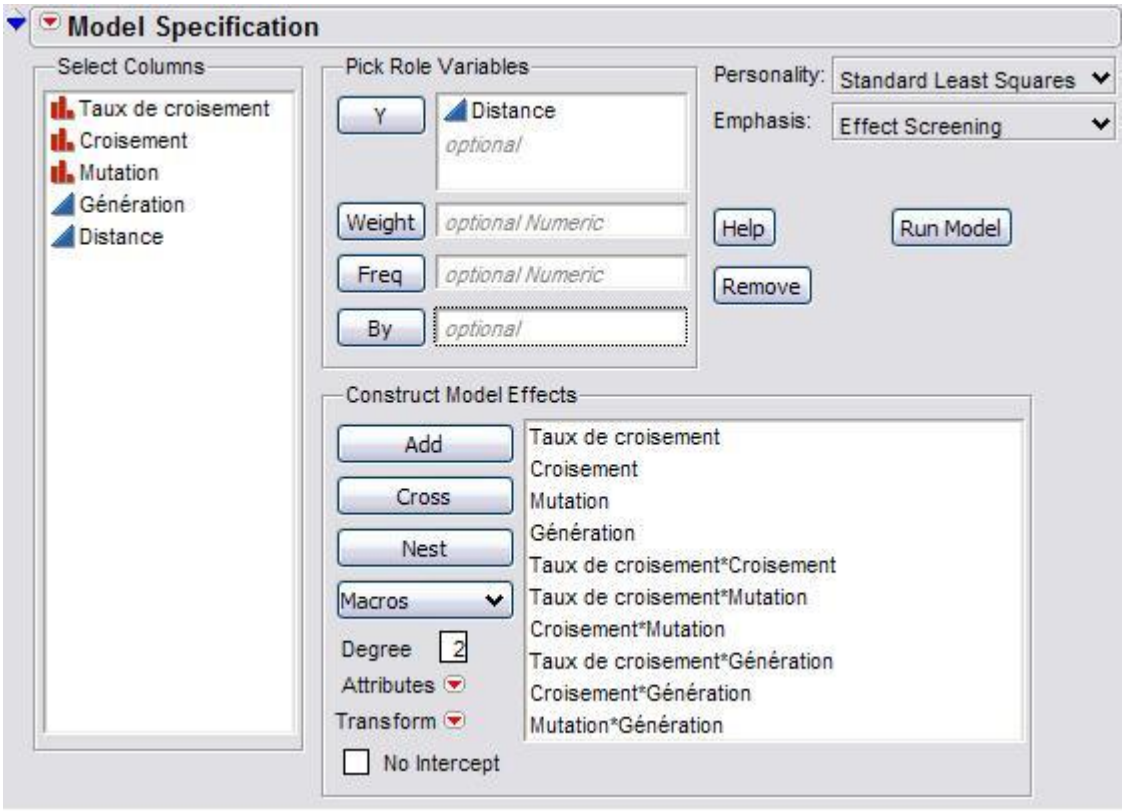


Fig. 4.7 – Définition du modèle d'analyse

Les tableaux (4.4) et (4.5) montrent ces sélections d'essais avec la valeur de distance estimée par le logiciel. Ce sont ces essais qui seront utilisés dans la phase suivante (phase de réglage) pour faire varier les facteurs croisement et mutation. Cette phase déterminera sur cette base les taux de croisement et nombre de générations qui seront utilisés pour le reste des expériences.

4.5.2.2 Phase de test « Réglage »

Dans cette phase de test, le but était de fixer le taux de croisement et le nombre de générations qui seront utilisés pour chacune des combinaisons du couple de facteurs (croisement, mutation) dans la phase Expérience. Dix exécutions supplémentaires de l'algorithme ont été réalisées pour chacun des 32 essais sélectionnés à la fin de la phase Préalable, avant de faire une nouvelle analyse. Les plans d'expérience correspondants sont présentés dans l'annexe C, sections C.2.1.a et C.2.2.a. Cette annexe présente également plus de détails sur les résultats de cette phase d'analyse. Seuls les éléments les plus marquants sont fournis ici pour chacun des codages considérés : codage direct et codage indirect type B.

Essais	Distance estimée par JMP
80 PMX Inversion 20000	1572,2
80 PMX Swapping 20000	2799,4
80 PMX SwapInvers 20000	1475,38
60 OX Inversion 20000	1110,78
80 OX Inversion 20000	1173,74
90 OX Inversion 20000	1119,23
75 OX Swapping 20000	1807,89
60 OX SwapInvers 20000	1087,55
80 OX SwapInvers 20000	1131,31
90 OX SwapInvers 20000	1133,58
65 UX Inversion 20000	1185,9
85 UX Inversion 15000	1199,56
90 UX Inversion 20000	1197,08
85 UX Swapping 15000	1345,38
85 UX SwapInvers 15000	1366,24
90 UX SwapInvers 20000	1332,11

Tab. 4.4 – Les 16 essais sélectionnés dans la phase Préalable pour le codage indirect

Essais	Distance estimée par JMP
60 PMX Inversion 10000	1012,22
85 PMX Inversion 15000	1043,01
95 PMX Inversion 10000	1019,09
95 PMX Inversion 15000	1033,39
60 PMX Swapping 10000	1400,8
80 PMX Swapping 5000	1405,09
60 PMX SwapInvers 10000	1039,55
85 PMX SwapInvers 15000	1031,97
90 PMX SwapInvers 10000	1048,51
95 OX Inversion 10000	1062,96
95 OX Swapping 10000	1090,68
95 OX SwapInvers 10000	1161,02
65 UX Inversion 20000	1143,62
65 UX Swapping 20000	1741,5
85 UX Swapping 15000	1761,01
65 UX SwapInvers 20000	1117,76

Tab. 4.5 – Les 16 essais sélectionnés dans la phase Préalable pour le codage direct

a. Résultats de la phase Réglage pour le codage indirect

La première étape de l'analyse a porté sur l'identification des facteurs influents. Pour cela, l'outil *Parameter Estimates* a été utilisé pour déterminer les valeurs de p-value pour les différents niveaux et pour certaines combinaisons de niveaux. Ces résultats sont présentés dans le tableau C.6 (section C.2.1.b de l'annexe C). Le tableau (4.6) est un

extrait du tableau C.6 dans lequel seuls les niveaux ou combinaisons de niveaux significatifs sont présentés. Il s'agit des éléments dont le logiciel a pu prouver l'influence (associés aux p-values < 0,001).

Tous les facteurs, lorsqu'ils sont considérés de manière indépendante, se révèlent influents, mis à part le taux de croisement. En conséquence, il n'apparaît pas dans le tableau. L'étude des combinaisons (croisement, mutation), quant à elle, montre que toutes les associations d'opérateurs ont un effet significatif détecté par le logiciel, sauf l'association du croisement OX et de la mutation swapping, elle aussi absente du tableau 4.6.

Term	**Prob>|t|**
Croisement[PMX]	<0,0001
Croisement[OX]	<0,0001
Croisement[UX]	<0,0001
Mutation[Inversion]	<0,0001
Mutation[Swapping]	<0,0001
Mutation[SwapInvers]	<0,0001
Génération	<0,0001
Croisement[PMX]*Mutation[Inversion]	<0,0001
Croisement[PMX]*Mutation[Swapping]	<0,0001
Croisement[PMX]*Mutation[SwapInvers]	<0,0001
Croisement[OX]*Mutation[Inversion]	<0,0001
Croisement[OX]*Mutation[SwapInvers]	<0,0001
Croisement[UX]*Mutation[Inversion]	<0,0001
Croisement[UX]*Mutation[Swapping]	<0,0001
Croisement[UX]*Mutation[SwapInvers]	<0,0001
Croisement[OX]*[Génération-14732,1]	<0,0001

Tab. 4.6 – Effets des facteurs de la phase réglage pour le codage indirect

Une seconde étape d'analyse a permis d'observer de manière interactive les réponses estimées par le logiciel pour chacun des essais, à l'aide de l'outil *Prediction profiler*. Les figures suivantes représentent quelques cas observables avec cet outil, pour illustrer le type d'informations sur lesquelles s'appuient les résultats. Elles illustrent les essais associés à la meilleure valeur de désirabilité pour chaque type de croisement. L'essai donnant la plus mauvaise réponse parmi les 16 combinaisons examinées est également présenté.

La figure (4.8) présente la meilleure combinaison identifiée par le logiciel (autrement dit la plus désirable, directement accessible par la commande *Maximize Desirability*) : 60 OX Inversion 20000 (ce qui représente un taux de croisement de 0,6, un croisement de type OX, la mutation inversion et 20000 générations). La distance estimée vaut (1101,095), avec un intervalle de confiance à 95% égal à +/- 100,9743. Le graphique montre que dans le cas de OX

- le taux de croisement et le nombre de générations n'ont pas beaucoup d'influence,

– le choix de la mutation a beaucoup plus d'effet, en particulier associer OX et swapping détériore significativement la valeur de la réponse, ce qui explique l'absence de cette combinaison d'opérateurs dans le tableau 4.6. En effet, la figure 4.9 montre que cette combinaison d'opérateurs n'apporte quasiment aucune amélioration de la désirabilité quand le nombre de générations augmente.

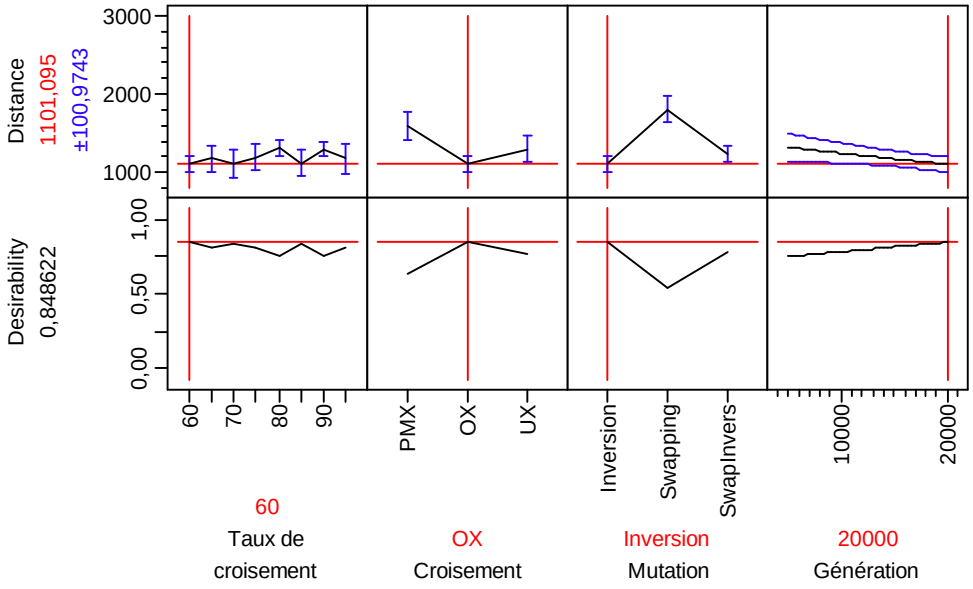


Fig. 4.8 – Exemple de graphique, combinaison (OX + Inversion) avec codage indirect

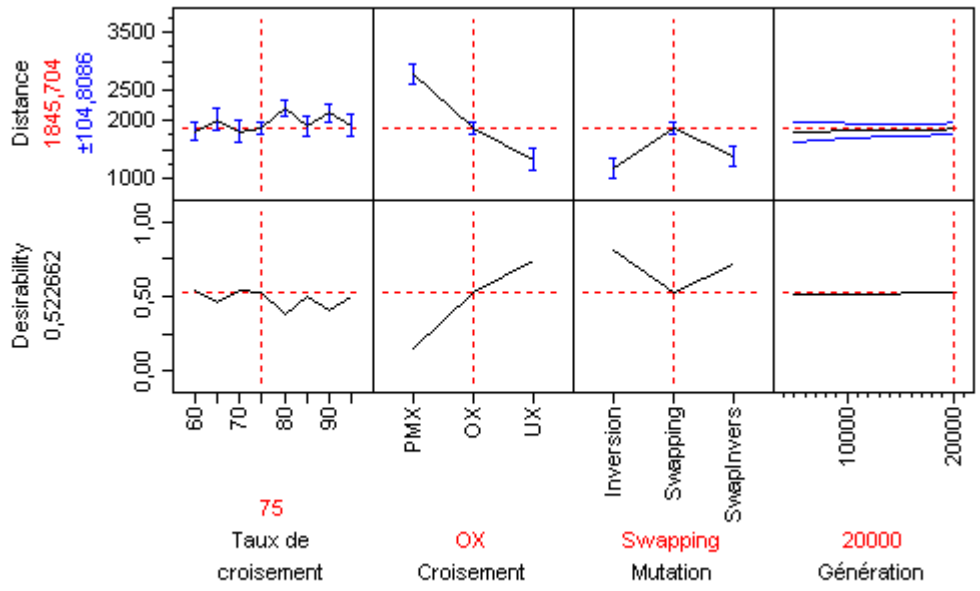


Fig. 4.9 – Exemple de graphique, combinaison (OX +Swapping) avec codage indirect

De même, dans la figure (4.10) où le croisement PMX est sélectionné, la mutation Swapping est associée à la plus mauvaise performance et aucun effet important n'est mis en évidence pour les autres facteurs, en particulier, les performances des mutations inversion et SwapInvers sont similaires, malgré un léger avantage pour l'inversion.

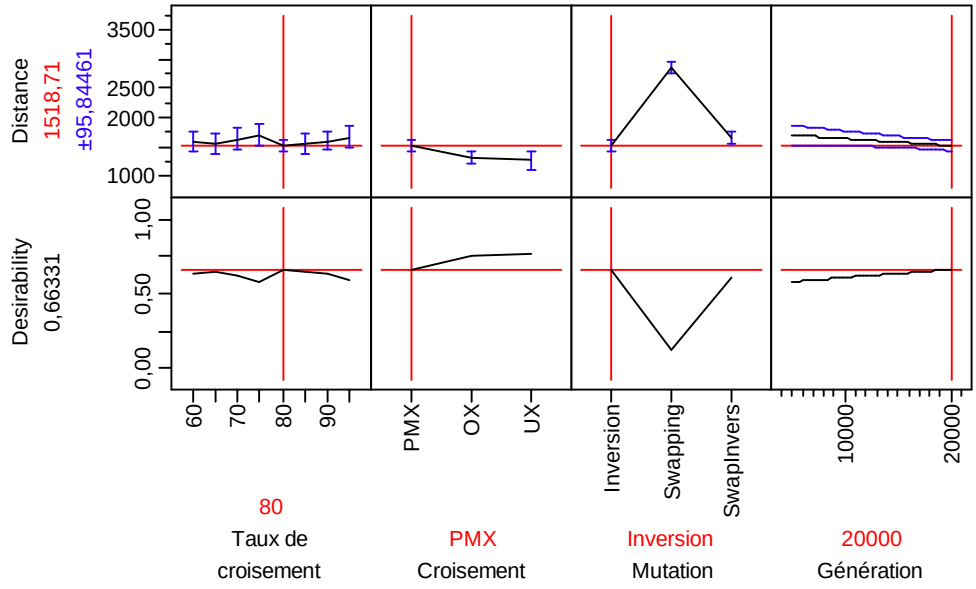


Fig. 4.10 – Exemple de graphique, combinaison (PMX + Inversion) avec codage indirect

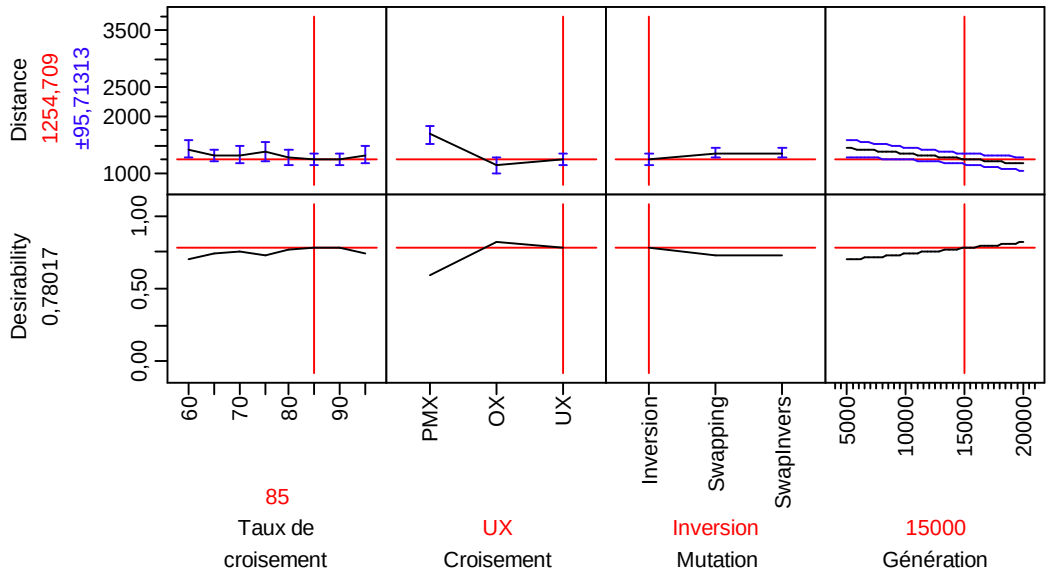


Fig. 4.11 – Exemple de graphique, combinaison (UX + Inversion) avec codage indirect

Dans la figure (4.11), qui concerne le croisement (UX), aucun des autres facteurs n'a vraiment d'effet important. Cependant, dans le cas de ce type de croisement,

l'association avec la mutation swapping donne des résultats sensiblement équivalents à ceux obtenus avec les autres types de mutation (contrairement à l'observation faite pour les deux autres types de croisement). La performance de la version d'algorithme associant UX et swapping est même bien meilleure que celle obtenue par PMX, quelle que soit la mutation associée.

Enfin, la figure (4.12) présente les paramètres les plus indésirables qui sont : un taux de croisement égal à 0.6, un croisement de type PMX, la mutation swapping et 5000 générations. La distance estimée dans ce cas est (3183,031). En fait, de manière générale, le croisement PMX semble ici le moins performant parmi les opérateurs de croisement considérés. L'association avec la mutation swapping est aussi la pire association possible ici et les variations du taux de croisement et du nombre de générations n'ont pas beaucoup d'effets.

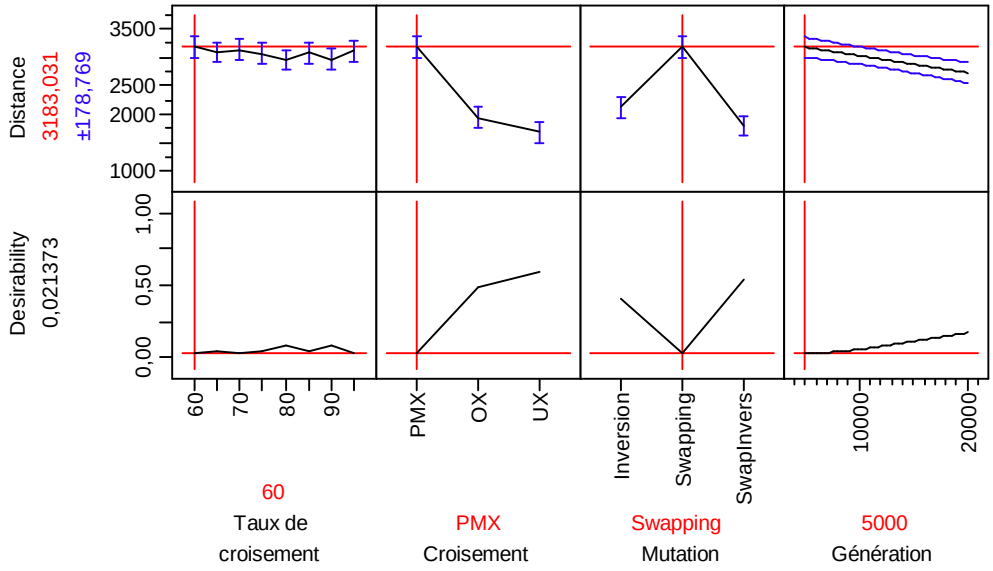


Fig. 4.12 – La combinaison la plus indésirable pour le codage indirect

Ainsi, les observations faites à l'aide de l'outil *Prediction profiler* ont permis de sélectionner les neuf combinaisons qui seront utilisées dans la suite du chapitre, et par conséquent de fixer la valeur du taux de croisement et du nombre de générations pour chaque association (croisement, mutation) étudiée.

Ces combinaisons apparaissent en gras dans le tableau (4.7), et seront toutes utilisées dans la phase **Expérience** pour exécuter l'algorithme évolutionnaire sur les 13 benchmarks de Christofides et Eilon (1969) en utilisant le codage indirect. Le tableau indique en troisième colonne les sigles qui seront utilisés pour leur désignation dans la phase expérience. Notons que, sur la base d'une seule instance (E-n101-k8), le tableau semble montrer une présence plus marquée des croisements OX et UX associés aux mutations Inversion et Swapping dans les meilleures combinaisons retenues pour le codage indirect. Mais la résolution de plusieurs benchmarks est néanmoins nécessaire pour confirmer ou infirmer cette observation.

Essais	Distance estimée par JMP	Id (chapitre 4)
80 PMX Inversion 20000	**1518,71**	**PMX+I**
80 PMX Swapping 20000	**2899,99**	**PMX+S**
80 PMX SwapInvers 20000	**1645,89**	**PMX+SI**
60 OX Inversion 20000	**1233,31**	**OX+I**
80 OX Inversion 20000	1768,93	
90 OX Inversion 20000	1764,57	
75 OX Swapping 20000	**1845,70**	**OX+S**
60 OX SwapInvers 20000	**1101,095**	**OX+SI**
80 OX SwapInvers 20000	1306,16	
90 OX SwapInvers 20000	1297,86	
65 UX Inversion 20000	1263,62	
85 UX Inversion 15000	**1254,7**	**UX+I**
90 UX Inversion 20000	1256,09	
85 UX Swapping 15000	**1358,82**	**UX+S**
85 UX SwapInvers 15000	**1359,35**	**UX+SI**
90 UX SwapInvers 20000	1516,5	

Tab. 4.7 – Résultats de la phase Réglage, 9 essais sélectionnés pour le codage indirect

b. Résultats de la phase Réglage pour le codage direct

De manière similaire, 9 combinaisons ont été sélectionnées pour le cas du codage direct, déterminant les valeurs de taux de croisement et de nombre de générations à utiliser. Le tableur de plan d'expériences contient aussi 448 essais et est fourni dans l'annexe C. Les effets des facteurs et des niveaux significatifs sont présentés dans le tableau (4.8).

Term	Prob>|t|
Croisement[PMX]	<0,0001
Croisement[OX]	<0,0001
Croisement[UX]	<0,0001
Mutation[Inversion]	<0,0001
Mutation[Swapping]	<0,0001
Mutation[SwapInvers]	<0,0001
Croisement[OX]*Mutation[Inversion]	<0,0001
Croisement[OX]*Mutation[Swapping]	<0,0001
Croisement[OX]*Mutation[SwapInvers]	<0,0001
Croisement[UX]*Mutation[Inversion]	<0,0001
Croisement[UX]*Mutation[Swapping]	<0,0001
Croisement[UX]*Mutation[SwapInvers]	<0,0001

Tab. 4.8 – Effets des facteurs de la phase Réglage pour le codage direct

Comme pour le croisement indirect, le taux de croisement n'apparait pas dans le tableau, il n'a pas été jugé suffisamment influent. En outre, pour le codage direct l'effet du facteur nombre de générations n'a pas non plus été prouvé. Enfin, l'opérateur de croisement PMX apparaît seul mais il n'apparait dans aucune association avec un opérateur de mutation, quel qu'il soit. Comme dans le cas de la combinaison OX + Swapping, la figure 4.13 montre par exemple que cette combinaison d'opérateurs n'apporte quasiment aucune amélioration de la désirabilité quand le nombre de générations augmente.

Pourtant la figure (4.13), qui concerne PMX, correspond aux paramètres les plus désirables pour le codage direct : taux de croisement de 0.6, croisement PMX, mutation inversion et 10000 générations. La distance estimée est égale à 1020,26. Cette valeur, et six autres valeurs de réponse apparaissant dans le tableau 4.9, toutes obtenues avec le croisement PMX, sont les meilleurs valeurs identifiées par l'algorithme pour le codage direct. Seule l'association avec Swapping, donne de moins bons résultats (comme pour le codage indirect), alors que la combinaison avec les mutations Inversion et SwapInvers ont des performances similaires. De plus, le taux de croisement et le nombre de générations n'ont pas de grands effets. L'ensemble de ces éléments tendrait à montrer par conséquent que PMX est particulièrement performant avec le codage direct, mais ceci de manière indépendante (ou presque) du type de mutation utilisée.

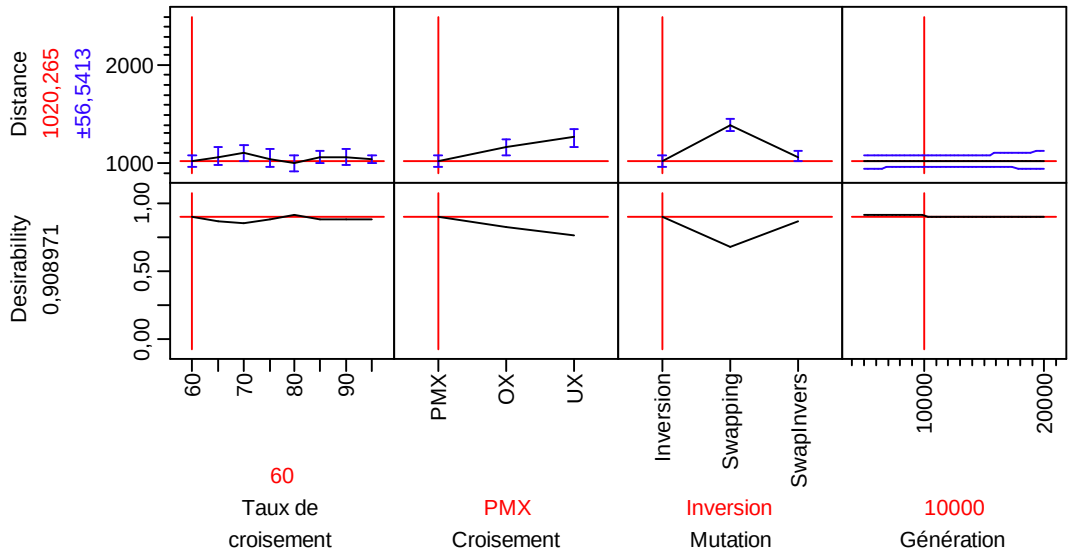


Fig. 4.13 – Exemple de graphique, combinaison (PMX + Inversion) avec codage direct

Par contre, la figure (4.14) illustre l'essai le plus indésirable parmi les 16 combinaisons étudiées : taux de croisement de 0.65, croisement UX, mutation Swapping et 5000 générations. La distance prévue est 1940,762. Ici, en sélectionnant la mutation Swapping, la courbe du type de croisement montre que le croisement qui s'associe le mieux à swapping est OX (pour lequel la courbe atteint un pic (maximal desirability)). PMX présente une désirabilité légèrement moins bonne, alors qu'UX s'associe nettement moins bien à swapping. A l'inverse la mutation Swapping est celle qui s'associe le moins bien à UX.

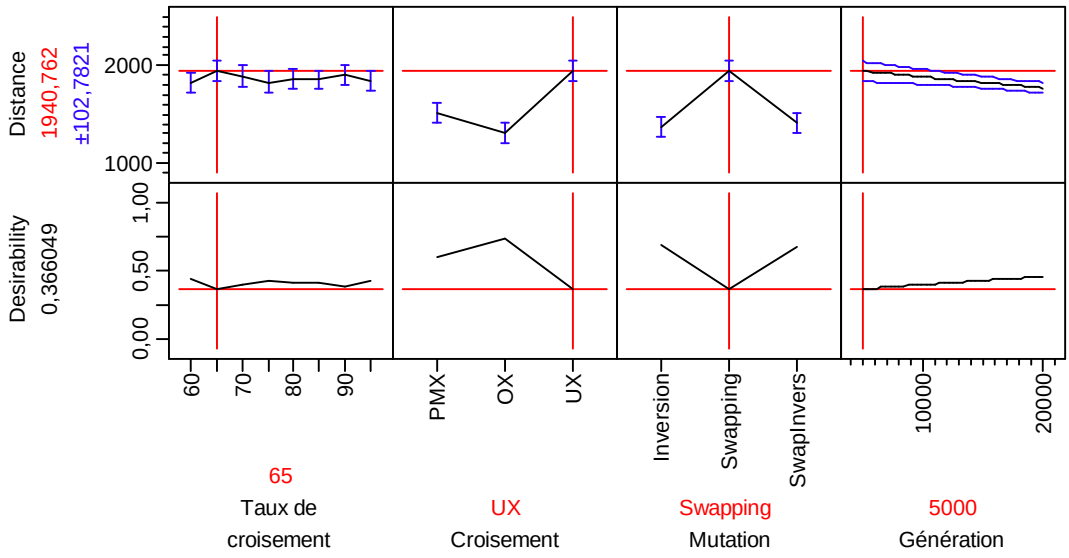


Fig. 4.14 – La combinaison la plus indésirable pour le codage direct

Dans la figure (4.15) qui concerne le croisement UX, les résultats s'améliorent légèrement avec l'augmentation du nombre de générations. Les performances des mutations Inversion et SwapInvers sont souvent similaires, alors que la mutation Swapping est la plus mauvaise. Aucun effet important n'a été indiqué pour les niveaux du facteur de taux de croisement.

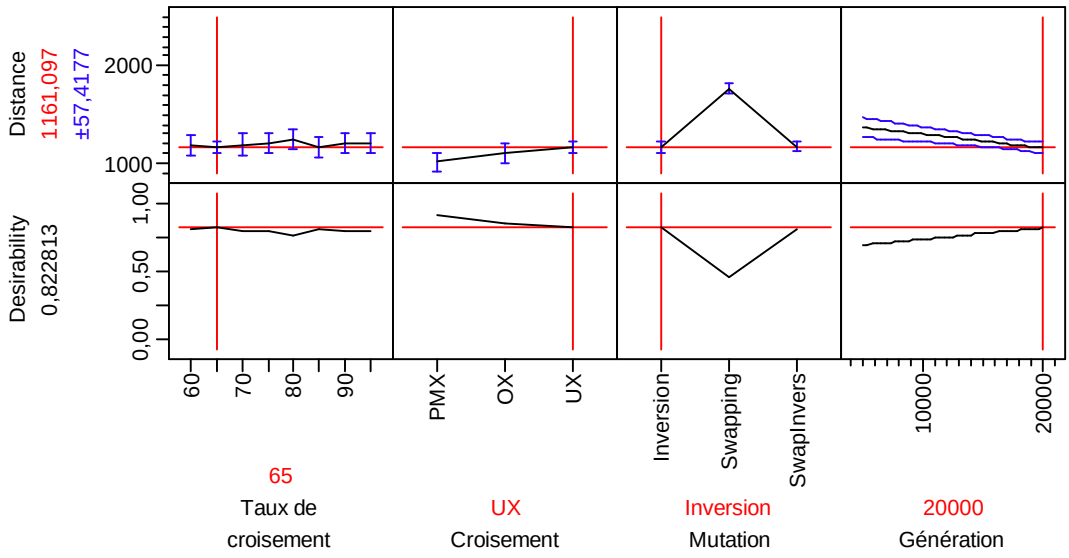


Fig. 4.15 – Exemple de graphique, combinaison (UX + Inversion) avec codage direct

Le croisement (OX) avec codage direct est l'un des rares cas où la mutation Swapping montre une performance similaire aux performances des mutations Inversion

et SwapInvers. Mais globalement, pour OX, aucun facteur n'a vraiment d'effet important, voir la figure(4.16).

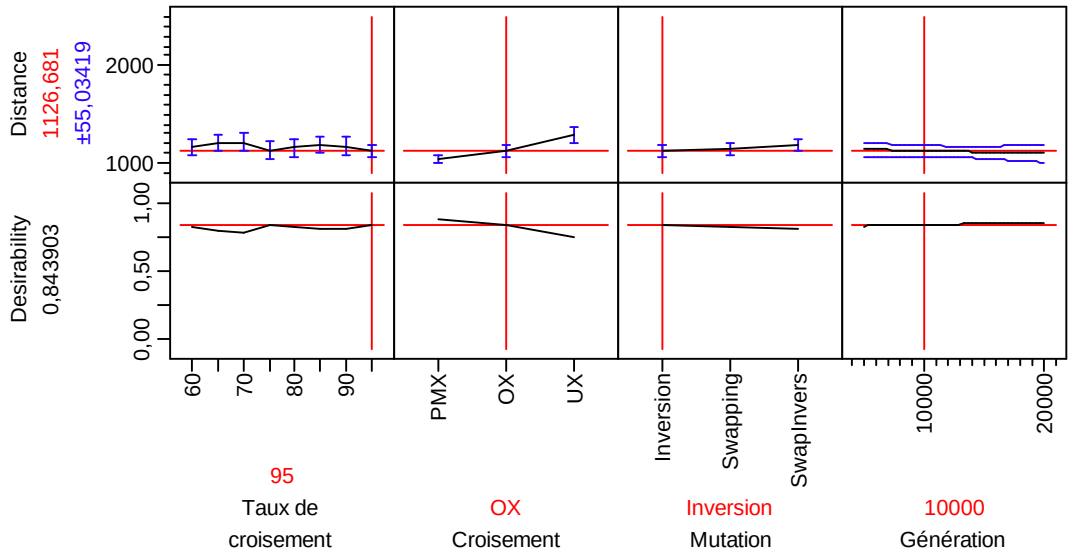


Fig. 4.16 – Exemple de graphique, combinaison (OX + Inversion) avec codage direct

Ainsi, sur la base de telles observations, 9 combinaisons ont été sélectionnées et les valeurs de taux de croisement et nombre de générations ont été déterminés pour le codage direct. Cette sélection apparaît en gras dans le tableau (4.9). Notons que le croisement PMX semble être le plus adapté pour le codage direct.

Essais	Distance estimée par JMP	Id (chapitre 4)
60 PMX Inversion 10000	**1020,26**	**PMX+I**
85 PMX Inversion 15000	1044,17	
95 PMX Inversion 10000	1045,88	
95 PMX Inversion 15000	1053,71	
60 PMX Swapping 10000	**1392,75**	**PMX+S**
80 PMX Swapping 5000	1393,22	
60 PMX SwapInvers 10000	1072,31	
85 PMX SwapInvers 15000	1069,53	
90 PMX SwapInvers 10000	**1043,81**	**PMX+SI**
95 OX Inversion 10000	**1126,68**	**OX+I**
95 OX Swapping 10000	**1147,7**	**OX+S**
95 OX SwapInvers 10000	**1185,04**	**OX+SI**
65 UX Inversion 20000	**1161,09**	**UX+I**
65 UX Swapping 20000	**1771,6**	**UX+S**
85 UX Swapping 15000	1772,81	
65 UX SwapInvers 20000	**1175,57**	**UX+SI**

Tab. 4.9 – Résultats de la phase Réglage, 9 essais sélectionnés pour le codage direct

En effet, il apparaît 9 fois dans le tableau qui compte 16 lignes et fournit avec les mutations Inversion et SwapInvers, les 7 meilleures valeurs des tableaux. Il faut toutefois attendre les résultats des tests incluant toutes les instances de problèmes pour le vérifier. Les essais en gras seront utilisés pour exécuter les versions d'algorithmes évolutionnaires correspondantes pour résoudre les 13 benchmarks de Christofides et Eilon (1969). Le tableau indique en troisième colonne les sigles qui seront alors utilisés pour leur désignation.

En conclusion, à l'issue de la phase Réglage, 18 combinaisons ont été définies en gras dans les tableaux (4.7), pour le codage indirect, et (4.9) pour le codage direct. Pour la suite des expériences, elles permettront de tester toutes les associations (type de croisement, type de mutation) possibles, sans faire varier le taux de croisement et le nombre de générations. Les valeurs de ces derniers ont été fixées pour chacune des 18 combinaisons sélectionnées. Ceci réduit le nombre de facteurs, et permet dans la suite

- d'une part de considérer les deux types de sélection et les deux types de méthode de découpage, (alors que jusqu'à présent seule la sélection aléatoire et la méthode de découpage B ont été étudiées),
- et d'autre part, d'appliquer l'algorithme évolutionnaire aux 13 benchmarks retenus pour les tests.

C'est l'objet de la phase de test Expérience, qui est décrite dans la section 4.5.2.3.

4.5.2.3 Phase de test Expérience

La dernière phase de test a pour objectif de déterminer quelle est, parmi les 54 versions décrites dans le chapitre 3, la version d'algorithme évolutionnaire la plus efficace pour résoudre la variante de VRP avec contrainte de capacité considérée. Elle ré-intègre par conséquent l'ensemble des facteurs décrits dans le tableau 4.2. mais sur la base des résultats des phases préparatoires précédentes. Ainsi, il existe deux types de facteurs :

- des facteurs exprimés sous forme explicite :
  - le type de codage, qui a trois niveaux (codage direct, codage indirect avec procédure de découpage A, basée sur la capacité uniquement et codage indirect avec procédure de découpage B, basée sur la distance et la capacité);
  - la sélection des parents pour reproduction, qui a deux niveaux : aléatoire et binaire;
- les facteurs croisement et mutation, pour lesquels trois niveaux sont définis (PMX, OX et UX, et Inversion, Swapping et SwapInvers respectivement), mais qui apparaissent de façon implicite. En effet, ils sont représentés par les 18 combinaisons sélectionnées par la phase Réglage (tableaux 4.7 et 4.9), de manière à utiliser les valeurs de taux de croisement et de nombre de générations fixés par cette dernière.

Pour chacun de ces 18 jeux d'essais, l'algorithme évolutionnaire a été exécuté dix fois (10 *runs*) pour chacun des 13 benchmarks de (Christofides et Eilon, 1969). Malheureusement, des difficultés techniques nous ont obligés à exécuter les tests sur des

machines différentes pour respecter les délais, ce qui implique que les temps de calcul des algorithmes utilisés ne sont pas objectivement comparables, c'est pourquoi ils ne sont pas fournis. Les résultats ont ensuite été comparés selon quatre critères :

- deux sont calculés pour chaque version d'algorithme et chaque benchmark, il s'agit de :
  - MEAN : la moyenne, sur les 10 *runs*, de la réponse retournée par l'algorithme à la fin de l'exécution (autrement dit la meilleure valeur de distance totale parcourue identifiée par l'algorithme);
  - BEST : la meilleure réponse parmi les 10 meilleures valeurs retournées (une par *run*) par l'algorithme en fin d'exécution.
- deux sont déterminés en considérant l'ensemble des versions d'algorithme, pour chaque benchmark :
  - B_MEAN : la meilleure moyenne trouvée pour le benchmark considéré, toutes versions d'algorithme confondues,
  - B_BEST : la meilleure réponse identifiée par l'algorithme pour le benchmark considéré, toutes versions d'algorithme confondues.

Plusieurs tableaux de synthèse ont été construits pour analyser les résultats sous divers angles. Ils apparaissent dans cette section et dans l'annexe D. Pour chacun de ces tableaux, les valeurs de MEAN sont comparées entre elles, alors que les valeurs de BEST sont comparées entre elles et avec la meilleure solution connue dans la littérature pour chaque instance (nommée MRL). Ces meilleures solutions (dont certaines sont optimales) sont consultables sur le site de *Vehicle Routing Data Sets*[13]. Elles sont fournies à titre indicatif.

Le but ici n'est pas d'entrer en compétition avec d'autres approches proposées pour la résolution du VRP dans la littérature. L'objectif est uniquement de vérifier s'il est possible de comparer les performances d'opérateurs évolutionnaires par expérimentation pour extraire des règles guidant la conception d'algorithme pour une variante de VRP particulière.

Le tableau 4.10 résume les différents angles de comparaison employés. Il les présente dans le même ordre que la suite de cette section. Il mentionne :

- les facteurs pour lesquels la valeur est identique pour l'ensemble du tableau de comparaison (Cas étudié)
- les facteurs qui font l'objet de la comparaison (Comparaison),
- le nombre de benchmarks sur lequel la comparaison est basée (N Benchmarks),
- le ou les critères utilisés pour la comparaison (Critères),
- la référence du tableau présentant les résultats (Tableaux).

[13] http://branchandcut.org/VRP/data

Cas étudié		Comparaison	N Benchmarks	Critères	Tableaux
Codage direct	Sélection aléatoire	[Croisement+Mutation] (en colonnes)	11 (en lignes)	MEAN	4.11
				BEST	4.12
	Sélection binaire	[Croisement+Mutation] (en colonnes)	11 (en lignes)	MEAN	D.1
				BEST	D.2
	Tous résultats de codage direct	Sélection	11 (en lignes)	MEAN	D.3
				BEST	D.4
		Sélection et [Croisement+Mutation]	11 (en lignes)	B_MEAN B_BEST	4.13
Codage indirect	Sélection aléatoire + Découpage A	[Croisement+Mutation] (en colonnes)	13 (en lignes)	MEAN	D.5
				BEST	D.6
	Sélection aléatoire + Découpage B	[Croisement+Mutation] (en colonnes)	12 (en lignes)	MEAN	D.7
				BEST	D.8
	Sélection binaire + Découpage A	[Croisement+Mutation] (en colonnes)	13 (en lignes)	MEAN	D.9
				BEST	D.10
	Sélection binaire + Découpage B	[Croisement+Mutation] (en colonnes)	12 (en lignes)	MEAN	D.11
				BEST	D.12
	Sélection aléatoire	[Croisement+Mutation] et Méthode de découpage	13 (en lignes)	B_MEAN B_BEST	4.14
	Sélection binaire	[Croisement+Mutation] et Méthode de découpage	13 (en lignes)	B_MEAN B_BEST	4.15
	Découpage A	Sélection et [Croisement+Mutation]	13 (en lignes)	B_MEAN B_BEST	4.16
	Découpage B	Sélection et [Croisement+Mutation]	12 (en lignes)	B_MEAN B_BEST	4.17
	Tous résultats de codage indirect	Découpage et Sélection et [Croisement+Mutation]	13 (en lignes)	B_MEAN B_BEST	4.18
Cas général		Tous : Codage et Découpage et Sélection et [Croisement+Mutation]	13 (en lignes)	B_MEAN	4.19
				B_BEST	4.20

Tab. 4.10 – Différents angles d'analyse de la phase Expérience

La première analyse qui peut être faite de manière globale porte sur le nombre de benchmarks résolus par chaque type d'algorithme. En effet, le tableau 10 montre que certaines versions d'algorithmes n'ont pas pu identifier de solutions faisables pour certains des benchmarks. Il s'agit :

- des versions basées sur le codage direct, qui ont échoué pour E-n30-k3 et E-n76-k14;
- et des versions basées sur le codage indirect et la méthode de découpage B, qui ont échoué pour E-n76-k14.

La suite de cette section résume les principales observations faites à partir des différents tableaux de résultats, dans l'ordre du tableau 10, c'est-à dire en particulier en commençant par le codage direct.

a. Analyse des résultats du codage direct

Pour le **codage direct avec sélection aléatoire**, les tableaux (4.11) et (4.12) sont utilisés pour **comparer l'efficacité des combinaisons Croisement+Mutation** du tableau 4.7 (qui figurent en colonnes), sur la base des valeurs de MEAN et de BEST respectivement. Les lignes correspondent aux onze *benchmarks* résolus avec le codage direct. Les nombres en gras dans ces tableaux montrent les meilleures performances obtenues par l'algorithme évolutionniste pour chaque *benchmark*.

Cette première analyse permet de classer ces combinaisons d'opérateurs par ordre de performance décroissant. Cela fournit le classement suivant, qui indique pour chaque combinaison son score sous la forme (nombre de meilleures moyennes fournies, nombre de meilleures solutions fournies) :

- PMX+SI (5,4),
- PMX+I (3,3),
- UX+SI (1,2),
- UX+I (1,1),
- OX+S (1,0),
- OX+I, OX+SI, PMX+S, UX+S (0,0).

Ainsi, la mutation Swapping, presque totalement absente de ce classement présente de manière générale des performances faibles lorsqu'elle est associée avec les croisements PMX et UX, mais sa combinaison avec OX donne quelquefois des résultats meilleurs que ceux d'OX avec SwapInvers.

Benchmark	OX+I	OX+S	OX+SI	PMX+I	PMX+S	PMX+SI	UX+I	UX+S	UX+SI
E-n22-k4	414,22	**397,07**	407,08	405,41	407,92	404,019	405,57	425,65	405,5
E-n23-k3	624,67	618,72	620,23	**594,13**	606,45	642,998	617,52	602,45	614,74
E-n30-k4	601,84	584,77	593,82	573,81	575,89	582,395	581,16	573,16	**548,43**
E-n33-k4	1023,1	1031,9	1028,5	1042,9	1026,3	1012,24	**967,82**	974,44	976,62
E-n51-k5	624,03	624,78	617,93	586,48	656,38	**583,89**	592,61	665,77	611,96
E-n76-k7	916,04	908,39	900,11	**847,9**	1101,9	850,333	877,86	1064,9	879,72
E-n76-k8	988,68	981,83	967,17	949,34	1009,9	**898,238**	925,72	1013,4	935,26
E-n76-k10	1086,9	1039,2	975,73	**962,92**	1019,4	965,211	963,35	1107,8	982,13
E-n76-k15	1284,8	1262	1267,4	1255,1	1234,7	**1230,65**	1316,8	1464,7	1339,9
E-n101-k8	1146,7	1166	1201,4	1010,5	1373,8	**1025,85**	1155,4	1679,2	1176,3
E-n101-k14	1446	1421,4	1474,5	1436,3	1437,7	**1388,32**	1513,2	1779,2	1550,3

Tab. 4.11 – Codage direct avec sélection aléatoire, valeurs de MEAN pour les combinaisons Croisement+Mutation

Benchmark	OX+I	OX+S	OX+SI	PMX+I	PMX+S	PMX+SI	UX+I	UX+S	UX+SI	MRL
E-n22-k4	383,93	383,52	383,52	**375,28**	377,43	377,431	377,69	377,69	377,43	375
E-n23-k3	585,26	595,53	579,25	570,56	**568,56**	588,321	571,4	571,05	571,4	569
E-n30-k4	559,95	532,59	555,28	522,62	537,24	526,124	523,63	511,91	**509,41**	-
E-n33-k4	913,99	965,87	930,31	946,98	951,74	897,459	**897,11**	910,75	909,53	835
E-n51-k5	588,03	591,6	574,94	560,94	577,54	546,985	539,36	595,78	**536,99**	521
E-n76-k7	802,32	830,51	817,84	**748,47**	970,5	792,845	816,76	829,07	789,4	683
E-n76-k8	927,99	916,25	913,49	867,64	951,23	**803,67**	853,4	903,15	884	735
E-n76-k10	953,07	995,61	967,04	**946,65**	978,79	952,55	963,8	988,46	960,1	830
E-n76-k15	1237,5	1169,2	1198,9	1212,8	1166,3	**1163,06**	1234,4	1337,1	1243,3	-
E-n101-k8	1016,2	1055,4	1077,5	968,55	1320,3	**927**	1095,8	1221,4	1060,4	815
E-n101-k14	1382,7	1342,1	1346,6	1348,6	1346	**1308,46**	1423,2	1636,2	1385,7	1071

Tab. 4.12 – Codage direct avec sélection aléatoire, valeurs de BEST pour les combinaisons Croisement+Mutation

Pour le **codage direct avec sélection binaire**, les tableaux (D.1) et (D.2) sont construits et utilisés de la même manière et conduisent au classement suivant :

- PMX+SI (4,6),
- PMX+I (4,4),
- UX+I (2,2),
- UX+SI (1,0),

– OX (quelle que soit la mutation), PMX+S, UX+S (0,0).

Ainsi, les résultats ressemblent aux résultats obtenus avec la sélection aléatoire, à quelques légères différences près :

– la mutation Inversion est légèrement meilleure que SwapInvers avec UX, alors qu'avec la sélection aléatoire c'était le contraire,

– les trois mutations donnent des résultats similaires lorsqu'elles sont associées à OX, Swapping ne se distingue plus des deux autres types de mutation quand elle est associée à OX. Par ailleurs, elle reste la mutation qui se combine le moins bien avec UX et PMX.

En conclusion, avec le codage direct, quel que soit le type de sélection utilisé, le croisement PMX associé aux mutations SwapInvers et Inversion est le plus performant, ce qui confirme l'analyse faite sur la base d'une seule instance dans la phase Réglage.

La **recherche de la meilleure sélection pour le codage direct**, s'appuie sur les tableaux (D.3) et (D.4) de l'annexe D, qui présentent le type de sélection ayant permis d'obtenir les meilleurs valeurs de MEAN et de BEST, pour chaque benchmark et pour chaque combinaison Croisement+Mutation. 99 cas sont comparés (9 combinaisons X 11 *benchmarks*), ce qui conduit aux résultats suivants :

– pour les moyennes, la sélection aléatoire a mieux résolu 60 cas, contre 39 pour la sélection binaire,

– pour les meilleures solutions, la sélection aléatoire est à nouveau plus performante (56 cas) que la sélection binaire (41 cas), malgré 2 cas d'égalité.

Pour le codage direct, la sélection aléatoire semble donc surpasser de manière générale la sélection binaire. Un dernier tableau (tableau 4.13) est construit et analysé pour réunir et compléter les résultats précédents, en considérant les combinaisons Croisement+Mutation et le type de sélection ayant permis d'obtenir les meilleures moyennes et les meilleures solutions pour chaque benchmark. Par exemple, la deuxième ligne de ce tableau contient les informations suivantes, respectivement :

– E-n22-k4: le nom du *benchmark*,

– 392,507 : la meilleure moyenne obtenue pour le benchmark E-n22-k4 en considérant toutes les versions d'algorithme basées sur le codage direct,

– PMX+I: la combinaison ayant permis d'obtenir cette meilleure moyenne,

– Binaire: le type de sélection ayant permis d'obtenir cette meilleure moyenne,

– 375,28 : la meilleure solution obtenue pour le benchmark E-n22-k4 en considérant toutes les versions d'algorithme basées sur le codage direct,

– PMX+I: la combinaison ayant permis d'obtenir cette meilleure solution,

– Aléatoire: le type de sélection ayant permis d'obtenir cette meilleure solution.

Benchmark	Meilleure moyenne			Meilleure solution		
	Valeur	Combinaison	Sélection	Valeur	Combinaison	Sélection
E-n22-k4	**392,507**	PMX+I	binaire	**375,28**	PMX+I	aléatoire
E-n23-k3	**594,13**	PMX+I	aléatoire	**568,56**	PMX+S	aléatoire
E-n30-k4	**548,43**	UX+SI	aléatoire	**509,41**	UX+SI	aléatoire
E-n33-k4	**967,82**	UX+I	aléatoire	**886,727**	PMX+SI	binaire
E-n51-k5	**583,89**	PMX+SI	aléatoire	**536,99**	UX+SI	aléatoire
E-n76-k7	**840,542**	PMX+I	binaire	**748,47**	PMX+I	aléatoire
E-n76-k8	**884,33**	UX+I	binaire	**786,766**	PMX+SI	binaire
E-n76-k10	**962,92**	PMX+I	aléatoire	**946,65**	PMX+I	aléatoire
E-n76-k15	**1209,23**	PMX+SI	binaire	**1156,54**	PMX+SI	binaire
E-n101-k8	**1025,85**	PMX+SI	aléatoire	**927**	PMX+SI	aléatoire
E-n101-k14	**1369,74**	PMX+SI	binaire	**1249,09**	PMX+SI	binaire

Tab. 4.13 – Codage direct, valeurs de B_MEAN et B_BEST, toutes combinaisons de sélection, croisement et mutation confondues

Ce tableau confirme l'analyse précédente des meilleurs opérateurs évolutionnaires pour le codage direct, en présentant un classement par performances décroissantes quasiment identique :

- PMX+SI (4,5),
- PMX+I (4,3),
- UX+SI (1,2),
- UX+I (2,0),
- PMX+S (0,1),
- OX (quelle que soit la mutation), et UX+S (0,0).

D'autre part, sur les 11 problèmes considérés, c'est la sélection aléatoire qui donne le plus souvent la meilleure valeur de moyenne (6 fois) et / ou la meilleure solution (7 fois). Mais le tableau permet de remarquer de plus que la sélection binaire apparaît plus souvent dans la partie inférieure du tableau alors que la sélection aléatoire occupe plus largement la partie supérieure. En conclusion, pour le codage direct,

- la **sélection aléatoire** doit plutôt être préconisée **pour les** plus petites **instances** de VRP (contenant **jusqu'à 50 clients**),
- alors que la **sélection binaire** donne de bons résultats **pour les instances incluant de 50 à 100 clients**.

b. Analyse des résultats du codage indirect

Pour le **codage indirect**, l'ensemble des tableaux D.5 à D.12 est utilisé pour comparer l'efficacité des combinaisons Croisement+Mutation du tableau 4.9 (qui figurent en colonnes). Cette première analyse globale permet de classer ces combinaisons d'opérateurs par ordre de performance décroissant, en fonction du type de sélection et du type de découpage. Ceci fournit les classements suivants :

- avec **sélection aléatoire et méthode de découpage A** [tableaux (D.5) et (D.6)],
  - UX+SI (5,6) et OX +I (4,7),
  - UX+I et UX+S (2,3),
  - OX+S (0,2),
  - OX+SI (0,1),
  - PMX, quel que soit le type de mutation (0,0);
- avec **sélection aléatoire et méthode de découpage B** [tableaux (D.7) et (D.8)],
  - OX +I (6,6),
  - OX+SI (3,5),
  - UX+I (2,5),
  - UX+SI (1,3),
  - PMX+SI, PMX+I (0,2),
  - UX+S (0,1),
  - OX+S et PMX+S (0,0);
- avec **sélection binaire et méthode de découpage A** [tableaux (D.9) et (D.10)]**,**
  - OX +I (13,13),
  - UX+SI et PMX+I (0,2),
  - UX+I, UX+S, OX+SI et PMX+SI (0,1),
  - OX+S et PMX+S (0,0);
- avec **sélection binaire et méthode de découpage B** [tableaux (D.11) et (D.12)]**,**
  - OX +I (11,10),
  - UX+SI (0,6),
  - PMX+I (1,2),
  - UX+I (0,1),
  - OX+S, OX+SI, PMX+S, PMX+SI et UX+S (0,0);

OX+I montre donc une supériorité marquée, pour tous les cas considérés, c'est-à-dire indépendamment du type de sélection et de la méthode de découpage. UX+SI se montre aussi parfois assez performant (voire même aussi performant que OX+I) mais sans que

cela ne semble particulièrement lié à une sélection ou un découpage. En effet, cela se produit pour deux cas totalement opposés : sélection aléatoire et méthode de découpage A d'une part, et sélection binaire et méthode de découpage B d'autre part. Enfin, pour les mutations, swapping se montre à nouveau la moins performante, en apparaissant 8 fois dans le dernier rang de classement avec différents opérateurs de croisement.

La **recherche de la meilleure méthode de découpage**, repose sur la comparaison des tableaux (D.5) à (D.12) de l'annexe D, qui présentent les meilleures valeurs de MEAN et de BEST pour les différentes combinaisons de Croisement + Mutation (en colonnes) et les différents benchmarks (en lignes). Cette analyse fournit les résultats suivants :

- **pour la sélection aléatoire**, sur 117 cas (produit de 9 combinaisons et 13 *benchmarks*) :
  - le découpage **B** fournit la meilleure moyenne dans **72 cas**, alors que cela n'est vrai que pour **45 cas** pour le découpage **A**;
  - le découpage **B** fournit la meilleure solution dans **70** cas, le découpage **A** dans 39 cas et il existe **9** cas d'égalité.
- **pour la sélection binaire**, sur 117 cas:
  - le découpage **B** fournit la meilleure moyenne dans **47** cas contre **70** cas pour le découpage **A**,
  - le découpage **B** fournit la meilleure solution dans **29** cas, contre **81** cas pour le découpage **A.**

Ainsi, il semble que la **méthode de découpage A** soit plus appropriée **pour** être associée à la **sélection binaire**, alors que la **méthode de découpage B** est à conseiller **avec** la **sélection aléatoire**.

La construction des tableaux (4.14 et 4.15) permet, de plus, de faire une observation complémentaire. Ces tableaux présentent la meilleure moyenne et la meilleure solution pour chaque *benchmark* en précisant la combinaison et la méthode de découpage ayant permis d'obtenir ces résultats. Le tableau 4.14 concerne le cas de la solution aléatoire, alors que le tableau 4.15 compare les algorithmes à sélection binaire. Ils confirment que la méthode B est beaucoup mieux représentée dans le cas de la sélection aléatoire, alors que A domine le tableau dans le cas de la sélection binaire. De plus, ils montrent que B se trouve plutôt dans la partie supérieure des tableaux, et A dans la partie inférieure. En conséquence, il semblerait que

- la **méthode de découpage B,** avec sélection aléatoire, doit plutôt être préconisée **pour les** plus petites **instances** de VRP (contenant **jusqu'à 76 clients**),
- alors que la **méthode de découpage A** donne de bons résultats **pour les instances incluant de 76 à 100 clients**.

Benchmark	Meilleure moyenne			Meilleure solution		
	Valeur	Combinaison	Découpage	Valeur	Combinaison	Découpage
E-n23-k3	**569,32**	UX+SI	A	**568,56**	UX+SI	=
E-n30-k3	**548,37**	OX+I	B	**528,7**	UX+I	B
E-n30-k4	**511,51**	UX+SI	B	**505,01**	OX+SI	B
E-n33-k4	**880,38**	UX+I	B	**846,53**	UX+I	B
E-n51-k5	**601,32**	OX+SI	B	**557,13**	OX+SI	B
E-n76-k7	**809,65**	OX+I	B	**761,49**	OX+I	B
E-n76-k8	**958,76**	OX+I	B	**843,24**	UX+I	B
E-n76-k10	**955,56**	OX+I	A	**950,16**	OX+I	B
E-n76-k14	**1425,91**	OX+SI	A	**1113,22**	OX+I	A
E-n76-k15	**1139,63**	OX+I	A	**1093,81**	OX+I	B
E-n101-k8	**1128,5**	OX+SI	B	**981,59**	OX+SI	B
E-n101-k14	**1531,23**	OX+I	B	**1291,43**	UX+SI	A

Tab. 4.14 – Codage indirect avec sélection aléatoire, valeurs de B_MEAN et de B_BEST avec combinaisons Croisement+Mutation et méthodes de découpage correspondantes.

Benchmark	Meilleure moyenne			Meilleure solution		
	Valeur	Combinaison	Découpage	Valeur	Combinaison	Découpage
E-n23-k3	**570,15**	OX+I	B	**568,56**	OX+I	=
E-n30-k3	**548,58**	OX+I	B	**535,8**	OX+I	A
E-n30-k4	**520,43**	PMX+I	B	**505,23**	OX+I	B
E-n33-k4	**896,86**	OX+I	B	**848,85**	OX+I	A
E-n51-k5	**607,7**	OX+I	B	**553,31**	OX+I	A
E-n76-k7	**858,26**	OX+I	B	**743,76**	OX+I	A
E-n76-k8	**841,45**	OX+I	A	**794,11**	OX+I	A
E-n76-k10	**940,41**	OX+I	A	**893,22**	OX+I	A
E-n76-k14	**1136,38**	OX+I	A	**1104,83**	OX+I	A
E-n76-k15	**1149,35**	OX+I	A	**1087,24**	OX+I	A
E-n101-k8	**1155,66**	OX+I	B	**991,19**	OX+I	A
E-n101-k14	**1351,94**	OX+I	A	**1214,64**	OX+I	A

Tab. 4.15 – Codage indirect avec sélection binaire, valeurs de B_MEAN et de B_BEST avec combinaisons Croisement+Mutation et méthodes de découpage correspondantes.

Enfin, la **recherche de la meilleure sélection** pour le codage indirect s'effectue en comparant les résultats du premier groupe de tableaux (D.5) et (D.6) avec les résultats du troisième groupe de tableaux (D.9) et (D.10) et ceux de (D.7) et (D.8) avec ceux de (D.11) et (D.12). Ces tableaux présentent la meilleure moyenne et la meilleure solution

pour chaque *benchmark* en précisant la (ou les) combinaison(s) et la (ou les) sélection(s) ayant permis d'obtenir ces résultats. Leur observation montre que

- avec la méthode de découpage A, sur 117 cas,
  - la **sélection aléatoire** a fourni **74** des meilleures moyennes et la **sélection binaire 43**,
  - la **sélection aléatoire a fourni 47** des meilleures solutions et la **sélection binaire 62**. Il existe également **8 cas d'égalité**.
- avec la méthode de découpage B, sur 108 cas (produit de 9 combinaisons par 12 *benchmarks*),
  - la **sélection aléatoire** a fourni **91** des meilleures moyennes et la **sélection binaire 17**,
  - la **sélection aléatoire a fourni 81** des meilleures solutions et la **sélection binaire 18**. Il existe également **9 cas d'égalité**.

Les tableaux (4.16) et (4.17) agrègent ces résultats pour montrer la meilleure moyenne et la meilleure solution pour chaque *benchmark*. Ils précisent quelle combinaison Croisement+Mutation et quel type de sélection ont conduit à ces résultats. Le tableau (4.16) compare les programmes qui utilisent la méthode A comme méthode de découpage, le tableau (4.17) ceux qui utilisent la méthode B.

Benchmark	Meilleure moyenne			Meilleure solution		
	Valeur	Combinaison	Sélection	Valeur	Combinaison	Sélection
E-n22-k4	**380,52**	OX+I	aléatoire	**375,28**	OX(I+SI)/UX+I	=
E-n23-k3	**569,32**	UX+SI	aléatoire	**568,56**	OX+I/UX(S+SI)	=
E-n30-k3	**571,73**	UX+SI	aléatoire	**535,8**	OX+I	binaire
E-n30-k4	**521,43**	OX+I	binaire	**505,01**	PMX+I	binaire
E-n33-k4	**912,78**	UX+SI	aléatoire	**848,85**	OX+I	binaire
E-n51-k5	**677,73**	OX+I	binaire	**553,31**	OX+I	binaire
E-n76-k7	**884,89**	UX+SI	aléatoire	**743,76**	OX+I	binaire
E-n76-k8	**841,45**	OX+I	binaire	**794,11**	OX+I	binaire
E-n76-k10	**940,41**	OX+I	binaire	**893,22**	OX+I	binaire
E-n76-k14	**1136,38**	OX+I	binaire	**1104,83**	OX+I	binaire
E-n76-k15	**1139,63**	OX+I	aléatoire	**1087,24**	OX+I	binaire
E-n101-k8	**1233,02**	UX+SI	aléatoire	**991,19**	OX+I	binaire
E-n101-k14	**1351,94**	OX+I	binaire	**1214,64**	OX+I	binaire

Tab. 4.16 – Codage indirect avec méthode de découpage A, valeurs de B_MEAN et B_BEST, avec combinaisons Croisement+Mutation et types de sélection correspondants

Benchmark	Meilleure moyenne	Combinaison de meilleure moyenne	Sélection de meilleure moyenne	Meilleure solution	Combinaison de meilleure solution	Sélection de meilleure solution
E-n22-k4	**379,52**	OX+I	binaire	**375,28**	(OX,PMX)+I/UX+SI	=
E-n23-k3	**570,15**	OX+I	binaire	**568,56**	(OX,PMX,UX)+I/UX+SI	=
E-n30-k3	**548,37**	OX+I	aléatoire	**528,7**	UX+I	aléatoire
E-n30-k4	**511,51**	UX+SI	aléatoire	**505,01**	OX+SI	aléatoire
E-n33-k4	**880,38**	UX+I	aléatoire	**846,53**	UX+I	aléatoire
E-n51-k5	**601,32**	OX+SI	aléatoire	**557,13**	OX+SI	aléatoire
E-n76-k7	**809,65**	OX+I	aléatoire	**761,49**	OX+I	aléatoire
E-n76-k8	**958,76**	OX+I	aléatoire	**843,24**	UX+I	aléatoire
E-n76-k10	**1064,67**	UX+I	aléatoire	**950,16**	OX+I	aléatoire
E-n76-k15	**1148,36**	OX+I	aléatoire	**1093,81**	OX+I	aléatoire
E-n101-k8	**1128,5**	OX+SI	aléatoire	**981,59**	OX+SI	aléatoire
E-n101-k14	**1531,23**	OX+I	aléatoire	**1362,69**	OX+I	aléatoire

Tab. 4.17 – Codage indirect avec méthode de découpage B, valeurs de B_MEAN et B_BEST, avec combinaisons Croisement+Mutation et types de sélection correspondants

Dans ces deux tableaux, la sélection binaire semble meilleure que la sélection aléatoire avec la méthode de découpage A, alors que c'est l'inverse pour la méthode de découpage B. Mais globalement, la sélection aléatoire semble dominer.

Finalement, pour compléter et vérifier les conclusions de cette section, tous les résultats obtenus pour le codage indirect ont été comparés.

La comparaison de tous les résultats du codage indirect utilise le tableau 4.18, qui présente les valeurs de B_MEAN et B_BEST pour les 13 benchmarks étudiés, et les différents opérateurs (sélection, combinaisons Croisement+Mutation, méthode de découpage) associés.

Ce tableau permet de classer, pour toutes les versions d'algorithme basées sur le codage indirect, les combinaisons d'opérateurs de croisement et de mutation par ordre de performance décroissant. Le classement, qui indique leurs scores respectifs (nombre de meilleures moyennes fournies, nombre de meilleures solutions fournies), est le suivant :

- OX+I (8,11),
- UX+SI et OX+SI (2,3), et UX+I (1,4),
- PMX+I (0,3),
- UX+S (0,1),
- OX+S, PMX+SI, PMX+S (0,0).

Ainsi, **OX montre une supériorité marquée**, en particulier lorsqu'il est associé **avec la mutation Inversion**, et **surtout dans la partie inférieure du tableau, mais pas**

Benchmark	**Meilleure moyenne**				**Meilleure solution**			
	Valeur	Combinaison	Sélection	Découpage	Valeur	Combinaison	Sélection	Découpage
E-n22-k4	**379,52**	OX+I	binaire	B	**375,28**	(OX, PMX)+I ou UX+SI	binaire/aléatoire	B
						OX+(I,SI) ou UX+I	binaire/aléatoire	A
E-n23-k3	**569,32**	UX+SI	aléatoire	A	**568,56**	(OX, PMX, UX)+I ou UX+SI	binaire/aléatoire	B
						OX+I ou UX+(S,SI)	binaire/aléatoire	A
E-n30-k3	**548,37**	OX+I	aléatoire	B	**528,7**	UX+I	aléatoire	B
E-n30-k4	**511,51**	UX+SI	aléatoire	B	**505,01**	OX+SI	aléatoire	B
						PMX+I	binaire	A
E-n33-k4	**880,38**	UX+I	aléatoire	B	**846,53**	UX+I	aléatoire	B
E-n51-k5	**601,32**	OX+SI	aléatoire	B	**553,31**	OX+I	binaire	A
E-n76-k7	**809,65**	OX+I	aléatoire	B	**743,76**	OX+I	binaire	A
E-n76-k8	**841,45**	OX+I	binaire	A	**794,11**	OX+I	binaire	A
E-n76-k10	**940,41**	OX+I	binaire	A	**893,22**	OX+I	binaire	A
E-n76-k14	**1136,38**	OX+I	binaire	A	**1104,83**	OX+I	binaire	A
E-n76-k15	**1139,63**	OX+I	aléatoire	A	**1087,24**	OX+I	binaire	A
E-n101-k8	**1128,5**	OX+SI	aléatoire	B	**981,59**	OX+SI	aléatoire	B
E-n101-k14	**1351,94**	OX+I	binaire	A	**1214,64**	OX+I	binaire	A

Tab. 4.18 – Codage indirect, valeurs de B_MEAN et B_BEST, avec combinaisons Croisement+Mutation, méthodes de découpage et types de sélection associés

uniquement. Le croisement **UX**, sauf quand il est associé à la mutation Swapping, est assez **performant également mais plutôt pour les instances comportant moins de 50 clients**, ce qui explique qu'il présente des scores inférieurs à OX. Le croisement PMX et la mutation de swapping quant à eux, sont presque relégués systématiquement dans le dernier rang du classement à quelques exceptions près.

Pour la méthode de découpage des solutions représentées par codage indirect, la méthode **B**, qui prend pourtant en compte plus d'informations (distance et capacité) ne domine que la partie supérieure du tableau pour les meilleures moyennes (jusqu'au problème E-n76-k7) et la plupart des meilleures solutions de cette partie. Le plus souvent, elle est **associée à la sélection aléatoire**. La méthode **A** quant à elle, est **plus souvent associée à la sélection binaire** et domine largement la partie inférieure du tableau (pour tous les résultats à l'exception d'une seule instance de problème E-n101-k8). Ainsi, la combinaison **sélection aléatoire + découpage B** est plutôt à préconiser pour les **instances allant jusqu'à une cinquantaine de clients**, alors que la combinaison **sélection binaire + découpage A** peut être conseillée pour les **instances plus grandes**.

Mais l'ensemble de ces conclusions reste propre aux versions d'algorithme basées sur le codage indirect. Il reste à ré-intégrer l'ensemble des facteurs considérés dans l'étude (codage, découpage, sélection, croisement, mutation) pour comparer tous les résultats globalement. C'est l'objet du paragraphe et des tableaux suivants.

c. Comparaison des résultats en considérant l'ensemble des facteurs :

Jusqu'à présent, des sous-ensembles de résultats ont été examinés séparément, pour effectuer l'analyse sous différents angles et détecter d'éventuelles relations entre les différents facteurs. Ce paragraphe présente les meilleurs résultats obtenus pour le problème de tournées à contrainte de capacité, en comparant les résultats obtenus par l'ensemble des 54 versions d'algorithmes définies en fin de chapitre 3.

Benchmark	Meilleure moyenne B_MEAN				
	Valeur	Combinaison	Sélection	Découpage	Codage
E-n22-k4	379,52	OX+I	binaire	B	indirect
E-n23-k3	569,32	UX+SI	aléatoire	A	indirect
E-n30-k3	548,37	OX+I	aléatoire	B	indirect
E-n30-k4	511,51	UX+SI	aléatoire	B	indirect
E-n33-k4	880,38	UX+I	aléatoire	B	indirect
E-n51-k5	583,89	PMX+SI	aléatoire	_	direct
E-n76-k7	809,65	OX+I	aléatoire	B	indirect
E-n76-k8	841,45	OX+I	binaire	A	indirect
E-n76-k10	940,41	OX+I	binaire	A	indirect
E-n76-k14	1136,38	OX+I	binaire	A	indirect
E-n76-k15	1139,63	OX+I	aléatoire	A	indirect
E-n101-k8	1025,85	PMX+SI	aléatoire	_	direct
E-n101-k14	1351,94	OX+I	binaire	A	indirect

Tab. 4.19 – Caractéristiques des versions d'algorithme ayant fourni les meilleurs moyennes B_MEAN pour chacun des 13 benchmarks étudiés

Cette étape est la plus importante pour déterminer les versions d'algorithmes les plus appropriées pour résoudre la variante de problèmes de tournées avec capacité considérée. Cette analyse globale s'appuie sur deux tableaux 4.19 et 4.20. Ces derniers présentent les valeurs de B_MEAN et B_BEST pour chacun des 13 benchmarks sur lesquels portait l'étude (en lignes), en précisant les caractéristiques (codage, découpage, sélection et croisement+mutation) de la (ou des) version(s) d'algorithme ayant permis de les déterminer.

Benchmark	**Meilleure solution**	**Combinaison de meilleure solution**	**Sélection de meilleure solution**	**Découpage de meilleure solution**	**Codage de meilleure solution**
E-n22-k4	375,28	(OX,PMX)+I ou UX+SI	binaire/aléatoire	B	indirect
		OX+(I,SI) ou UX+I	binaire/aléatoire	A	indirect
		PMX+I	aléatoire	_	direct
E-n23-k3	568,56	(OX, PMX, UX)+I ou UX+SI	binaire/aléatoire	B	indirect
		OX+I ou UX+(S,SI)	binaire/aléatoire	A	indirect
		PMX+S	aléatoire	_	direct
E-n30-k3	528,7	UX+I	aléatoire	B	indirect
E-n30-k4	505,01	OX+SI	aléatoire	B	indirect
		PMX+I	binaire	A	indirect
E-n33-k4	846,53	UX+I	aléatoire	B	indirect
E-n51-k5	536,99	UX+SI	aléatoire	_	direct
E-n76-k7	743,76	OX+I	binaire	A	indirect
E-n76-k8	786,766	PMX+SI	binaire	_	direct
E-n76-k10	893,22	OX+I	binaire	A	indirect
E-n76-k14	1104,83	OX+I	binaire	A	indirect
E-n76-k15	1087,24	OX+I	binaire	A	indirect
E-n101-k8	927	PMX+SI	aléatoire	_	direct
E-n101-k14	1214,64	OX+I	binaire	A	indirect

Tab. 4.20 – Caractéristiques des versions d'algorithme ayant fourni les meilleures solutions B_BEST pour chacun des 13 benchmarks étudiés

L'observation de ces tableaux conduits aux principales conclusions suivantes :
Premièrement, pour les deux plus petites instances (22 et 23 clients), quasiment tous les niveaux des facteurs apparaissent dans au moins une combinaison permettant de déterminer la meilleure solution, à part la mutation Swapping qui n'apparait dans aucune combinaison pour la première instance (E-n22-k4). Toutefois, plus globalement

- **le codage indirect domine largement le codage direct** (11 moyennes et 13 meilleures solutions contre 2 moyennes et 5 meilleures solutions, dont 2 également déterminées avec le codage indirect);
- lorsque le **codage direct** apparaît, il est majoritairement associé au croisement **PMX** (6 cas sur 7), plutôt avec la **mutation SwapInvers** (4 cas sur 7), et la **sélection aléatoire** (6 cas sur 7);
- la **méthode de découpage B** est essentiellement performante **avec** la **sélection aléatoire**, et domine majoritairement la partie supérieure des tableaux, c'est-à-dire celle associée aux **instances de petite taille (jusqu'à 76 clients)**;
- la **méthode de découpage A** est essentiellement performante **avec** la **sélection binaire**, et domine majoritairement la partie inferieure des tableaux, c'est-à-dire celle associée aux **instances de grande taille (de 76 à 100 clients)**;
- le croisement **OX** se montre **de loin le plus performant** (8 meilleures moyennes sur 13 et 11 meilleures solutions, dont l'une deux fois avec des mutations différents : Inversion et SwapInvers), et ce **toujours en association avec le codage indirect,** et **le plus souvent** avec la sélection binaire (14 cas) et la mutation Inversion (18 cas). Il ne semble **pas particulièrement sensible au choix de la méthode de découpage**. Il apparaît 13 fois avec la méthode de découpage A et 6 fois avec la méthode de découpage B, mais cela est sans doute lié au fait que le découpage B est meilleur pour les petites instances et le découpage A meilleur pour les grandes instances.
- Le croisement **UX** présente **quelques bons résultats** mais pour les instances les plus petites **(jusqu'à 50 clients)**, **toujours avec une sélection aléatoire** des parents, et **majoritairement avec les mutations Inversion et SwapInvers**.

En résumé, trois versions d'algorithme peuvent être plus particulièrement conseillées pour résoudre la variante de problèmes de tournées avec capacité considérée. Ces versions ont les caractéristiques suivantes :

- AE1 : codage direct, sélection aléatoire, croisement PMX, et mutation SwapInvers,
- AE2 : codage indirect, avec méthode de découpage B, sélection aléatoire, croisement OX et mutation Inversion, à conseiller parfois pour de petites instances,
- AE3 : codage indirect, avec méthode de découpage A, sélection binaire, croisement OX et mutation Inversion, conseillée plutôt pour les plus grandes instances.

Néanmoins, globalement, l'algorithme AE3 dispose d'une supériorité certaine, tant en terme de performances que de robustesse (à la taille des instances et à la variation éventuelle de la sélection et de la méthode de découpage).

4.6 Conclusion

Ce chapitre a présenté les résultats obtenus par les 54 approches évolutionnaires (définies à la fin du chapitre précédent) pour résoudre le problème de tournées de véhicules avec contrainte de capacité. L'objectif était de vérifier s'il est possible d'extraire des règles pour guider la conception d'un algorithme en comparant les performances de différentes combinaisons d'éléments évolutionnaires.

Les performances d'un sous-ensemble d'éléments évolutionnaires (défini dans la section 3.3) ont donc été comparées en les appliquant à 13 benchmarks de (Christofides et Eilon, 1969). Cet ensemble comportait 3 types de codage (en considérant la méthode de découpage pour le codage indirect), 2 types de sélection, 3 types de croisement, 3 types de mutation, le taux de croisement et le nombre de générations. Ces paramètres ont constitués les 6 facteurs à faire varier au cours des tests, dans une démarche basée sur les plans d'expérience, en s'appuyant sur un logiciel de statistiques nommé JMP pour l'analyse.

Etant donné le nombre de facteurs et de niveaux, deux phases préparatoires ont été effectuées en considérant dans un premier temps une seule instance de problèmes à 100 clients (E-n101-k8). Elles ont servi à

- examiner l'influence des différents facteurs ou de combinaisons de ces facteurs, en tenant compte de leurs interactions,
- déterminer les taux de croisement et nombre de générations à utiliser dans la suite des expériences pour les deux types de codage.
- Elles ont permis de
- montrer que le taux de croisement et le nombre de générations étaient des facteurs beaucoup moins significatifs que les autres. Autrement dit, elles ont confirmés qu'un mauvais choix du type d'opérateur à utiliser, par exemple, ne peut être compensé par un réglage optimisé de ces paramètres;
- déterminer 18 combinaisons d'opérateurs (Croisement + Mutation), soit 9 pour le codage direct et 9 pour le codage indirect, chacune avec taux de croisement et nombre de générations fixés.

Finalement, sur la base de ces 18 combinaisons, les quatre facteurs restants (codage, en incluant le type de découpage, type de sélection et type de croisement et de mutation) ont été utilisés pour la phase d'expérimentation finale. Dans celle-ci, les 13 benchmarks de (Christofides et Eilon, 1969) ont été utilisés pour les tests. Chaque version d'algorithme a été appliquée dix fois à chacun d'entre eux. Puis les résultats moyens par algorithme et par benchmark, et les meilleures solutions fournies ont été examinés.

Analyser ces résultats sous différents angles de comparaison a permis notamment d'identifier des relations entre

- type de codage et types de croisement : par exemple, avec le codage direct, PMX est le croisement le plus efficace, alors qu'avec le codage indirect il s'agit d'OX,

- type de croisement et type de mutation : par exemple, dans le cas du codage indirect, la mutation Swapping est très peu performante avec OX et PMX, mais permet de surpasser PMX lorsqu'elle est associée à UX,
- type de sélection et méthode de découpage : la méthode de découpage A s'associe plutôt bien avec la sélection aléatoire, alors que pour la méthode de découpage B les associations avec la sélection binaire semblent plus souvent efficaces,
- type d'éléments évolutionnaires et taille de problème à résoudre : par exemple, en codage direct, la sélection aléatoire peut être conseillée pour des instances contenant jusqu'à 50 clients. Pour des problèmes plus grands, la sélection binaire semble plus appropriée.

Puis finalement, une comparaison globale des performances des 54 versions d'algorithme évolutionnaire a permis de mettre en évidence

- la supériorité de certains types d'éléments évolutionnaires pour la variante considérée : par exemple le codage indirect est plus efficace que le codage direct,
- de déterminer des combinaisons d'éléments plus particulièrement efficaces : ainsi 3 versions d'algorithme sur 54 ont été identifiées comme étant les plus pertinentes pour cette variante.

Mais ces résultats ne sont pas à considérer dans l'absolu. D'autres éléments évolutionnaires de la littérature, qui n'ont pas été pris en compte dans cette étude, pourraient très bien les modifier. En particulier, il existe de nombreuses approches permettant d'obtenir de meilleures performances que celles obtenues ici, mais avec lesquelles aucune compétition n'était recherchée. La comparaison se faisait ici entre versions d'algorithmes présentant une base commune très simple, mais elle a permis d'atteindre les objectifs fixés, autrement dit :

- prouver qu'il existe des interdépendances parfois très fortes entre les choix effectués lors de la conception d'un algorithme,
- montrer qu'il est possible par expérimentation de détecter ces relations et d'identifier les combinaisons de choix les plus efficaces pour une variante de problème donnée.

Ceci prouve la faisabilité du projet de construction d'une base de règles guidant la construction d'un algorithme, par exemple

- en proposant au concepteur un sous ensemble d'éléments évolutionnaires ou de combinaisons de ces éléments qui semblent plus pertinents pour la variante qu'il cherche à résoudre,
- en réduisant le nombre de choix proposés au fur et à mesure que le concepteur prend des décisions (par exemple, lui proposer les opérateurs de croisement les plus pertinents à tester en fonction du type de codage qu'il utilise).

Pour clore l'illustration de l'usage de la notation dans ce domaine, il resterait à renouveler cette démarche expérimentale pour une autre variante du problème (par exemple en ajoutant à celle considérée ici une contrainte de fenêtres temporelles), puis à comparer les résultats obtenus pour les deux variantes pour déterminer l'impact qu'a l'ajout de cette contrainte.

Conclusion

Ce mémoire de thèse a présenté, illustré et validé la faisabilité d'une méthodologie pour aider à la conception d'algorithmes évolutionnaires pour la résolution des problèmes de tournées (ou VRP). Outre la méthodologie et les résultats expérimentaux issus de son application, une des principales contributions de ce travail est la proposition d'une classification permettant d'identifier de façon univoque les problèmes de tournées de véhicule.

Ce travail montre également, en analysant de nombreux exemples, qu'il existe une corrélation entre la nature du problème à optimiser et la qualité des résultats de l'optimisation en fonction des opérateurs génétiques utilisés. Il prouve ainsi la nécessité d'étudier les relations entre les problèmes et les méthodes de résolution. Ce travail ouvre la voix à de nombreux autres travaux tout en fournissant la base méthodologique.

Le chapitre 1 a rappelé la diversité des applications dans lesquelles ce type de problème doit être résolu en planification industrielle : certaines évidentes, comme le transport de personnes et de marchandises, d'autres moins intuitives, comme l'optimisation de la fabrication de bouteilles ou de polymères. Il a ainsi souligné les enjeux importants qui reposent sur une résolution optimale de différentes variantes de VRP. Il a montré que cela a suscité beaucoup d'intérêt de la part des chercheurs, et qu'en conséquence la littérature est très riche, tant en termes de variantes de problème étudiées que de méthodes de résolution proposées. Ainsi, en présence d'un problème de tournées à résoudre, ce contexte complique à la fois l'étude bibliographique et le choix d'une méthode de résolution appropriée.

Le chapitre 2 a proposé une notation des problèmes de tournées palliant les inconvénients du système d'identification actuel, qui repose sur des sigles. Pour faciliter son utilisation, elle repose sur un schéma du type $\pi/\alpha/\beta/\gamma$ assez classique, utilise des règles de construction systématiques et des symboles correspondant autant que possible à d'anciens sigles ou à des moyens mnémotechniques. La suite du chapitre 2 a illustré comment elle facilite l'étude bibliographique, tant du point de vue des problèmes traités que des méthodes de résolution. Elle permet par exemple

- d'identifier sans ambiguïté les variantes de problèmes de tournées, et donc de les comparer de manière plus fiable,
- de classifier les variantes de problèmes, et de repérer les variantes les plus étudiées (ou au contraire les pistes encore inexplorées), ainsi que l'évolution de ces classes et des travaux qui leur sont consacrés au cours du temps.

Les résultats d'un nombre restreint de références (dont la notation est suffisamment similaire, et qui utilisent toutes des approches évolutionnaires) a également été

effectuée. Mais elle n'a pas permis d'analyser finement les performances des différents opérateurs composant ces algorithmes.

Ainsi, le chapitre 2 a prouvé que la notation générait des apports indéniables pour répondre au premier objectif fixé. Son utilisation, et celle de la classification qu'elle permet de générer, facilite l'étude bibliographique de la vaste littérature consacrée au VRP. Mais il en a également souligné trois limites majeures. Tout d'abord, la notation doit être confrontée à un nombre plus important de références pour étoffer l'effectif de l'échantillon sur lequel portent les analyses. Ce travail est nécessaire pour rendre la classification et son analyse significatives. Ce mémoire ne représente qu'une illustration de la démarche proposée, en quelque sorte une étude de faisabilité. Deuxièmement, la notation est difficile à utiliser en se contentant d'un formalisme sur papier. En effet, lors de sa conception nous avons tenté de limiter sa composition à un sous-ensemble de paramètres pertinents le plus limité possible. Des valeurs par défaut permettant de représenter les variantes les plus simples de manière concise ont également été définies. Mais la diversité des cas rencontrés dans la littérature ne cesse de s'accroître. De plus, une seule caractéristique (une contrainte par exemple) peut avoir un impact important sur l'espace de recherche du problème et donc des conséquences sur le choix de la méthode de résolution. Il est donc difficile de trouver des caractéristiques pouvant être omises dans la notation. De ce fait, elle comporte beaucoup de paramètres et peut sembler lourde à mettre en œuvre. D'autre part, l'analyse par comparaison des notations d'articles est difficile dans un tableau statique regroupant de nombreuses références. Il est difficile d'y trouver une information pertinente, et impossible de trier les références selon un certain angle de comparaison. Finalement, le chapitre 2 a aussi conclu que la comparaison des approches en termes de performances semblait beaucoup plus difficilement réalisable si elle ne se basait que sur les résultats de la littérature. Or cette analyse est celle qui permettrait à un utilisateur de déterminer quel type de méthode ou quels opérateurs semblent plus efficaces pour résoudre la variante de problème qu'il considère. Pour répondre à ce second objectif, une démarche expérimentale a donc été proposée et illustrée dans les deux derniers chapitres du mémoire, en considérant le cas des opérateurs évolutionnaires.

Le chapitre 3 a tout d'abord présenté le projet dans sa globalité. Celui-ci consiste à associer la notation des problèmes de VRP à une base d'éléments évolutionnaires puis à construire une base de règles liant les deux. L'objectif est de permettre à un utilisateur de décrire le problème à résoudre, puis d'en générer la notation et d'en déduire finalement quels opérateurs ont la plus forte probabilité d'être efficaces. Ceci permet de le guider dans sa recherche d'un algorithme approprié.

La définition de la base d'éléments évolutionnaires a ensuite été illustrée. Cette section a recensé, tout en rappelant rapidement les principes généraux des algorithmes évolutionnaires, différents éléments que pourrait contenir une telle base. Le chapitre s'est plus spécifiquement intéressé aux types de croisement présents dans la littérature. Finalement, la dernière section du chapitre 3 a sélectionné un sous-ensemble d'éléments (parmi ceux susceptibles d'appartenir à cette base), en vue de réaliser la phase expérimentale.

En effet, cette dernière, présentée dans le chapitre 4, est à nouveau une illustration de la démarche proposée. C'est en quelque sorte une étude de faisabilité de la dernière phase du projet : la construction de la base de règles. Un sous-ensemble restreint

d'éléments permettait donc de démontrer la faisabilité de la démarche tout en limitant la durée de la phase d'expérimentation pour pouvoir présenter des résultats rapidement. Trois types de croisement et trois types de mutation ont donc été sélectionnés pour comparer expérimentalement l'efficacité des 9 combinaisons croisement+mutation correspondantes. Un schéma général d'algorithme très simple dans lequel intégrer, pour chaque série de tests, une combinaison croisement+mutation différente a également été défini. Trois types de codage et deux types de sélection ont également été retenus. Le but était d'estimer la sensibilité des croisements et mutations, dans le premier cas, au choix du codage (souvent réalisé avant le choix des opérateurs), et dans le second cas à l'importance de la pression sélective de l'algorithme. Cela a donc permis de définir 54 versions d'algorithme évolutionnaire différentes dont le chapitre 4 a comparé les performances.

Pour cette comparaison, la variante de VRP choisie était une variante très simple avec juste la contrainte de capacité. 13 benchmarks de (Christophides et Eilon, 1969) ont été utilisés.

Le chapitre 4 a tout d'abord présenté la méthodologie proposée, basée sur les plans d'expérience et un logiciel de statistiques. Deux phases d'expérience ont ensuite permis de fixer les valeurs de taux de croisement et de nombre de générations pour chacune des 9 combinaisons Croisement+Mutation, avec l'un des codages indirects (avec découpage basé sur la capacité) et avec le codage direct. Puis les tests ont été effectués sur cette base en faisant varier de plus le type de codage, et de sélection. Cette étude de faisabilité a été concluante elle a permis de vérifier qu'il est possible

- d'estimer quels sont les éléments significatifs de l'algorithme (c'est-à-dire ayant un impact détectable sur les performances),
- d'établir des relations éventuelles entre différents choix de conception de l'algorithme (par exemple un type de codage et un type de croisement), ou entre l'instance de problème et l'efficacité des opérateurs.

Ainsi, différents types de relations ont été identifiés. En particulier, avec le codage direct le croisement le plus performant est PMX, alors que pour les codages indirects il s'agit de OX. Certaine relations tiennent compte en outre de la taille de problème considéré. Finalement, trois versions d'algorithme ont pu être sélectionnées pour les proposer à un utilisateur ayant un VRP avec capacité à résoudre. Mais naturellement ces résultats ne sont pas à considérer dans l'absolu. Il existe des approches de résolution dans la littérature donnant de meilleurs résultats. En effet, le but n'était pas ici d'entrer en compétition avec ces dernières, mais de vérifier la faisabilité de la démarche. Dès lors l'algorithme de base proposé est basique.

Il existe trois principales directions dans lesquelles ces travaux doivent être poursuivis :

Premièrement, la notation doit être encore confrontée à de nombreux articles de la littérature

- pour la valider définitivement, c'est-à-dire vérifier si elle est composée de paramètres tous pertinents et si elle permet de représenter tous les cas rencontrés,

- pour que le nombre d'articles ainsi référencé soit suffisant pour obtenir une classification des problèmes significative et intéressante.

Deuxièmement, la notation est sans doute vouée à l'oubli si les chercheurs doivent l'appliquer sous la même forme que celle employée ici pour faire la démonstration de son utilité. L'étude détaillée d'un article pour identifier clairement toutes ses caractéristiques représente un effort non négligeable.

Traduire ensuite le résultat de ce travail dans la notation standard proposée représente encore un effort supplémentaire. Renouveler cette démarche pour tous les articles de la littérature, ou à défaut pour une majorité d'entre eux, demande donc un effort humain considérable et plutôt dissuasif.

C'est pourquoi l'une des perspectives de ce travail est le développement d'un portail basé sur

- une interface qui permettrait à l'utilisateur de décrire le problème étudié de manière simple, sous forme de questions réponses, sans avoir à connaître les paramètres de la notation,
- un traducteur générant la notation associée à la description ainsi saisie,
- une base de données dont la structure serait conforme à la notation, pour stocker les notations ainsi construites, et faciliter la recherche d'articles et l'analyse de la littérature (en utilisant des requêtes traduisant divers critères de recherche).

De plus, si ce portail est collaboratif cela permettrait de répartir l'effort nécessaire à la saisie dans la base, et de mettre cet outil à la disposition des chercheurs. Par exemple, on peut imaginer que chaque équipe intéressée y référence elle-même au minimum ses propres publications, ce qui éviterait les erreurs et augmenterait plus vite le nombre de références enregistrées. Par ce biais, chaque équipe pourrait bénéficier en contre-partie de cet outil pour ses études bibliographiques et également favoriser la citation de ses travaux.

De même, la base d'éléments évolutionnaires doit être mise en place et son contenu doit être complété en continuant à recenser les mécanismes évolutionnaires décrits dans la littérature.

Enfin, pour la construction de la base de règles, l'étape suivante consiste à renouveler la phase d'expérimentation du chapitre 4 avec d'autres variantes de VRP légèrement différentes, puis à comparer les jeux de résultats pour voir s'il existe une relation entre variante de problème (autrement dit apparition de certains paramètres dans la notation du problème) et efficacité des opérateurs.

Annexes

Annexe A

La logistique et la planification industrielle

Le terme «logistique» vient du mot français (loger). Historiquement, ce mot a trouvé son origine dans deux contextes[14]: les mathématiques où le mot logistique était équivalent au raisonnement, et le contexte militaire, où la logistique désignait un ensemble d'opérations physiques liées à l'approvisionnement et au transport des armes, des munitions et de la nourriture. Le dictionnaire anglais "*Oxford*" définit la logistique en la remettant dans son contexte historique militaire : "une branche de la science militaire ayant comme but l'obtention, le maintien et le transport des matières, du personnel et des équipements.

Cette définition historique n'est nullement obsolète aujourd'hui mais, plus récemment, une autre définition est utilisée. Celle-ci résulte de la plus grande attention portée aux méthodes sur lesquelles s'appuie la gestion des opérations. Dès lors, la logistique est devenue une branche des sciences du management, appréhendant des activités spécifiques d'une entreprise industrielle et commerciale, ayant axées à la gestion des flux physiques et des flux d'information, depuis le fournisseur initial jusqu'au client final, dans l'objectif de répondre à des exigences de qualité, de délais et de coût. Pour réaliser cet objectif, il est nécessaire d'optimiser l'affectation des ressources, des capacités de production, des stocks et des capacités de transport. De nos jours, la logistique est devenue plus conceptuelle, et très liée à la chaîne logistique (ou *supply chain*). Dans ce qui suit, nous définissons la chaîne logistique, les notions liées à sa gestion et à sa planification, et nous nous attarderons beaucoup plus sur le management des ressources de distribution et sur le transport.

A.1 La chaîne logistique

Pour plus de clarté, nous présenterons dans cette section d'abords quelques définitions de la chaîne logistique trouvées dans la littérature. Ensuite, nous citerons les différents composants de cette chaîne, et enfin nous terminerons par les différentes catégories de flux.

A.1.1 Définitions

De nombreuses définitions ont été proposées dans la littérature pour la chaîne logistique. Une définition simplifiée peut être trouvée dans (Poirier et *al.*, 2001), qui compare la chaîne logistique à un système grâce auquel les entreprises amènent leurs produits et leurs services jusqu'à leurs clients. (Rota *et al.*, 2001) définit la chaîne logistique comme un ensemble d'entreprises qui interviennent dans les processus de fabrication, de distribution et de vente du produit, du premier fournisseur au client final.

[14] http://fr.wikipedia.org/wiki/Histoire_de_la_logistique

Une définition plus opérationnelle de la chaîne logistique a été donnée par (Lee *et al*., 1993), qui représente celle-ci par un réseau d'installations ayant pour but d'assurer les fonctions d'approvisionnement en matières premières, de transformation de ces matières premières en composants puis en produits finis, et de distribution des produit finis vers les clients.

Certains auteurs, comme (Eymery, 1998), parlent de chaîne logistique interne pour distinguer l'ensemble des flux de produits internes à l'entreprise lié aux opérations d'approvisionnement, de transformation, de stockage, de manutention et de transport.

Plus récemment, (Génin, 2003) définit la chaîne logistique comme un réseau d'organisations ou de fonctions géographiquement dispersés sur plusieurs sites, qui coopèrent pour réduire les coûts, améliorer la qualité et diminuer les délais de réponses des processus et des activités entre les fournisseurs et les clients. D'après (Giard et Mendy, 2006), la chaîne logistique est un enchaînement de processus de production ou de transport, liés par des relations de type (client-fournisseur) orientés vers la satisfaction de la demande des clients.

Enfin, la définition la plus utilisée est celle donnée par (Tayur *et al*., 1999), qui décrit la chaîne logistique comme un système de sous-traitants, de producteurs, de distributeurs, de détaillants et de clients ; entre lesquels s'échangent des flux physiques qui circulent des fournisseurs vers les clients et des flux d'information qui circulent des fournisseurs vers les clients et des clients vers les fournisseurs.

A.1.2 Les composants de la chaîne logistique

Nous pourrons déterminer les composants principaux de la chaîne logistique comme suit, figure (A.1) :

- Les fournisseurs : c'est la source capable d'apporter les éléments de base indispensables à la construction d'une chaîne logistique comme les matières premières, les fournitures, les produits de base, etc.
- Les producteurs : c'est le maillon de la chaîne qui fabrique, assemble, transforme ou fournit un produit.
- Les distributeurs : c'est les différents points de vente qui reçoivent au moment où ils ont besoin des quantités appropriées de produits finis. Dans certains réseaux, un grossiste est intégré à la chaîne. Il achète une quantité importante de produits pour livrer ensuite les petits commerçants (détaillants). Les grossistes ont généralement leurs propres entrepôts mais ils peuvent aussi utiliser d'autres sites de stockage pour faire transiter les produits avant de les livrer à leurs clients.
- Les consommateurs : ce maillon comprend aussi bien le commerce de proximité que les grands magasins, les discounters, les clubs d'achat, les super et hypermarchés ; où est effectué l'achat final.

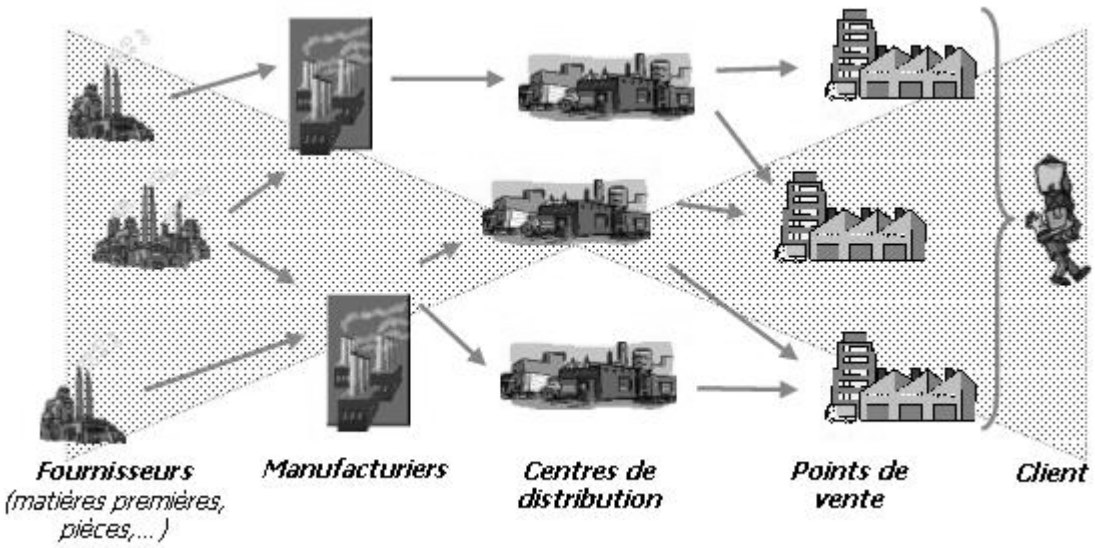


Fig. A.1 – Les composants d'une chaîne logistique

A.1.3 Les catégories de flux

Dans une chaîne logistique, il existe trois catégories de flux :

- les flux physiques (ou flux de marchandises) : à titre d'exemple, nous pourrons citer les matières premières, les produits finis, les pièces détachées, les emballages et les produits à recycler. En général, les flux circulent de l'amont vers l'aval, mais quelques fois, ils peuvent circuler dans le sens inverse pour diverses raisons, comme par exemple des produits défectueux ou qui ne répondent pas au besoin du client. Ce type de processus où les flux circulent de l'aval à l'amont est appelé logistique inverse. Il veille au contrôle des matières premières, des encours de fabrication, des produits finis et de l'information relatifs à ces flux inverses, voir figure (A.2).

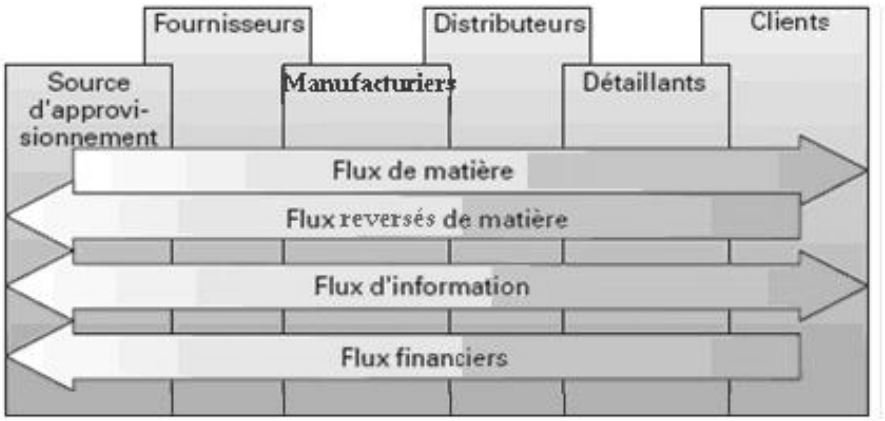


Fig. A.2 – Les flux physiques, financiers et d'information d'une entreprise industrielle

- les flux d'information : circulent de l'amont vers l'aval et de l'aval vers l'amont, voir figure (A.2).Grâce au système d'information et de communication (SIC), l'entreprise peut parvenir à piloter les flux physiques.
- les flux financiers : circulent de l'amont vers l'aval. La plupart des moyens de payement (chèque, carte de crédit,...) entraînent des échanges d'information électroniques (EDI) entre les banques du client et du fournisseur.

A.2 Gestion des chaînes logistiques (*Supply Chain Management*)

La gestion de la chaîne logistique, en abrégé GCL (en anglais SCM pour *supply chain management*) peut être définie comme la coordination stratégique des fonctions opérationnelles classiques et des fonctions tactiques à l'intérieur d'une même entreprise et entre entreprises partenaires au sein de la chaîne logistique, dans le but d'améliorer la performance à long terme de chaque entreprise de la chaîne logistique (Mentzen *et al*., 2001). En d'autres termes, il s'agit de gérer des flux physiques et des flux d'information au sein de l'entreprise et entre l'entreprise et son environnement (ressources humaines, sources d'énergie et carburants, entrepôts, machines, services divers,…).

Nous pourrons résumer les objectifs de la gestion de chaîne logistique par les points suivants :

- Prévoir les ventes clients et les plans de production;
- Améliorer le service client et l'administration des ventes;
- Améliorer la logistique entre les usines, les entrepôts et les clients;
- Améliorer la communication entre les diverses fonctions de l'entreprise et celles des partenaires;
- Améliorer la maîtrise des prévisions afin de réduire les stocks et assurer un taux de service maximal;
- Contractualiser les relations avec les fournisseurs et les sous-traitants;
- Réduire la complexité des produits par une recherche de composants standard.

En résumé, l'objectif global de la gestion de la chaîne logistique (*Supply Chain Management*) est la recherche du meilleur profit, tout en assurant un meilleur service aux clients et en optimisant les coûts, qui sont principalement des coûts de conception des produits, des coûts d'approvisionnement, des coûts de production, des coûts de stockage, des coûts de distribution, des coûts de rupture et des coûts financiers. Par conséquence, les principales activités de GCL (ou SCM) consistent à :

- gérer les achats et le management des fournisseurs et des sous-traitants;
- gérer les activités de transformation et de production des produits;
- gérer l'ensemble des stocks de l'entreprise visant le meilleur service client;
- gérer la demande client et le management de la distribution.

A.3 La planification industrielle

Les lois actuelles de l'économie imposent à tout dirigeant d'entreprise de prévoir ses activités afin d'optimiser sa politique d'investissement, de fabrication, de vente… Dans toute entreprise, il existe donc au moins un planning permettant de matérialiser ces prévisions.

Donc, nous pourrons définir la planification industrielle comme un processus qui consiste à élaborer et à réviser un ensemble de plans interdépendants (ventes, fabrication, achats, finances..) afin de garantir un meilleur équilibre entre les ressources de l'entreprise (matières premières, mains d'œuvres, moyens de production,…) et la demande du marché (commandes, marketing, vente..) en tout point de la chaîne logistique et à tout moment. Une autre définition de la planification industrielle parmi les plus utilisées, peut être trouvée dans le site de EIONET[15] (*European Environment Information and Observation Network*). Elle se concentre sur le rôle central de la planification industrielle dans la gestion de production au sein d'une entreprise ; qui est de faciliter la fabrication, la production et la transformation des produits (ou des marchandises).

Pour pouvoir donner une vision plus claire de la notion de la planification industrielle, les paragraphes suivants tracent les grandes lignes des différents niveaux de décisions de la planification industrielle.

A.3.1 Niveaux décisionnels de la planification industrielle

Dès 1965, (Anthony, 1965) distingue trois niveaux de décisions dans le cadre de la gestion des chaînes logistiques ou plus largement des systèmes industriels. Ces niveaux de planification sont déclinés selon les termes «stratégique», «tactique » et «opérationnel», voir figure (A.3). Ils sont repris par un grand nombre d'auteurs (Shapiro, 1999) (Huang *et al.*, 2003) (Gourgand *et al*., 2007) (Lemoine ,2008). Notons que l'objectif principal de la planification est d'améliorer la qualité des décisions prises par les directeurs en leur fournissant une manière disciplinée pour évaluer les conséquences de leurs actions et coordonner l'ensemble des activités dans l'entreprise.

Fig. A.3 – Niveaux de planification

[15] http://www.eionet.europa.eu

A.3.1.1 Niveau stratégique

Ce niveau de planification traite des décisions stratégiques qui ont un effet durable et qui sont considérées souvent comme une activité séparée nécessitant une mise à jour peu fréquente et des horizons assez longs pour l'analyse des résultats. Ceci, en revanche, exige des considérations d'incertitude et de risque dans le processus de prise de décision. De plus, ces décisions sont prises à un niveau supérieur, et sont concernées par les informations à la fois internes et externes à l'entreprise. Généralement, elles conduisent à définir les activités, les ressources mises en œuvre et l'infrastructure du système de production (localisation d'usines, d'entrepôts...) ou en autres termes, concevoir la structure du réseau logistique.

D'après (Genin *et al*., 2005) et (Miller, 2001), les décisions de planification de la chaîne logistique concernant le niveau stratégique sont les suivantes :

- La localisation optimale des sites (usines, entrepôts….);
- La conception d'usines et d'entrepôts;
- L'attribution des activités et des relations à ces sites;
- La capacité globale des sites;
- Le nombre de sites actifs (par exemple le positionnement et/ou la fermeture de sites);
- La conception de réseaux de transports;
- Le choix du type d'approvisionnement (mono ou multi-sources);
- La sélection des fournisseurs;
- Les marchés sur lesquels il serait intéressant de se développer;
- Le choix du type de gestion de production (par exemple la fabrication à la commande ou la fabrication sur stock);
- Les décisions de lancement de nouvelles gammes de produits;
- La définition des indicateurs et des standards pour la chaîne logistique;
- etc.

Ainsi, compte tenu de l'évolution rapide des marchés, ces décisions sont très importantes pour l'entreprise et nécessitent une grande prise de risque.

A.3.1.2 Niveau tactique

Les décisions tactiques jouent un rôle significatif dans la gestion des opérations d'une chaîne logistique. Elles correspondent à un ensemble de décisions à moyen terme inscrites dans le cadre des décisions définis par la planification stratégique. Principalement, ces décisions définissent les conditions de satisfaction de la demande des clients en élaborant des plans et des budgets, et en définissant les procédures, les politiques et les équipes de travail. (Genin *et al*., 2005) définissent le plan tactique comme un ensemble de plans définissant les volumes de distribution, de production et d'approvisionnement, ainsi que les capacités des ressources pour satisfaire les besoins

des clients finaux. Alors, une meilleure utilisation des ressources (lignes de production, sites de stockage, personnel, moyens de transport, matières premières, ...) est nécessaire.

D'après (Miller, 2001) et (Shapiro, 1998), les décisions tactiques sont :

- La répartition des approvisionnements;
- La définition des volumes et des périodes de production;
- La définition du taux d'utilisation des capacités planifiées en production, par usine et à tout niveau du réseau;
- La définition des besoins en main d'œuvre (heures normales et heures supplémentaires);
- La définition de la politique commerciale;
- La gestion des transferts intersites;
- L'affectation des demandes aux centres de distribution;
- La localisation des stocks dans le réseau de distribution et les niveaux cibles (de couverture et de sécurité);
- La définition du réseau de distribution;
- Le choix du mode de transport et des transporteurs.
- etc.

Les décisions présentées précédemment ne représentent qu'une partie de toutes les décisions pouvant être prises au niveau tactique. Quelques plans qui se déclinent de ces décisions (plan industriel et commercial, programme directeur de production,…) vont être présentés dans la section (A.4).

A.3.1.3 Niveau opérationnel

Le troisième niveau, appelé niveau opérationnel, traite des problèmes d'affectation et de contrôle des flux physiques et élabore des plans à court terme. Les décisions opérationnelles correspondent à la gestion journalière de l'entreprise dans le respect des décisions tactiques et dirigent l'exécution des tâches et l'utilisation des ressources selon les procédures établies. Les objets traités sont alors les stocks, les ordres de fabrication, d'achats et de transports ainsi que les tournées. À ce niveau, avec le respect du plan tactique et du taux de service, selon (Shapiro,1998) et (Miller, 2001), les décisions opérationnelles concernent :

- Les volumes de production et les dates d'exécution par produit, composant et sous-ensembles;
- L'ordonnancement quotidien des ressources de production;
- L'utilisation détaillée de la capacité par équipe;
- L'ordonnancement et la gestion des entrepôts;
- L'ordonnancement quotidien et hebdomadaire des unités de stockages;
- Le traitement, l'expédition et l'ordonnancement des commandes clients;

- L'ordonnancement de la main d'œuvre pour la fabrication et l'entreposage;
- L'organisation des tournées de véhicules;
- Les plans de chargement;
- La sélection des transporteurs;
- etc.

Ainsi, les décisions opérationnelles vont être déclinées de manière à ce qu'elles soient applicables au niveau de façon plus détaillée.

A.3.2 Le lien entre les différentes décisions

Comme vu précédemment, les trois classes de décisions diffèrent non seulement par l'horizon qui les caractérise, mais aussi par le niveau de compétence hiérarchique et celui d'agrégation de la décision. La figure (A.4) décrit les principales décisions données pour chaque niveau de planification et montre les liens complexes entre ces décisions pour la gestion du flux physique d'approvisionnement, de production, de distribution et de vente. Cette typologie des décisions est donnée par (Stadtler et Kilger, 2000).

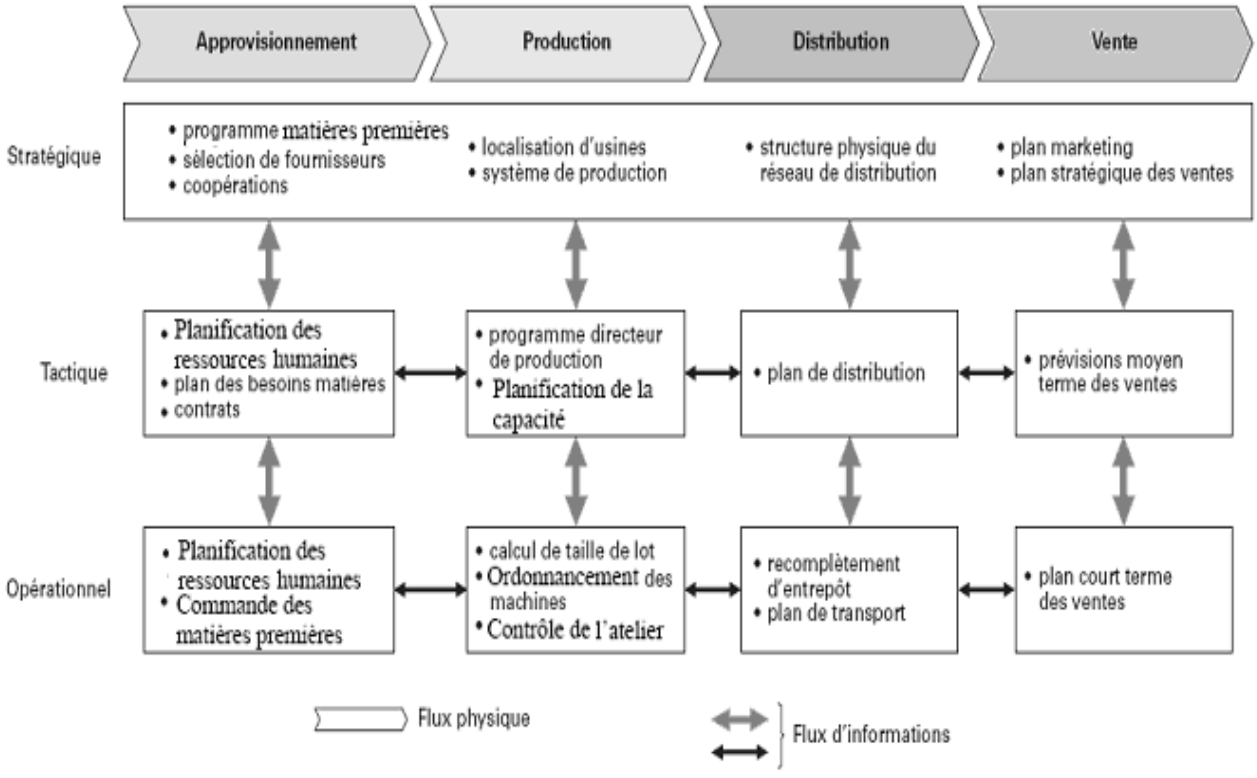


Fig. A.4 – Planification de la chaîne logistique

Dans cette section, nous avons donné les idées générales de la planification de la chaîne logistique qui vise une meilleure coordination du planning sur l'ensemble de la chaîne logistique (la gestion des demandes, le planning des fournisseurs, de la

production et de distribution). Dans la section suivante, nous nous intéressons plus spécialement au planning de production.

A.4 Management des ressources de production (*Manufacturing Resources Planning*)

Dans cette section, nous exposons les idées générales de la Planification des Besoins en Composants (CBN) ou (MRP en anglais pour *Material Requirements Planning*), son extension (*Manufacturing Resources Planning* ou MRP 2) et ses niveaux de planification.

D'abord, nous commençons par la définition de la Planification des Besoins en Composants (*Material Requirements Planning* ou MRP). C'est une méthode de gestion de production permettant d'anticiper les besoin des clients et de les planifier dans le temps. En d'autres termes, c'est une méthode de gestion des flux consistant à planifier l'ensemble des ordres de fabrication et d'approvisionnement d'après la demande commerciale prévisionnelle. Elle essaye de synchroniser les différents stocks de produits finis, de sous-ensembles, de composants, de matières premières en quantité et dans le temps sur la base d'une demande commerciale réelle et prévisionnelle.

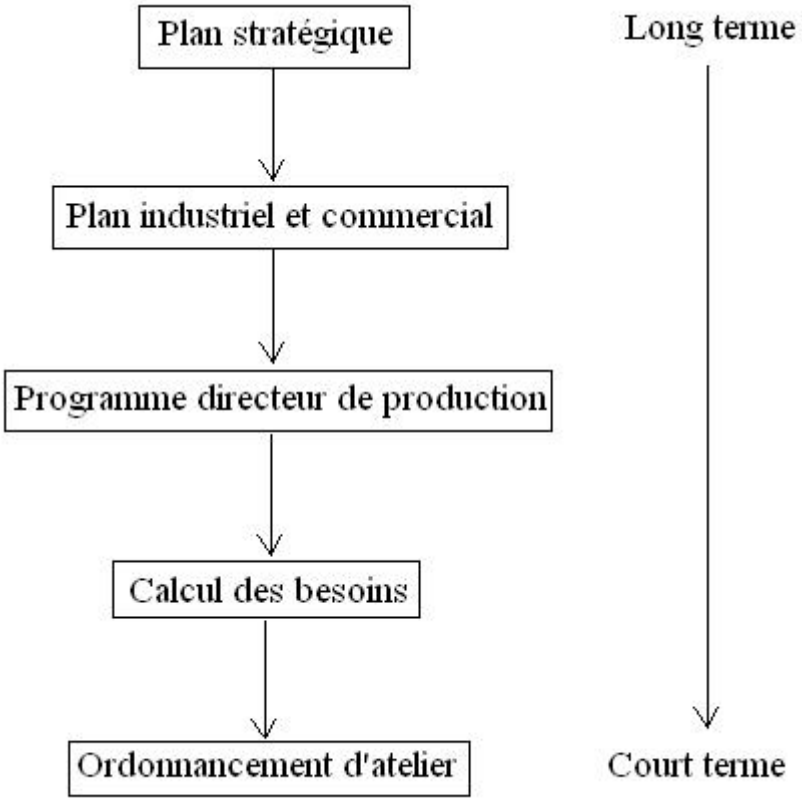


Fig. A.5 –Structure de MRP

Suite à l'évolution de la gestion industrielle, cette méthode est devenue plus globale, et cherche à planifier l'ensemble des flux et des ressources de production. Elle se

nomme MRP2 ou *Manufacturing Resources Planning*. A partir des prévisions et des demandes de clients, cette méthode gère la production du long terme au court terme, à partir de cinq niveaux de planification, voir figure (A.5) : plan stratégique, plan industriel et commercial, programme directeur de production, calcul des besoins, ordonnancement. A chaque niveau, il s'agit de satisfaire la demande du client (priorité du marché) en intégrant la disponibilité des ressources. Le niveau plan stratégique a été déjà décrit dans (A.3.1.1). Dans la suite nous présentons les quatre niveaux restants de la planification du MRP.

A.4.1. Plan industriel et commercial (niveau stratégique et tactique)

Le Plan Industriel et Commercial (PIC), ou en anglais *Sales and Operations Planning* (S&OP), ou peut être pour certains auteurs *Sales, Inventories and Operations Planning* (SIOP) comme (Womack *et al.*, 1992), se situe juste en dessous du plan stratégique. A ce niveau de détail de la planification, il fixe :

- Les volumes de production par période;
- Les niveaux de stock en début et en fin de période;
- Les ressources humaines et matérielles nécessaires;
- Les besoins financiers.

Ces plans doivent satisfaire les objectifs fixés par le plan stratégique. Ils sont le résultat d'un consensus entre les directions de marketing, des finances, de la production et de la direction de l'entreprise. Généralement, le PIC ignore les détails des produits (couleur, style, options, variantes…). A partir des prévisions de vente portant sur des familles de produits, des niveaux de stock désirés et des contraintes de production, on établit le PIC, puis on vérifie globalement l'adéquation entre la charge de production et la capacité disponible afin d'assurer le meilleur service aux clients. A ce niveau de planification, plusieurs stratégies sont envisagées (voir section A.4.2.2.1) et discutées entre les différentes directions mentionnées plus haut, pour aboutir à un consensus.

Basé sur des données commerciales et marketing telles que l'analyse de la demande, le lancement de nouveaux produits ou l'ouverture de nouveaux marchés, le PIC permet d'adapter globalement l'outil de production pour mettre un bon plan prévisionnel de production en proposant des quantités de produits par familles et des niveaux de stock disponibles à chaque période.

Plusieurs possibilités peuvent être envisagées pour satisfaire la charge et/ou la capacité, comme :

- Embaucher ou licencier du personnel;
- Envisager des heures supplémentaires ou des heures chômées, ou encore prévoir un travail en équipe;
- Elaborer du stock pendant les périodes creuses pour assurer les périodes de forte demande;
- Sous-traiter certains travaux ou louer du matériel;
- Etc.

D'après l'*American Production and Inventory Control Society* (APICS), le PIC est un processus qui permet aux décideurs de gérer leur entreprise stratégiquement, de manière à tirer un avantage compétitif sur une base d'amélioration continue en intégrant des plans marketing centrés sur le client pour des produits nouveaux et existants au *Supply Chain.*

Management : Ce processus rassemble tous les plans de l'entreprise (ventes, marketing, développement, production, approvisionnement et financier) dans un ensemble de plans intégrés. Le processus doit réconcilier les plans d'approvisionnement, de demande et d'introduction de nouveaux produits aux niveaux à la fois agrégés et détaillés et doit être lié au plan d'activités de l'entreprise. Il s'agit donc d'établir des plans à court et moyen termes qui couvrent un horizon suffisant pour planifier les ressources et appuyer le processus de planification annuel de l'entreprise (Galasso, 2007).

(Vollman *et al*., 1997) montre explicitement que le PIC se situe entre le niveau stratégique, en définissant les stratégies de la chaîne sur la base des demandes, et le niveau tactique, en définissant des plans pour satisfaire ces stratégies : « ce plan reflète les stratégies (augmenter la part de marché) et les tactiques (augmenter les stocks pour un service amélioré) qui sont réalisables par la chaîne logistique ».

Enfin, ces plans sont révisés couramment tous les mois, sur un horizon à long terme environ de 6 à 18 mois (Niveau stratégique en long terme).
Principalement, il y'a deux stratégies au niveau du PIC,

- *Production à la demande,* où il s'agit de produire les quantités, uniquement, en fonction de la demande. Cela suppose une capacité suffisamment flexible en personnel, matériel, équipements et machines lors de fortes variabilités de la demande.
- *Production nivelée,* ou il s'agit de produire des quantités relativement constantes par période (lissage de la production) comme par exemple les vêtements (produits saisonniers). Ainsi, la capacité de l'outil de production reste inchangée sur des périodes suffisamment longues. Comme la production ne suit pas la demande réelle, des stocks se forment en période de faible consommation que l'on restitue en période de forte demande (production sur stock).

Généralement, les deux stratégies ci-dessus sont combinées (production mixte). D'autres possibilités existent, comme sous-traiter de la planification ou louer du matériel (achat de capacité).

A.4.2. Programme directeur de production (Niveau tactique)

Le programme directeur de production, appelé PDP ou en anglais *Master Production Schedule* (MPS), est un programme qui définit la quantité d'un produit donné, nécessaire par période. Il est élaboré à partir des plans industriels et commerciaux et se situe juste avant le calcul des besoins. La somme des quantités définies pour les différents PDP relatifs à une famille de produits doit correspondre à la quantité totale définie par le PIC. (Genin *et al*., 2005) ont donné la définition suivante de PDP : « le programme directeur de production (PDP) est un programme établi par anticipation qui précise pour chaque produit fini, les quantités à produire, période par période ».

Le PDP est une référence pour le commercial qui doit satisfaire sa clientèle et le service production qui doit programmer sa production en tenant en compte les contraintes industrielles. Selon (Javel, 2004), il existe deux types de besoins (ou produits à besoin) : les besoins indépendants qui sont issus par exemple des ventes de pièces détachées ; et les besoins dépendants (matière première, composant acheté, sous-ensemble fabriqué) qui sont issus du calcul des besoins. Par conséquent, les besoins indépendants ne peuvent être, la plupart des cas, qu'estimés par prévision, alors que les besoins dépendants doivent être calculés.

A partir des besoins indépendants (prévisions et commandes fixes), des prévisions de ventes et des niveaux de stock des produits, le PDP peut être établi et a pour objectifs :

- D'anticiper les besoins des commerciaux (prévisions);
- De traduire la volonté de la direction, exprimée dans le PIC, sous forme opérationnelle, c'est-à-dire en quantités à produire ou à acheter;
- De transmettre l'échéancier et les quantités à produire pour le calcul des besoins;
- De suivre l'évolution des ventes par rapport aux prévisions;
- De garantir un taux de service client tout en minimisant le niveau de stock;
- De faire le meilleur usage possible du matériel, de l'équipement et des ressources;
- De maintenir les immobilisations stockées aux niveaux requis;
- D'informer le commercial des produits disponibles à la vente.

L'horizon du PDP se définit, au minimum, par le cumul des délais les plus longs d'approvisionnement, de fabrication et d'assemblage pour les composants et les produits finis. Il peut varier de 3 mois à un an selon les types de produits. Il est révisé généralement chaque semaine.

Enfin, pour une vision plus claire, nous présentons deux notions importantes pour le PDP : disponible à la vente et stock prévisionnel.

- *Disponible à la vente* (ATP: Available to promise) : c'est l'ensemble des produits fabriqués à partir des prévisions de vente et en attente de confirmation des commandes fixes. Cela comprend tout ce qui peut être vendu, sans modification du PDP. Le disponible à la vente est un outil de gestion des commandes pour le commercial. A un instant (t) du planning, le disponible à la vente des périodes à venir correspond à la zone ombrée (figure A.6).
- *Stock prévisionnel* : c'est le stock prévu pour chaque période à venir dans le calcul des besoins commerciaux (prévisions et commandes clients). Il correspond pour chaque période, au stock physique prévisionnel moins les besoins (la plus grande valeur entre les prévisions et les commandes), additionné des réceptions prévues et des ordres prévisionnels.

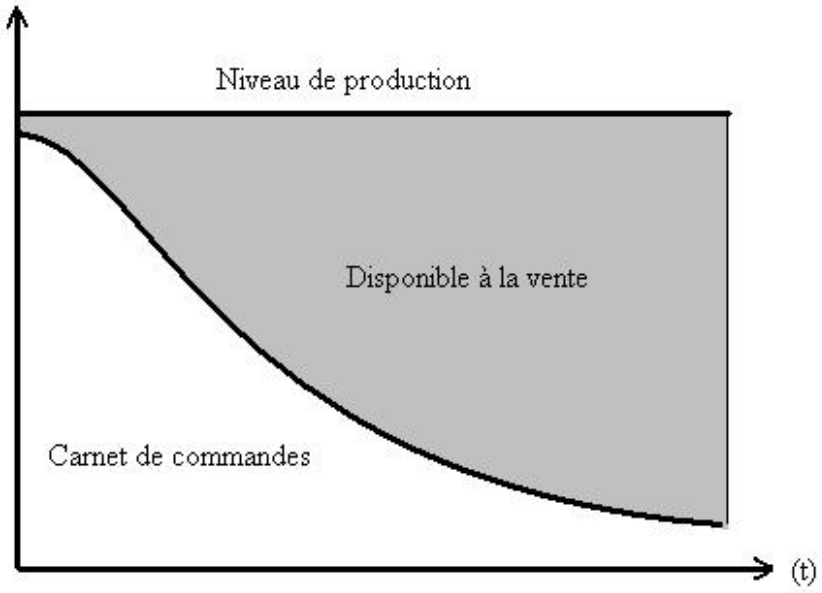


Fig. A.6 –Disponible à la vente

A.4.3 Calcul des besoins

Le calcul des besoins constitue l'étude détaillée des besoins dépendants de l'échéancier des PDP. L'objectif du calcul des besoins est de respecter le niveau du service client tout en optimisant l'utilisation du matériel, de la main d'œuvre et de l'équipement au moindre coût.
Là, nous distinguons deux types de besoins :

- *Le besoin brut* : c'est le total des demandes dépendantes et indépendantes d'un même article sans prendre en compte le niveau de stock ni les réceptions prévues. Ces demandes dépendantes sont obtenues en multipliant le coefficient d'emploi de chaque article par les quantités figurant à des niveaux supérieurs de nomenclatures et décalées du délai d'obtention.
- *Le besoin net* : c'est la quantité obtenue en déduisant des besoins bruts et les réservations (quantités allouées à un ordre de prélèvement, non encore sorties du magasin), le stock disponible ainsi que les réceptions prévues, et en ajoutant le stock de sécurité. Suivant la technique d'affectation de lot retenue et le décalage lié au délai d'obtention, les besoins nets deviennent des ordres prévisionnels (achat ou fabrication).

L'idée principale du calcul des besoins est de répondre aux questions suivantes : " Quel article fabriquer, combien en fabriquer et quand programmer sa fabrication?" ; et de fournir l'ensemble des ordres de fabrication et d'achat prévisionnels des articles à partir d'un échéancier journalier, en prenant en compte :

- L'échéancier des programmes directeurs de production.
- Les nomenclatures "produits" : la liste de tous les sous-ensembles, éléments et matières premières nécessaire à la fabrication d'un produit.

- Les délais d'assemblage et de fabrication des articles et des composants, ainsi que les délais d'approvisionnement des articles achetés.
- Les niveaux de stocks par article et par composant.

Le calcul des besoins doit respecter les contraintes de capacité des machines ou des centres de production. Le résultat du calcul des besoins est de proposer des ordres (de fabrication et d'achat) et d'émettre des messages d'action et d'alerte sur différents ordres de fabrication (OF) (Lancer, avancer, repousser, annuler, ...).

Pour construire un échéancier de calcul des besoins, certaines informations (données de production) sont nécessaires, tels que :

- *La taille de lot :* c'est la quantité déterminée d'articles à produire,qui est généralement la quantité économique. Si la quantité à produire est supérieure à la taille de lot, alors on prend un multiple de ce lot. Plus on se place très haut dans la nomenclature (articles coûteux), plus la taille de lot devrait correspondre aux besoins nets afin de réduire les coûts de stockage et se rapprocher du " juste nécessaire". On appelle cela « lot pour lot ».
- *Le délai d'obtention* : c'est la durée nécessaire à la réalisation de l'ordre (de fabrication ou d'approvisionnement).
- *Le délai de fabrication* : comprend le temps d'attente devant la machine, le temps de préparation de la machine, le temps de transformation, et le temps de transfert.
- *Le délai d'approvisionnement* : correspond à la durée entre l'expression d'une demande et la mise à disposition des produits correspondant à la demande.
- *Le lien de nomenclature :* c'est la relation entre le composé et le (ou les) composant(s) qui le constitue (nt). Ce sont les relations "parent-enfant".
- *Le stock physique de départ :* il correspond au stock réel d'articles en magasin au moment du calcul des besoins.
- *Le stock prévisionnel :* c'est le stock obtenu après chaque calcul des besoins, par période.

Notons qu'un ordre proposé par le système a une date de début et une date de fin, calculées grâce au délai d'obtention. Le gestionnaire transforme les ordres proposés selon les périodes de planification suivant deux statuts :

- *L'ordre ferme (gelé) en quantité et dans le temps,* qui ne peut pas être modifié automatiquement par le système informatique, mais uniquement par le responsable du planning.
- *L'ordre lancé,* signifie qu'une décision d'exécution est prise (fabrication ou commande) et qu'une date de réception attendue est prévue.

Enfin, l'horizon de planification tient compte des délais d'achat et de fabrication des éléments à réaliser. Ils sont révisés généralement toutes les semaines, parfois tous les jours, sur un horizon de 1 à 3 mois.

A.4.4 Ordonnancement d'atelier

L'ordonnancement d'atelier ou quelques fois (gestion et pilotage d'atelier) couvre un ensemble d'actions qui transforment les décisions de fabrication définies par le programme directeur de production (PDP) en instructions d'exécution détaillées destinées à piloter et contrôler à court terme l'activité des postes de travail dans l'atelier. Alors, l'horizon de planification est de l'ordre de la journée à la semaine, et la révision journalière est parfois horaire.

La fonction ordonnancement d'atelier peut être décomposée en trois sous-fonctions :

- Elaboration des ordres de fabrication : cette tâche consiste à transformer les informations du programme directeur de production (suggestions de fabrication) en OF (Ordres de fabrication);
- Elaboration du planning d'atelier : cette tâche consiste, en fonction de ces ordres de fabrication et de la disponibilité des ressources consommables (matières premières, composants) et partageables (postes de travail), à déterminer le calendrier prévisionnel de fabrication (cela revient à transformer les prévisions de fabrication à court terme en ordres d'exécution à très court terme);
- Lancement–suivi : cette tâche consiste à distribuer aux postes de travail les documents nécessaires à la bonne exécution des fabrications (lancement en fabrication) et à suivre l'exécution des fabrications (suivi de production).

Ces trois sous-fonctions s'enchaînent de la manière suivante dans la figure (A.7), (javel, 2004):

L'objectif final de l'ordonnancement est avant tout de piloter la production de l'entreprise. Ce pilotage peut être soit centralisé, dans ce cas, il est réalisé par la fonction ordonnancement de l'entreprise, ou décentralisé, dans ce cas, il est réalisé au pied de chaque poste de travail. Dans le cas d'un ordonnancement centralisé, qui correspond au type le plus répandu dans les entreprises, la structure de fonctionnement correspond à la figure (A.7). Cette solution a l'avantage de proposer un planning d'atelier très complet mais elle a l'inconvénient de centraliser la prise de décision. Dans un ordonnancement décentralisé, ou local, la décision est prise en fonction d'informations sur les lots en attente devant un poste de charge sans avoir à considérer la situation des autres files d'attente. Cette solution a l'avantage de réduire, quelquefois, les délais de réalisation mais a l'inconvénient de ne pas régler le problème de la gestion des capacités des postes et de ne pas fournir un planning d'atelier de synthèse.

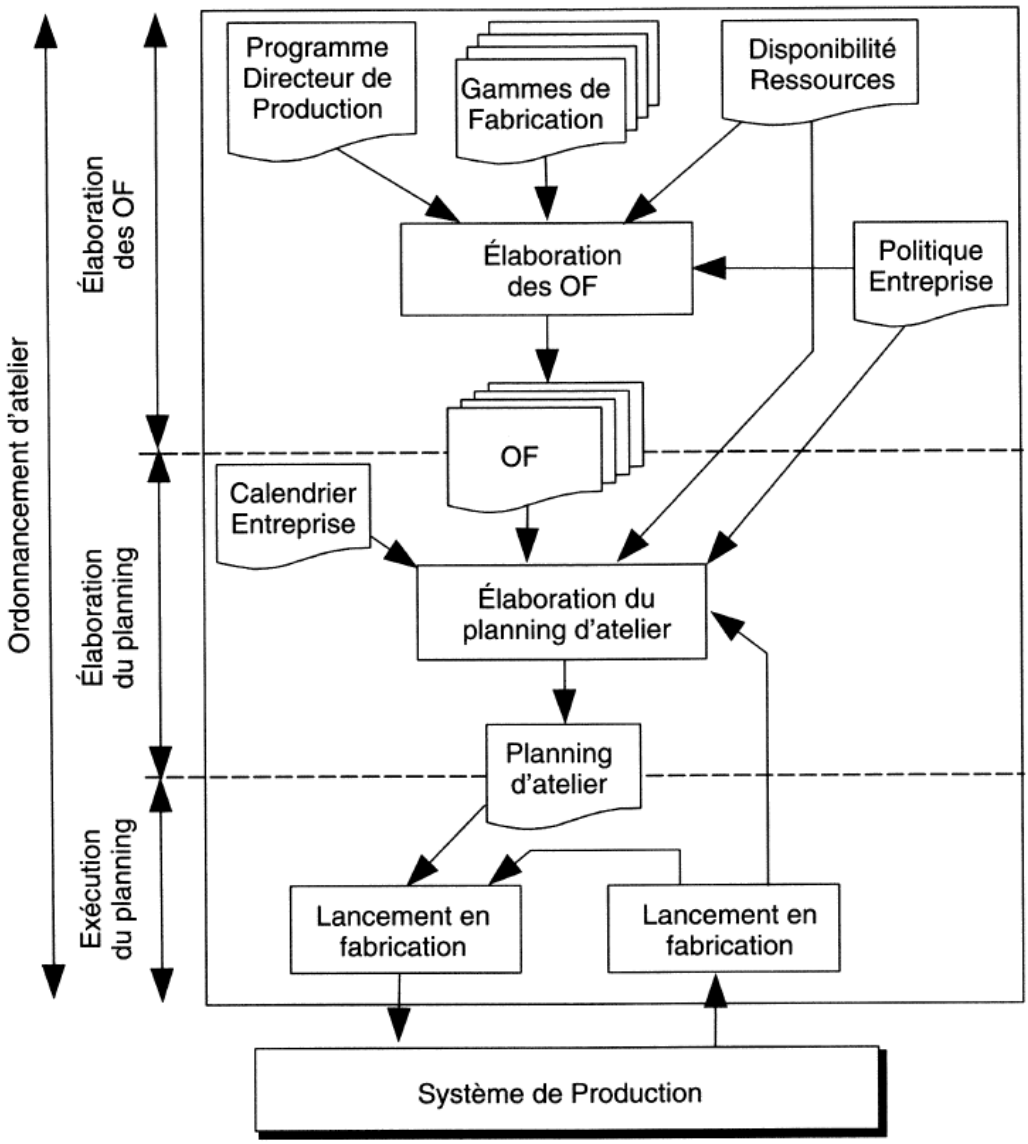


Fig. A.7 –Fonction d'ordonnancement

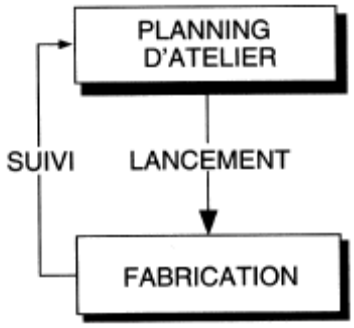


Fig. A.8 –Les opérations de planning d'atelier

Pour piloter un atelier, il faut mettre en œuvre les procédures permettant de faire travailler le système de production conformément aux prévisions établies. Ces procédures correspondent aux opérations de (lancement) en fabrication et de (suivi) de la fabrication suivant l'enchaînement simplifié présenté en figure (A.8), qui présente deux grandes opérations : le lancement et le suivi de la production.

A.4.4.1 Le lancement de production

Le lancement de production consiste, comme son nom l'indique, à distribuer les documents nécessaires à la bonne exécution des fabrications conformément aux prévisions établies sur le planning de façon à synchroniser la circulation des pièces entre les postes et d'assurer ainsi, la continuité du flux physique.

Il est possible, à ce stade, d'envisager le changement éventuel de machine en cas d'indisponibilité de celles prévues initialement, ou pour anticiper le début de fabrication, à condition d'avoir spécifié des machines de substitution dan la gamme de fabrication et que celles-ci soient en sous-charge. De même, cette fonction effectue un dialogue permanent avec la fabrication dans le cas de rattrapage de retards. Si les retards ne peuvent être résolus dans l'atelier, le lancement étudiera toutes les possibilités d'absorption dans des limites qui ne remettent pas en cause le planning. Au-delà de ces limites, il faudra revoir la planification des fabrications. Le lancement se fait entre 1 et 3 unités. Pour une unité de lancement, il est possible d'accorder 2 unités de marge avant la fin de la fabrication prévue, au-delà on re-planifie la production. Généralement, ce lancement peut se faire par différentes techniques :

- *Par plan* : il n'existe pas vraiment de consignes de travail. Le début d'exécution est déclenché lors de la remise d'un plan de réalisation à un poste susceptible de réaliser le produit. Dans ce cas il n'est guère possible de faire des prévisions et il n'existe généralement pas de planning, ou, s'il en existe un, nous pouvons émettre beaucoup de doutes quant à sa représentativité;
- *Par ordre de fabrication* : les consignes de fabrication sont transmises au premier poste de transformation sous forme d'une fiche suiveuse accompagnée de tous les documents nécessaires à la réalisation (plan, bon de sortie matière, fiche de description de phase,…).

Dès que l'opérateur du poste a terminé le travail, décrit sur la fiche suiveuse, qui lui était demandé, il transmet cette fiche au poste suivant en même temps qu'il transfère le produit à fabriquer. Ce type de lancement diminue les contrôles administratifs mais demande une compétence et une attention soutenue de l'agent de maîtrise. Cette technique est acceptable dans une entreprise ayant peu d'en-cours et dont les cycles de réalisation sont courts.

- *Par phase de travail* : les consignes de fabrication sont transmises au début de chaque période de temps à chaque poste de transformation sous forme de bons de travail accompagnés de tous les documents nécessaires à la réalisation (plan, bon de sortie matière, fiche de description de phase…). Chaque bon travail correspond à une phase d'une gamme de fabrication. Cette technique, plus courante dans les grandes entreprises, nécessite des règles organisationnelles strictes puisque l'ordonnancement est réalisé par plusieurs personnes. Elle a

l'inconvénient d'allonger les cycles de réalisation, elle sera donc privilégiée pour les entreprises ayant beaucoup d'en-cours et des cycles de productions élevés.

A.4.4.2 Suivi de production

Toujours lié à la fonction lancement, le suivi de production a pour but d'informer les responsables de l'entreprise de l'état d'avancement des travaux. Il est nécessaire de suivre avec beaucoup de soins les fabrications qui nécessitent une attention toute particulière du fait de leur importance (urgence, chemin critique). Le suivi de production peut avoir trois objectifs :

- Le suivi technique, qui consiste à rassembler les indications concernant l'état d'avancement des travaux. Ces indications fournies par suivi doivent permettre :
  - De définir les ordres suivants à lancer en fabrication;
  - De mesurer l'efficacité de l'outil de production;
  - De pouvoir renseigner les clients sur l'état d'avancement de la réalisation de leurs commandes.
- Le *suivi comptable* : ce suivi consiste à rassembler les indications concernant les temps de réalisation des travaux afin de pouvoir déterminer les coûts de revient des produits. La connaissance immédiate de l'information étant moins impérative. Ce suivi est souvent un résultat du suivi technique.
- Le *suivi de la main d'œuvre* : ce suivi est souvent confondu avec le suivi comptable. Il consiste à rassembler les indications concernant les temps de présence des opérateurs afin d'établir les informations nécessaires à l'établissement des salaires des employés.

A.5 Transport (distribution)

La définition générale du transport est le fait de porter quelque chose, ou quelqu'un, d'un lieu à un autre, en utilisant des moyens de transport appelés des véhicules (véhicules automobiles, trains, aéronautiques, navires, ...) et des infrastructures empruntées qui consistent en des voies de communication (comme des routes, des voies ferrées, des canaux, des fleuves, ..) et d'aménagements (parkings, ports, aéroports, ...).Le transport requiert de la main-d'œuvre, des équipements, des consommations intermédiaires, et modifie les caractéristiques physiques des produits (leurs coordonnées spatio-temporelles) tout comme les opérations manufacturières en modifient la composition ou la morphologie. Dans ce cas, il faudra distinguer entre le transport et la manutention (*Handling*). Pratiquement, la différence entre eux s'est établie en fonction des matériels utilisés (un camion n'est pas un chariot élévateur) et des infrastructures empruntées. En effet, la manutention ne sort guère des emprises propres à un établissement tandis que le transport utilise le plus souvent une infrastructure publique et partagée entre de nombreux usagers. Nous pouvons alors dire que la manutention est

un transport des produits dans l'entrepôt (mise en stock, préparation de commandes, réapprovisionnement du picking, etc.)

En général, la valeur du transport s'incorpore à la valeur totale du produit, qui additionne coût des matières premières, coût de fabrication, coût de transport, coût de stockage, etc. En outre, l'opération de transport se déroule dans le temps, et le temps de transport s'ajoute intégralement au cycle de rotation du produit depuis l'engagement de sa fabrication jusqu'à sa vente sur le marché. En termes d'organisation, l'accomplissement de ce processus se répartit entre deux régimes (types du transport). Certaines entreprises industrielles ou commerciales utilisent des moyens de transport (humains et matériels) pour effectuer elles-mêmes tout ou une partie du transport requis par leur activité. Dans ce cas, nous parlons alors de **transport pour compte propre**. Mais une large part du transport de produits est assurée par des entreprises spécialisées, et nous parlons alors de **transport pour compte d'autrui**.

D'autre part, nous appelons le chemin commercial parcouru par un produit pour aller du producteur au client (consommateur) final par le *canal de distribution*. Avant de présenter les différents types de canaux de distribution, nous devons présenter la définition et les types d'intermédiaires.

A.5.1 Les types d'intermédiaires

L'intermédiaire dans une chaîne de distribution de produits est quelqu'un qui achète de grandes quantités de marchandises et les revend à des commerçants plutôt qu'au consommateur final. Nous pourrons distinguer deux types d'intermédiaires :

- *Le grossiste:* peut être défini comme une entreprise indépendante du producteur. Il achète du producteur et revend à d'autres intermédiaires en recherchant ou non un profit, comme par exemple, un grossiste qui achète une grande quantité d'aliments chez des producteurs et la revend à des restaurants, à des institutions scolaires ou hospitalières, à des détaillants. Plusieurs avantages proviennent d'existence des grossistes dans les canaux de distribution, nous les résumons par les points suivants[16] :
  - l'entreposage régional, assez proche des clients intermédiaires et d'accès facile pour eux;
  - la livraison des marchandises chez les clients intermédiaires, par certains grossistes;
  - le crédit offert, parfois, aux clients intermédiaires;
  - la réception de grandes quantités de produits en provenance des producteurs et la répartition géographique de ces produits;
  - la présentation de plusieurs qualités aux détaillants;
  - la diminution des coûts de transport de marchandises, puisque celles-ci sont expédiées en grandes quantités du lieu de production jusqu'aux entrepôts du grossiste;
  - la transmission d'informations vers l'aval (c'est-à-dire vers le détaillant et le consommateur final) et vers l'amont (producteur);

[16] http://www.mvtechnologie.com/MarkDistribution.htm

  - la réduction du nombre de transactions : un producteur devrait transiger avec plusieurs centaines de détaillants éparpillés, et pour de petites quantités de marchandises. Le grossiste permet le regroupement des besoins des détaillants; il passe une commande unique aux producteurs, ce qui entraîne une diminution des coûts administratifs (facturation, courrier, téléphone, etc.).
- *Le détaillant :* qui achète les produis du producteur ou du grossiste pour revendre au consommateur final, en ajoutant des valeurs intangibles pour offrir quelques services aux consommateurs, comme par exemple[16]:
  - la proximité et la commodité d'accès, pour le consommateur final (stationnement, confort, variété, etc.);
  - des heures d'ouverture parfois très tardives;
  - un classement et une exposition des produits par catégories, facilitant ainsi le choix et la comparaison entre les différentes marques et les différentes tailles d'emballages;
  - parfois un service de crédit;
  - parfois la livraison et l'installation à domicile de certains équipements ménagers;
  - le service après-vente comme les réparations;
  - le conseil de vendeurs sur les lieux mêmes de la vente.

A.5.2 Les types de canaux de distribution

- **Le canal court** : ici, nous pouvons trouver aussi deux cas. Un *canal ultra court* qui met directement en relation le producteur et le consommateur final sans intermédiaires entre eux, comme par exemple un traiteur qui prépare la nourriture et le vend directement au client. Et l'autre cas (plus commun) est un *canal court* qui contient un seul intermédiaire (souvent un détaillant) entre le producteur et le consommateur comme par exemple la vente de vêtements dans une boutique. En plus, les producteurs de produits peuvent utiliser leurs propres moyens de vente au consommateur, comme par exemple l'utilisation des employés (vendeurs-représentants) qui jouent le rôle des conseillers auprès des clients et de négociateurs des termes de la vente.
  Notons que le coût d'utilisation d'un canal court est élevé pour le producteur car il assume souvent les frais de contact et de distribution jusqu'au consommateur. Donc, Il doit en tenir compte dans l'établissement de son prix de vente.
- **Le canal long** fait intervenir un nombre d'intermédiaires indépendants égal ou supérieur à deux entre le producteur et le consommateur. Ces intermédiaires ajoutent des valeurs tangibles et intangibles aux produits et aux services en cours de distribution, comme par exemple la vente de fruits et légumes au marché. Alors, pour le producteur, le coût d'utilisation d'un canal long est moindre que celui d'un canal court, que les tâches de contact avec les clients, de transport et d'entreposage sont assurées par plusieurs intermédiaires.

Annexe B

Les symboles de la notation des problèmes de tournées classés par ordre alphabétique

π: Le problème étudié [RP pour problème de tournées (Routing Problem)].
α1: Le champ de ressources fixes.
α2: Le champ de ressources mobiles.
α 3: Le champ de demandes.
β: Contraintes.
γ: Objectifs.
↔: Compatibilités.
A: Localisations sur Arcs.
Aut: Autonomie.
Aut: Ve: Autonomie des véhicules.
Aut: Wo: Temps limité fixant la durée de travail maximale des travailleurs.
B: Livreur(Backhaul).
C: Clients.
C:B: Client livreur (Backhaul).
C:(Li,B): Deux types de clients: receveur (Linehaul), livreur (Backhaul).
C:(Li+B): Client receveur (Linehaul) et livreur(Backhaul) en même temps.
Cap: Capacité.
Cap: Ve: Capacité limitée des véhicules.
Cap:Dp: Capacité limitée de stockage dans les dépôts.
Cap:C: Capacité limitée de stockage chez les clients.
Cap:Ve:C Capacités limitées de véhicules et de stockage chez les clients.
Comb: Choix de service selon un ensemble de combinaisons de périodes.
D: Dynamique.
Dd: demande.
Dd : D: Demande dynamique.
Dd : S: Demande stochastique
Dd : D: C: Nombre de clients dynamique.
Dd : D: Q: Quantité dynamique.
Dd : D: Lo: Localisation de client dynamique.
Dp: Dépôt.
Dp:Li: Dépôt receveur (Linehaul).
Dp: (B+Li): Dépôt livreur(Backhaul) et receveur (Linehaul) en même temps.
Dp:(B, Li): Deux types de dépôts: le premier livreur(Backhaul) et le deuxième receveur (Linehaul).
DpVar: Dépôt de départ d'un véhicule variable pendant l'horizon de planification.
Dp:W: Dépôt de type Entrepôt (Warehouse).
(Dp, C): Les produits livrés proviennent des dépôts et des clients.

F: Fréquence du service.
H: Horizon de planification.
Het : Flotte hétérogène de véhicules.
Li: Receveur (Linehaul).
Lo: Localisation.
MDp: Multi dépôts.
MDpP: Multi dépôts de départ.
Mix: Localisations mixtes (Arcs+Noeuds).
MixNet: Réseau mixte (Mixed Network).
NoR: Retour du véhicule à son dépôt original non obligatoire.
MP: Multi périodes.
MPr: Multi produits.
MT: Multi trips.
Net: Réseau (Network).
NoSer: Nombre de clients non servis.
NVe: Nombre de véhicules fixé > 1.
NVe : 1: Nombre de véhicule fixé à 1.
NWo: Nombre de travailleurs (Workers).
ONet: Réseau Orienté (Oriented Network).
OT: Autre cas (Other).
Over: Violation.
Over: β: Violation d'une contrainte.
Over: TW: Violation de la contrainte de fenêtre de temps (retard chez les clients).
Over: Cap: Violation de la contrainte de capacité.
Over: Aut: Violation de la contrainte d'autonomie.
Over: Aut:Wo: Violation des heures supplémentaires des travailleurs.
P: Période.
Paired: Demandes appariées.
Pr: produit.
Prec: Précédence.
Q: Quantité.
S: Stochastique.
SD: Split Delivery (partage possible par demande et par produit).
SD :NoPr: Split Delivery par demande seulement (cas des véhicules à compartiments).
Ser: Coût de service.
Ser:T: Temps de service.
Spa:Min~Max: Espacement minimum (∞ si illimité) et maximum (∞ si illimité) entre deux services successifs pour un même client.
Tr: Coût de transport.
Tr:Dis: Distance totale parcourue.
Tr:T: Temps de transport.
Tra: Conditions de trafic.
Tw: Fenêtre de temps (Time windows).
Tw : C: Fenêtre de temps à respecter pour pouvoir servir le client.
Tw : Dp: Fenêtre de temps traduisant les horaires d'ouverture du dépôt.
Tw : Net: Fenêtre de temps de certaines parties du réseau.

Tw : Ve: Fenêtre de temps définissant la disponibilité des véhicules.
Tw : Wo: Fenêtre de temps traduisant les horaires de travail des travailleurs.
Ve: Véhicules.
W: Entrepôt (Warehouse).
Wo: travailleurs (Workers).
WT: Temps d'attente chez les clients (Waiting Time).
Unpaired: Demandes indépendantes

Annexe C

Les résultats des expériences du chapitre 4

C.1 Résultats de la phase de test Préalable

C.1.1 Résultats de la phase de tests Préalable pour le codage indirect type B

C.1.1.1 Plan des essais

Ce plan contient 288 essais.

Taux de croisement	Croisement	Mutation	Génération	Distance
85	UX	Swapping	15000	1238,61
65	PMX	Swapping	10000	2881,24
60	PMX	Swapping	10000	2865,99
90	UX	SwapInvers	15000	1311,91
80	PMX	Swapping	5000	2973,67
80	PMX	Swapping	10000	2907,66
85	UX	Inversion	10000	1495,66
95	UX	Inversion	15000	1281,60
60	PMX	Swapping	5000	2955,71
75	OX	Inversion	15000	1168,23
90	OX	Swapping	15000	1728,98
70	PMX	SwapInvers	5000	1954,64
70	OX	Swapping	10000	2126,00
75	UX	Swapping	10000	1392,26
75	PMX	Swapping	15000	2867,32
70	PMX	Inversion	10000	1762,85
95	OX	SwapInvers	15000	1197,56
95	UX	SwapInvers	20000	1412,99
95	OX	Inversion	15000	1249,05
90	UX	Inversion	5000	1186,25
85	OX	Swapping	15000	1870,18
70	UX	Inversion	5000	1232,93
60	OX	SwapInvers	5000	1325,42
60	PMX	Swapping	15000	2820,40
65	PMX	Inversion	20000	1338,21
85	UX	Swapping	20000	1253,26
70	UX	Swapping	10000	1543,84
65	OX	Swapping	10000	2084,09
80	OX	Swapping	20000	2029,79
80	PMX	Swapping	15000	2862,30

Taux de croisement	Croisement	Mutation	Génération	Distance
80	UX	Inversion	10000	1472,19
60	OX	Inversion	20000	1231,80
85	PMX	Inversion	15000	1516,81
65	UX	Swapping	10000	1141,68
85	OX	Inversion	15000	1077,21
95	OX	Swapping	15000	1886,28
65	OX	Swapping	20000	2067,43
65	UX	Swapping	5000	1179,79
70	OX	Inversion	10000	1295,64
70	UX	SwapInvers	10000	1588,12
90	OX	Swapping	10000	2064,83
90	UX	Inversion	15000	1350,62
85	PMX	SwapInvers	5000	1858,51
70	PMX	Swapping	5000	2951,63
80	UX	Inversion	15000	1493,15
60	PMX	SwapInvers	20000	1390,11
75	PMX	SwapInvers	10000	1816,69
80	UX	Swapping	20000	1303,27
80	PMX	Inversion	20000	1378,48
85	PMX	Inversion	10000	1655,86
65	UX	SwapInvers	10000	1515,86
60	PMX	Swapping	20000	2883,94
70	OX	Inversion	5000	1485,61
80	PMX	Inversion	15000	1989,16
90	UX	Swapping	5000	2023,01
70	UX	SwapInvers	15000	1428,28
95	UX	SwapInvers	10000	1478,32
95	PMX	Inversion	15000	1828,44
95	OX	SwapInvers	5000	1439,52
85	OX	Swapping	5000	2142,57

Taux de croisement	Croisement	Mutation	Génération	Distance
65	OX	SwapInvers	15000	1265,22
90	OX	Inversion	15000	1212,20
75	PMX	Inversion	5000	1954,11
75	PMX	SwapInvers	5000	1842,89
80	UX	Inversion	20000	1357,33
95	UX	Swapping	20000	1363,50
70	UX	Swapping	5000	1949,06
65	OX	Inversion	5000	1319,23
60	UX	Inversion	5000	1667,44
95	OX	SwapInvers	10000	1439,68
85	OX	Swapping	20000	1958,66
65	OX	Inversion	15000	1160,56
65	UX	SwapInvers	15000	1320,74
65	PMX	Swapping	5000	2952,15
85	OX	Inversion	10000	1287,06
90	PMX	Swapping	20000	2947,15
65	OX	SwapInvers	10000	1313,16
60	OX	Swapping	20000	1801,91
70	UX	Swapping	20000	1125,51
70	OX	SwapInvers	15000	1267,55
90	UX	Swapping	20000	1637,47
65	UX	Inversion	15000	1068,85
60	UX	Inversion	10000	1073,60
90	PMX	SwapInvers	15000	1372,66
90	UX	Inversion	10000	1446,30
80	OX	SwapInvers	10000	1201,98
75	OX	SwapInvers	10000	1134,21
95	PMX	SwapInvers	5000	1888,66
85	PMX	Swapping	5000	2979,99
70	OX	Swapping	20000	1592,80
85	OX	SwapInvers	5000	1317,44
85	UX	Inversion	5000	1158,45
90	OX	SwapInvers	5000	1359,21
85	UX	SwapInvers	15000	1535,16
60	OX	Inversion	5000	1248,55
75	OX	SwapInvers	15000	1116,46
85	OX	SwapInvers	10000	1243,41
85	PMX	Inversion	20000	1410,16
60	PMX	SwapInvers	15000	1732,04
75	UX	Swapping	20000	1105,19
65	UX	Swapping	20000	1542,91
60	OX	Swapping	10000	1803,07
80	UX	Swapping	10000	1393,91
95	PMX	Swapping	5000	2796,67
70	PMX	Inversion	20000	1748,37
80	UX	Inversion	5000	1141,28
70	PMX	SwapInvers	20000	1681,88
65	PMX	SwapInvers	15000	1562,86
60	UX	Inversion	15000	1355,64
65	PMX	Inversion	5000	2067,99
80	PMX	Inversion	5000	1807,78
90	PMX	SwapInvers	20000	1597,75
90	PMX	Swapping	10000	2867,58
75	OX	Swapping	15000	1836,53
90	OX	Swapping	20000	1791,00
90	PMX	Inversion	10000	1746,21
95	PMX	Swapping	15000	2893,28
70	PMX	Inversion	15000	1749,42

Taux de croisement	Croisement	Mutation	Génération	Distance
85	OX	SwapInvers	20000	1156,58
90	PMX	Inversion	5000	2264,47
90	UX	Swapping	10000	1218,22
75	UX	SwapInvers	20000	1368,50
65	OX	SwapInvers	5000	1416,92
80	PMX	SwapInvers	5000	1891,53
65	PMX	SwapInvers	10000	1693,31
90	OX	SwapInvers	15000	1355,15
60	UX	SwapInvers	5000	1617,63
60	UX	Swapping	5000	2525,41
75	UX	Inversion	5000	1408,15
75	OX	Inversion	20000	1233,06
75	PMX	Swapping	10000	2984,96
80	OX	Inversion	15000	1354,07
85	UX	SwapInvers	5000	1742,98
80	PMX	Swapping	20000	2941,30
80	OX	Swapping	5000	1841,95
75	OX	SwapInvers	5000	1253,12
60	OX	Inversion	10000	1369,54
60	OX	SwapInvers	10000	1208,43
95	UX	SwapInvers	15000	1210,97
90	PMX	Swapping	5000	2905,41
95	UX	Inversion	20000	1347,12
85	PMX	Swapping	20000	2906,71
70	PMX	Inversion	5000	2025,03
75	UX	Inversion	15000	1334,35
65	UX	SwapInvers	20000	1209,15
90	UX	SwapInvers	20000	1432,71
75	OX	Swapping	20000	1690,26
70	UX	Inversion	10000	1292,64
95	PMX	SwapInvers	15000	1513,84
95	PMX	SwapInvers	10000	1936,19
60	UX	Swapping	15000	1310,95
85	OX	SwapInvers	15000	1229,97
75	OX	Swapping	5000	1983,29
85	PMX	Swapping	15000	2952,30
80	UX	SwapInvers	15000	1414,29
75	PMX	Inversion	10000	1694,12
60	OX	Swapping	15000	1986,68
80	OX	SwapInvers	20000	1154,21
75	PMX	SwapInvers	15000	1775,20
95	PMX	SwapInvers	20000	1824,33
75	UX	SwapInvers	15000	1553,60
75	UX	Inversion	20000	1424,18
80	OX	SwapInvers	5000	1202,19
75	PMX	SwapInvers	20000	1615,18
60	OX	Swapping	5000	1957,96
70	OX	SwapInvers	20000	1155,99
90	UX	SwapInvers	5000	1637,80
90	UX	Swapping	15000	1338,13
60	PMX	SwapInvers	5000	1932,92
75	OX	SwapInvers	20000	1138,09
80	OX	Swapping	10000	1955,38
90	UX	SwapInvers	10000	1369,17
70	OX	SwapInvers	5000	1327,33
90	PMX	SwapInvers	10000	1885,96
95	OX	SwapInvers	20000	1136,21
65	UX	SwapInvers	5000	1498,20

Taux de croisement	Croisement	Mutation	Génération	Distance
90	PMX	Swapping	15000	2977,40
90	OX	Inversion	5000	1363,66
95	PMX	Inversion	10000	1742,63
70	UX	Swapping	15000	1194,00
65	PMX	Inversion	15000	1562,62
60	UX	SwapInvers	15000	1332,20
65	UX	Inversion	20000	1424,40
80	UX	Swapping	15000	1136,91
90	PMX	SwapInvers	5000	1753,80
95	UX	Inversion	5000	1098,02
80	PMX	SwapInvers	15000	1583,25
90	PMX	Inversion	20000	1619,04
65	OX	SwapInvers	20000	1210,52
75	OX	Inversion	5000	1264,91
85	OX	Swapping	10000	2019,61
65	PMX	Swapping	15000	2902,80
80	OX	Inversion	20000	1072,62
70	UX	Inversion	20000	1293,79
65	UX	Swapping	15000	1953,55
60	OX	SwapInvers	20000	1349,27
95	OX	Swapping	20000	1880,77
95	UX	Inversion	10000	1568,52
60	PMX	SwapInvers	10000	1818,74
70	OX	Inversion	15000	1436,24
85	PMX	Swapping	10000	2941,77
80	OX	SwapInvers	15000	1340,59
70	OX	Inversion	20000	1217,90
85	UX	SwapInvers	10000	1468,09
95	UX	SwapInvers	5000	1482,47
65	OX	Swapping	15000	2104,58
95	UX	Swapping	15000	1592,35
60	UX	SwapInvers	10000	1490,16
85	PMX	SwapInvers	10000	1804,78
80	OX	Swapping	15000	1964,13
85	PMX	SwapInvers	15000	1467,07
60	PMX	Inversion	20000	1714,40
65	UX	Inversion	10000	1578,23
75	UX	Swapping	15000	1346,47
65	OX	Swapping	5000	2087,49
90	OX	Swapping	5000	1848,47
70	OX	Swapping	15000	1793,34
65	PMX	SwapInvers	20000	1410,23
80	UX	Swapping	5000	2242,69
95	PMX	Inversion	20000	1624,66
75	PMX	Inversion	15000	1597,76
60	UX	Swapping	10000	1252,41
70	PMX	Swapping	20000	2930,20
70	UX	SwapInvers	20000	1407,47
80	UX	SwapInvers	5000	1537,46
95	OX	Swapping	10000	1916,76
65	UX	Inversion	5000	1335,73
85	UX	Inversion	15000	1123,00
95	OX	Inversion	10000	1243,97
95	OX	Inversion	20000	1092,67
60	PMX	Inversion	5000	2003,32
85	PMX	SwapInvers	20000	1599,61
90	UX	Inversion	20000	981,59
70	PMX	SwapInvers	10000	2034,42

Taux de croisement	Croisement	Mutation	Génération	Distance
95	PMX	Swapping	10000	2916,01
60	PMX	Inversion	10000	1899,11
70	PMX	Swapping	15000	2904,16
95	PMX	Swapping	20000	2995,71
85	UX	SwapInvers	20000	1462,27
80	PMX	SwapInvers	10000	1639,22
90	OX	SwapInvers	20000	1255,13
90	OX	SwapInvers	10000	1232,85
65	PMX	Swapping	20000	2958,02
90	OX	Inversion	10000	1166,12
75	PMX	Inversion	20000	1883,91
80	OX	Inversion	5000	1288,68
70	PMX	Swapping	10000	2879,59
65	PMX	Inversion	10000	1611,24
80	OX	Inversion	10000	1213,92
75	PMX	Swapping	20000	2879,56
85	UX	Inversion	20000	1472,81
65	PMX	SwapInvers	5000	2130,66
90	OX	Inversion	20000	1093,74
60	PMX	Inversion	15000	1595,36
95	PMX	Inversion	5000	2139,07
75	UX	SwapInvers	10000	1452,42
70	PMX	SwapInvers	15000	1565,42
70	UX	Inversion	15000	1143,95
95	OX	Inversion	5000	1419,01
85	OX	Inversion	5000	1375,66
75	UX	SwapInvers	5000	1548,69
95	UX	Swapping	10000	1231,71
70	UX	SwapInvers	5000	1550,43
65	OX	Inversion	20000	1106,78
80	PMX	Inversion	10000	1672,14
85	PMX	Inversion	5000	2240,59
60	UX	Inversion	20000	1361,22
80	UX	SwapInvers	10000	1453,53
60	UX	Swapping	20000	1218,30
95	OX	Swapping	5000	1997,60
80	UX	SwapInvers	20000	1444,17
60	UX	SwapInvers	20000	1363,51
85	OX	Inversion	20000	1242,65
70	OX	SwapInvers	10000	1358,87
65	OX	Inversion	10000	1206,02
75	UX	Swapping	5000	1325,90
85	UX	Swapping	5000	1236,15
75	OX	Inversion	10000	1458,76
95	UX	Swapping	5000	1604,01
75	OX	Swapping	10000	2002,77
75	PMX	Swapping	5000	2964,14
70	OX	Swapping	5000	1643,77
60	OX	Inversion	15000	1045,08
85	UX	Swapping	10000	1127,75
90	PMX	Inversion	15000	1512,93
80	PMX	SwapInvers	20000	1317,99
75	UX	Inversion	10000	1531,92
60	OX	SwapInvers	15000	1174,54

Tab. C.1 – Le plan d'expériences de la phase Préalable pour le codage indirect type B205

C.1.1.2 Analyse des effets (avec l'outil Scaled Estimates)

Term	Prob>|t|
Taux de croisement[60]	0,6140
Taux de croisement[65]	0,9938
Taux de croisement[70]	0,6460
Taux de croisement[75]	0,8037
Taux de croisement[80]	0,8251
Taux de croisement[85]	0,5254
Taux de croisement[90]	0,7316
Taux de croisement[95]	0,6231
Croisement[PMX]	<,0001
Croisement[OX]	<,0001
Croisement[UX]	<,0001
Mutation[Inversion]	<,0001
Mutation[Swapping]	<,0001
Mutation[SwapInvers]	<,0001
Génération	<,0001
Taux de croisement[60]*Croisement[PMX]	0,7833
Taux de croisement[60]*Croisement[OX]	0,4108
Taux de croisement[60]*Croisement[UX]	0,2728
Taux de croisement[65]*Croisement[PMX]	0,2776
Taux de croisement[65]*Croisement[OX]	0,1738
Taux de croisement[65]*Croisement[UX]	0,7828
Taux de croisement[70]*Croisement[PMX]	0,3145
Taux de croisement[70]*Croisement[OX]	0,7193
Taux de croisement[70]*Croisement[UX]	0,5176
Taux de croisement[75]*Croisement[PMX]	0,4067
Taux de croisement[75]*Croisement[OX]	0,4392
Taux de croisement[75]*Croisement[UX]	0,9550
Taux de croisement[80]*Croisement[PMX]	0,2433
Taux de croisement[80]*Croisement[OX]	0,9557
Taux de croisement[80]*Croisement[UX]	0,2217
Taux de croisement[85]*Croisement[PMX]	0,9422
Taux de croisement[85]*Croisement[OX]	0,3726
Taux de croisement[85]*Croisement[UX]	0,4127
Taux de croisement[90]*Croisement[PMX]	0,9786
Taux de croisement[90]*Croisement[OX]	0,7711
Taux de croisement[90]*Croisement[UX]	0,7507
Taux de croisement[95]*Croisement[PMX]	0,4282
Taux de croisement[95]*Croisement[OX]	0,9619
Taux de croisement[95]*Croisement[UX]	0,4010
Taux de croisement[60]*Mutation[Inversion]	0,9245
Taux de croisement[60]*Mutation[Swapping]	0,8083
Taux de croisement[60]*Mutation[SwapInvers]	0,7359
Taux de croisement[65]*Mutation[Inversion]	0,2127
Taux de croisement[65]*Mutation[Swapping]	0,1036

Term	Prob>|t|
Taux de croisement[65]*Mutation[SwapInvers]	0,7012
Taux de croisement[70]*Mutation[Inversion]	0,6997
Taux de croisement[70]*Mutation[Swapping]	0,1773
Taux de croisement[70]*Mutation[SwapInvers]	0,3345
Taux de croisement[75]*Mutation[Inversion]	0,1423
Taux de croisement[75]*Mutation[Swapping]	0,1608
Taux de croisement[75]*Mutation[SwapInvers]	0,9479
Taux de croisement[80]*Mutation[Inversion]	0,9231
Taux de croisement[80]*Mutation[Swapping]	0,2611
Taux de croisement[80]*Mutation[SwapInvers]	0,3041
Taux de croisement[85]*Mutation[Inversion]	0,8376
Taux de croisement[85]*Mutation[Swapping]	0,5544
Taux de croisement[85]*Mutation[SwapInvers]	0,4261
Taux de croisement[90]*Mutation[Inversion]	0,5116
Taux de croisement[90]*Mutation[Swapping]	0,4441
Taux de croisement[90]*Mutation[SwapInvers]	0,9131
Taux de croisement[95]*Mutation[Inversion]	0,7986
Taux de croisement[95]*Mutation[Swapping]	0,6765
Taux de croisement[95]*Mutation[SwapInvers]	0,8712
Croisement[PMX]*Mutation[Inversion]	<,0001
Croisement[PMX]*Mutation[Swapping]	<,0001
Croisement[PMX]*Mutation[SwapInvers]	<,0001
Croisement[OX]*Mutation[Inversion]	0,9651
Croisement[OX]*Mutation[Swapping]	0,2448
Croisement[OX]*Mutation[SwapInvers]	0,2629
Croisement[UX]*Mutation[Inversion]	<,0001
Croisement[UX]*Mutation[Swapping]	<,0001
Croisement[UX]*Mutation[SwapInvers]	<,0001
Taux de croisement[60]*(Génération-12500)	0,1584
Taux de croisement[65]*(Génération-12500)	0,7088
Taux de croisement[70]*(Génération-12500)	0,5832
Taux de croisement[75]*(Génération-12500)	0,4623
Taux de croisement[80]*(Génération-12500)	0,8033
Taux de croisement[85]*(Génération-12500)	0,8916
Taux de croisement[90]*(Génération-12500)	0,7499
Taux de croisement[95]*(Génération-12500)	0,4316
Croisement[PMX]*(Génération-12500)	0,0473
Croisement[OX]*(Génération-12500)	0,1098
Croisement[UX]*(Génération-12500)	0,6979
Mutation[Inversion]*(Génération-12500)	0,8761
Mutation[Swapping]*(Génération-12500)	0,2728
Mutation[SwapInvers]*(Génération-12500)	0,3466

Tab. C.2 – Effets des facteurs de la phase préalable pour le codage indirect type B

C.1.1.3 Observations des estimations à l'aide de l'outil Prediction profiler

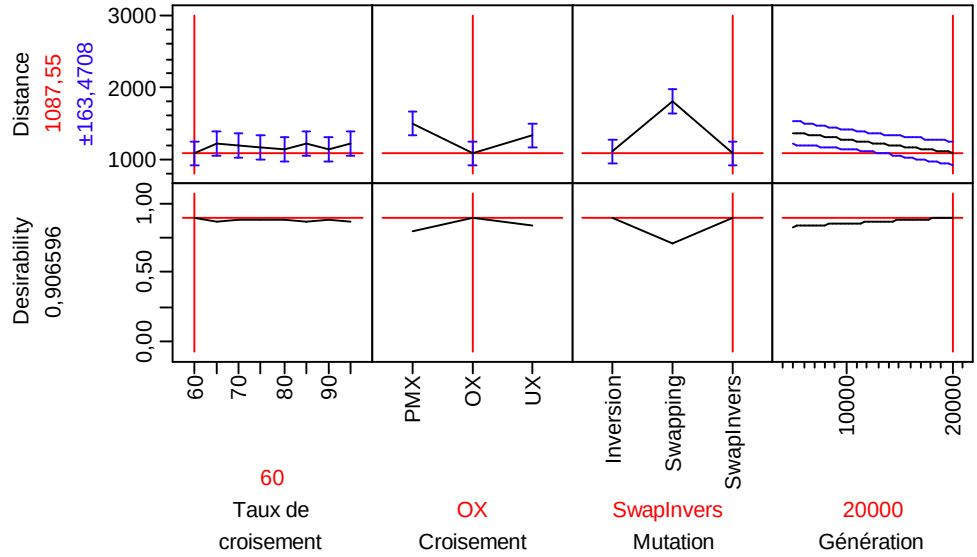


Fig. C.1 – Meilleur résultat de la phase préalable du croisement OX pour le codage indirect

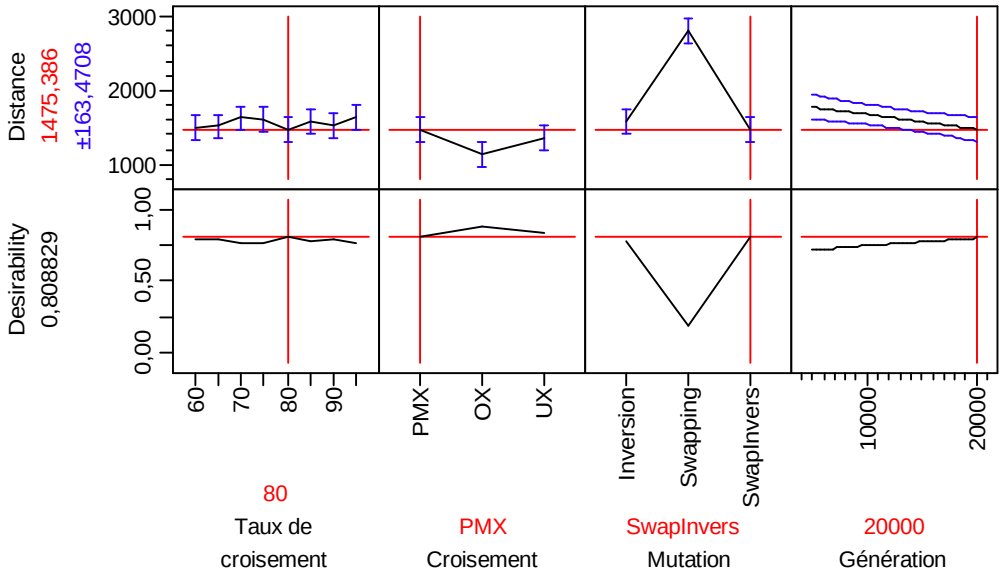


Fig. C.2 – Meilleur résultat de la phase Préalable du croisement PMX pour le codage indirect

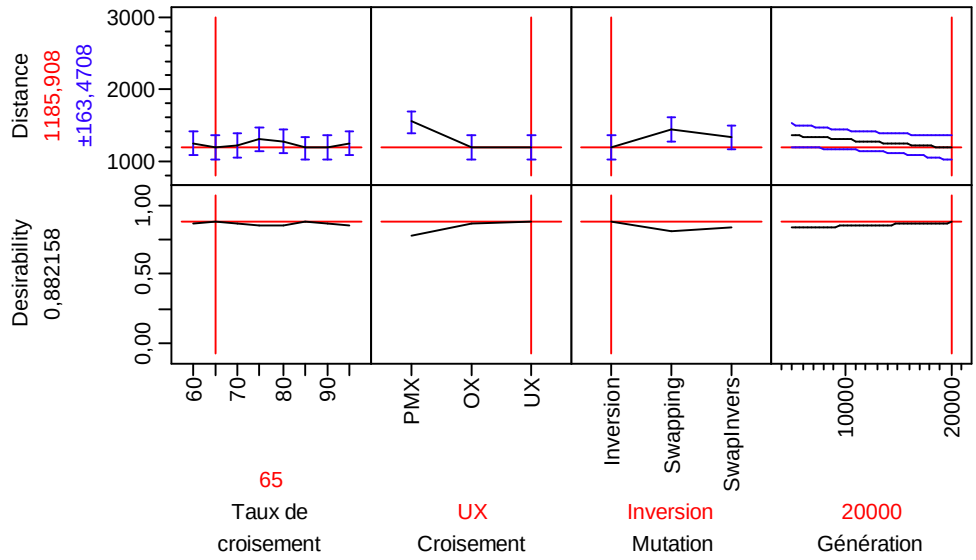


Fig. C.3 – Meilleur résultat de la phase Préalable du croisement UX pour le codage indirect

C.1.1.4 Observation des estimations à l'aide de l'outil Surface profiler

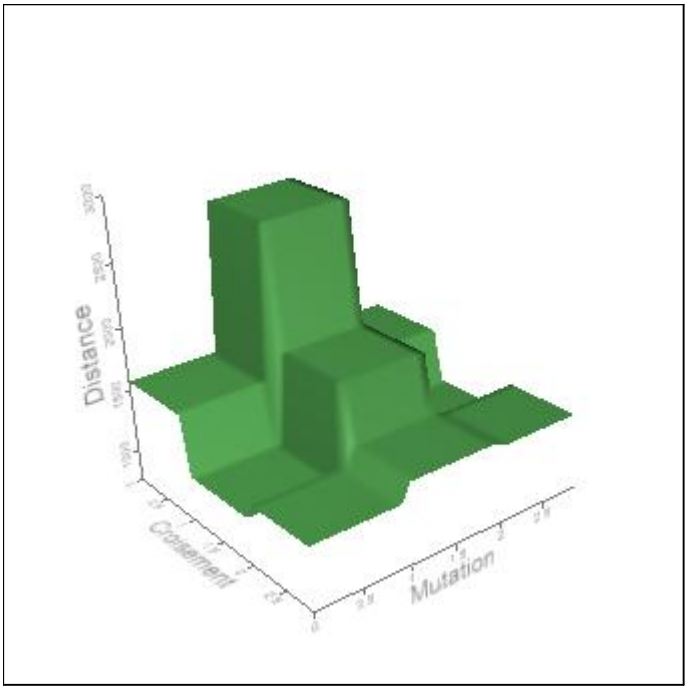


Fig. C.4 – Graphique 3D distance=f(croisement, mutation) de la phase Préalable pour le codage indirect

C.1.2 Résultats de la phase Préalable pour le codage direct

C.1.2.1 Le plan des essais

Ce plan contient 288 essais.

Taux de croisement	Croisement	Mutation	Génération	Distance
85	UX	Swapping	15000	1733
65	PMX	Swapping	10000	1544
60	PMX	Swapping	10000	1320
90	UX	SwapInvers	15000	1198
80	PMX	Swapping	5000	1204
80	PMX	Swapping	10000	1252
85	UX	Inversion	10000	1428
95	UX	Inversion	15000	1089
60	PMX	Swapping	5000	1516
75	OX	Inversion	15000	1167
90	OX	Swapping	15000	1089
70	PMX	SwapInvers	5000	1094
70	OX	Swapping	10000	1185
75	UX	Swapping	10000	1829
75	PMX	Swapping	15000	1605
70	PMX	Inversion	10000	1126
95	OX	SwapInvers	15000	1077
95	UX	SwapInvers	20000	1357
95	OX	Inversion	15000	1116
90	UX	Inversion	5000	1380
85	OX	Swapping	15000	1215
70	UX	Inversion	5000	1317
60	OX	SwapInvers	5000	1241
60	PMX	Swapping	15000	1372
65	PMX	Inversion	20000	1019
85	UX	Swapping	20000	1883
70	UX	Swapping	10000	2155
65	OX	Swapping	10000	1177
80	OX	Swapping	20000	1277
80	PMX	Swapping	15000	1456
80	UX	Inversion	10000	1370
60	OX	Inversion	20000	1242
85	PMX	Inversion	15000	991
65	UX	Swapping	10000	2017
85	OX	Inversion	15000	1002
95	OX	Swapping	15000	1093
65	OX	Swapping	20000	1049
65	UX	Swapping	5000	1914
70	OX	Inversion	10000	1281
70	UX	SwapInvers	10000	1308
90	OX	Swapping	10000	1194
90	UX	Inversion	15000	1139
85	PMX	SwapInvers	5000	1208
70	PMX	Swapping	5000	1514

Taux de croisement	Croisement	Mutation	Génération	Distance
80	UX	Inversion	15000	1132
60	PMX	SwapInvers	20000	1026
75	PMX	SwapInvers	10000	1157
80	UX	Swapping	20000	2054
80	PMX	Inversion	20000	1126
85	PMX	Inversion	10000	1004
65	UX	SwapInvers	10000	1328
60	PMX	Swapping	20000	1333
70	OX	Inversion	5000	1173
80	PMX	Inversion	15000	1016
90	UX	Swapping	5000	1634
70	UX	SwapInvers	15000	1189
95	UX	SwapInvers	10000	1292
95	PMX	Inversion	15000	982
95	OX	SwapInvers	5000	1187
85	OX	Swapping	5000	1134
65	OX	SwapInvers	15000	1221
90	OX	Inversion	15000	1242
75	PMX	Inversion	5000	971
75	PMX	SwapInvers	5000	1196
80	UX	Inversion	20000	1102
95	UX	Swapping	20000	2229
70	UX	Swapping	5000	1926
65	OX	Inversion	5000	1204
60	UX	Inversion	5000	1342
95	OX	SwapInvers	10000	1193
85	OX	Swapping	20000	1058
65	OX	Inversion	15000	1110
65	UX	SwapInvers	15000	1333
65	PMX	Swapping	5000	1561
85	OX	Inversion	10000	1204
90	PMX	Swapping	20000	1494
65	OX	SwapInvers	10000	1177
60	OX	Swapping	20000	1122
70	UX	Swapping	20000	1862
70	OX	SwapInvers	15000	1130
90	UX	Swapping	20000	1634
65	UX	Inversion	15000	1368
60	UX	Inversion	10000	1209
90	PMX	SwapInvers	15000	1105
90	UX	Inversion	10000	1129
80	OX	SwapInvers	10000	1136
75	OX	SwapInvers	10000	1194
95	PMX	SwapInvers	5000	1217

Taux de croisement	Croisement	Mutation	Génération	Distance
85	PMX	Swapping	5000	1529
70	OX	Swapping	20000	1153
85	OX	SwapInvers	5000	1185
85	UX	Inversion	5000	1451
90	OX	SwapInvers	5000	1232
85	UX	SwapInvers	15000	1166
60	OX	Inversion	5000	1060
75	OX	SwapInvers	15000	1108
85	OX	SwapInvers	10000	1108
85	PMX	Inversion	20000	1027
60	PMX	SwapInvers	15000	985
75	UX	Swapping	20000	1911
65	UX	Swapping	20000	1221
60	OX	Swapping	10000	1158
80	UX	Swapping	10000	2485
95	PMX	Swapping	5000	1367
70	PMX	Inversion	20000	1088
80	UX	Inversion	5000	1344
70	PMX	SwapInvers	20000	967
65	PMX	SwapInvers	15000	972
60	UX	Inversion	15000	1182
65	PMX	Inversion	5000	1080
80	PMX	Inversion	5000	1026
90	PMX	SwapInvers	20000	963
90	PMX	Swapping	10000	1599
75	OX	Swapping	15000	1156
90	OX	Swapping	20000	1216
90	PMX	Inversion	10000	981
95	PMX	Swapping	15000	1401
70	PMX	Inversion	15000	1132
85	OX	SwapInvers	20000	1270
90	PMX	Inversion	5000	1054
90	UX	Swapping	10000	1994
75	UX	SwapInvers	20000	1142
65	OX	SwapInvers	5000	1150
80	PMX	SwapInvers	5000	1120
65	PMX	SwapInvers	10000	969
90	OX	SwapInvers	15000	1067
60	UX	SwapInvers	5000	1247
60	UX	Swapping	5000	1661
75	UX	Inversion	5000	1267
75	OX	Inversion	20000	1180
75	PMX	Swapping	10000	1510
80	OX	Inversion	15000	1015
85	UX	SwapInvers	5000	1343
80	PMX	Swapping	20000	1570
80	OX	Swapping	5000	1243
75	OX	SwapInvers	5000	1215
60	OX	Inversion	10000	1107
60	OX	SwapInvers	10000	1202
95	UX	SwapInvers	15000	1184
90	PMX	Swapping	5000	1642
95	UX	Inversion	20000	1067
85	PMX	Swapping	20000	1504
70	PMX	Inversion	5000	1102
75	UX	Inversion	15000	1165
65	UX	SwapInvers	20000	1123
90	UX	SwapInvers	20000	1264

Taux de croisement	Croisement	Mutation	Génération	Distance
75	OX	Swapping	20000	1124
70	UX	Inversion	10000	1382
95	PMX	SwapInvers	15000	1052
95	PMX	SwapInvers	10000	1004
60	UX	Swapping	15000	1729
85	OX	SwapInvers	15000	1139
75	OX	Swapping	5000	1244
85	PMX	Swapping	15000	1233
80	UX	SwapInvers	15000	1248
75	PMX	Inversion	10000	1007
60	OX	Swapping	15000	1259
80	OX	SwapInvers	20000	1158
75	PMX	SwapInvers	15000	1137
95	PMX	SwapInvers	20000	1267
75	UX	SwapInvers	15000	1299
75	UX	Inversion	20000	1237
80	OX	SwapInvers	5000	1225
75	PMX	SwapInvers	20000	1059
60	OX	Swapping	5000	1386
70	OX	SwapInvers	20000	1180
90	UX	SwapInvers	5000	1437
90	UX	Swapping	15000	2089
60	PMX	SwapInvers	5000	1108
75	OX	SwapInvers	20000	1317
80	OX	Swapping	10000	1262
90	UX	SwapInvers	10000	1248
70	OX	SwapInvers	5000	1160
90	PMX	SwapInvers	10000	927
95	OX	SwapInvers	20000	1133
65	UX	SwapInvers	5000	1441
90	PMX	Swapping	15000	1464
90	OX	Inversion	5000	1223
95	PMX	Inversion	10000	947
70	UX	Swapping	15000	1336
65	PMX	Inversion	15000	1022
60	UX	SwapInvers	15000	1111
65	UX	Inversion	20000	1193
80	UX	Swapping	15000	1680
90	PMX	SwapInvers	5000	1035
95	UX	Inversion	5000	1373
80	PMX	SwapInvers	15000	1084
90	PMX	Inversion	20000	1013
65	OX	SwapInvers	20000	1295
75	OX	Inversion	5000	1171
85	OX	Swapping	10000	1349
65	PMX	Swapping	15000	1513
80	OX	Inversion	20000	1196
70	UX	Inversion	20000	1159
65	UX	Swapping	15000	1745
60	OX	SwapInvers	20000	1035
95	OX	Swapping	20000	1013
95	UX	Inversion	10000	1350
60	PMX	SwapInvers	10000	1083
70	OX	Inversion	15000	1061
85	PMX	Swapping	10000	1358
80	OX	SwapInvers	15000	1296
70	OX	Inversion	20000	1139
85	UX	SwapInvers	10000	1375

Taux de croisement	Croisement	Mutation	Génération	Distance
95	UX	SwapInvers	5000	1435
65	OX	Swapping	15000	1253
95	UX	Swapping	15000	1800
60	UX	SwapInvers	10000	1373
85	PMX	SwapInvers	10000	1026
80	OX	Swapping	15000	1213
85	PMX	SwapInvers	15000	1038
60	PMX	Inversion	20000	1036
65	UX	Inversion	10000	1389
75	UX	Swapping	15000	1887
65	OX	Swapping	5000	1599
90	OX	Swapping	5000	1151
70	OX	Swapping	15000	1239
65	PMX	SwapInvers	20000	1010
80	UX	Swapping	5000	1555
95	PMX	Inversion	20000	1096
75	PMX	Inversion	15000	1002
60	UX	Swapping	10000	1851
70	PMX	Swapping	20000	1460
70	UX	SwapInvers	20000	1063
80	UX	SwapInvers	5000	1254
95	OX	Swapping	10000	1055
65	UX	Inversion	5000	1333
85	UX	Inversion	15000	1176
95	OX	Inversion	10000	1081
95	OX	Inversion	20000	1143
60	PMX	Inversion	5000	1004
85	PMX	SwapInvers	20000	1012
90	UX	Inversion	20000	1351
70	PMX	SwapInvers	10000	1181
95	PMX	Swapping	10000	1533
60	PMX	Inversion	10000	968
70	PMX	Swapping	15000	1333
95	PMX	Swapping	20000	1345
85	UX	SwapInvers	20000	1024
80	PMX	SwapInvers	10000	1053
90	OX	SwapInvers	20000	1044
90	OX	SwapInvers	10000	1119
65	PMX	Swapping	20000	1781
90	OX	Inversion	10000	1303
75	PMX	Inversion	20000	1160
80	OX	Inversion	5000	1174
70	PMX	Swapping	10000	1400

Taux de croisement	Croisement	Mutation	Génération	Distance
65	PMX	Inversion	10000	1043
80	OX	Inversion	10000	1141
75	PMX	Swapping	20000	1709
85	UX	Inversion	20000	1167
65	PMX	SwapInvers	5000	1059
90	OX	Inversion	20000	1072
60	PMX	Inversion	15000	1110
95	PMX	Inversion	5000	1059
75	UX	SwapInvers	10000	1069
70	PMX	SwapInvers	15000	1012
70	UX	Inversion	15000	1344
95	OX	Inversion	5000	1068
85	OX	Inversion	5000	1241
75	UX	SwapInvers	5000	1560
95	UX	Swapping	10000	1303
70	UX	SwapInvers	5000	1355
65	OX	Inversion	20000	1103
80	PMX	Inversion	10000	1187
85	PMX	Inversion	5000	1153
60	UX	Inversion	20000	1302
80	UX	SwapInvers	10000	1325
60	UX	Swapping	20000	1828
95	OX	Swapping	5000	1066
80	UX	SwapInvers	20000	1173
60	UX	SwapInvers	20000	1354
85	OX	Inversion	20000	1143
70	OX	SwapInvers	10000	1139
65	OX	Inversion	10000	1199
75	UX	Swapping	5000	1771
85	UX	Swapping	5000	1814
75	OX	Inversion	10000	1135
95	UX	Swapping	5000	2101
75	OX	Swapping	10000	1133
75	PMX	Swapping	5000	1324
70	OX	Swapping	5000	1275
60	OX	Inversion	15000	1115
85	UX	Swapping	10000	1698
90	PMX	Inversion	15000	1132
80	PMX	SwapInvers	20000	1074
75	UX	Inversion	10000	1225
60	OX	SwapInvers	15000	1136

Tab. C.3 – Le plan d'expériences de la phase Préalable pour le codage direct (288 essais)

C.1.2.2 Analyse des effets (avec l'outil Scaled Estimates)

Term	Prob>|t|
Taux de croisement[60]	0,3398
Taux de croisement[65]	0,3273
Taux de croisement[70]	0,8737
Taux de croisement[75]	0,4577
Taux de croisement[80]	0,5610
Taux de croisement[85]	0,5659
Taux de croisement[90]	0,9415
Taux de croisement[95]	0,3140
Croisement[PMX]	<,0001
Croisement[OX]	<,0001
Croisement[UX]	<,0001
Mutation[Inversion]	<,0001
Mutation[Swapping]	<,0001
Mutation[SwapInvers]	<,0001
Génération	0,0020
Taux de croisement[60]*Croisement[PMX]	0,4921
Taux de croisement[60]*Croisement[OX]	0,4802
Taux de croisement[60]*Croisement[UX]	0,9848
Taux de croisement[65]*Croisement[PMX]	0,9765
Taux de croisement[65]*Croisement[OX]	0,4681
Taux de croisement[65]*Croisement[UX]	0,4503
Taux de croisement[70]*Croisement[PMX]	0,8966
Taux de croisement[70]*Croisement[OX]	0,9437
Taux de croisement[70]*Croisement[UX]	0,8410
Taux de croisement[75]*Croisement[PMX]	0,3322
Taux de croisement[75]*Croisement[OX]	0,7969
Taux de croisement[75]*Croisement[UX]	0,4759
Taux de croisement[80]*Croisement[PMX]	0,3821
Taux de croisement[80]*Croisement[OX]	0,6751
Taux de croisement[80]*Croisement[UX]	0,6488
Taux de croisement[85]*Croisement[PMX]	0,7592
Taux de croisement[85]*Croisement[OX]	0,6944
Taux de croisement[85]*Croisement[UX]	0,9311
Taux de croisement[90]*Croisement[PMX]	0,8489
Taux de croisement[90]*Croisement[OX]	0,7292
Taux de croisement[90]*Croisement[UX]	0,8762
Taux de croisement[95]*Croisement[PMX]	0,5839
Taux de croisement[95]*Croisement[OX]	0,0880
Taux de croisement[95]*Croisement[UX]	0,2453
Taux de croisement[60]*Mutation[Inversion]	0,9056
Taux de croisement[60]*Mutation[Swapping]	0,6898
Taux de croisement[60]*Mutation[SwapInvers]	0,7791
Taux de croisement[65]*Mutation[Inversion]	0,9048
Taux de croisement[65]*Mutation[Swapping]	0,4906

Term	Prob>|t|
Taux de croisement[65]*Mutation[SwapInvers]	0,5688
Taux de croisement[70]*Mutation[Inversion]	0,2437
Taux de croisement[70]*Mutation[Swapping]	0,7645
Taux de croisement[70]*Mutation[SwapInvers]	0,3859
Taux de croisement[75]*Mutation[Inversion]	0,2928
Taux de croisement[75]*Mutation[Swapping]	0,7214
Taux de croisement[75]*Mutation[SwapInvers]	0,4864
Taux de croisement[80]*Mutation[Inversion]	0,6011
Taux de croisement[80]*Mutation[Swapping]	0,5410
Taux de croisement[80]*Mutation[SwapInvers]	0,9294
Taux de croisement[85]*Mutation[Inversion]	0,4492
Taux de croisement[85]*Mutation[Swapping]	0,4549
Taux de croisement[85]*Mutation[SwapInvers]	0,9924
Taux de croisement[90]*Mutation[Inversion]	0,6943
Taux de croisement[90]*Mutation[Swapping]	0,4117
Taux de croisement[90]*Mutation[SwapInvers]	0,2253
Taux de croisement[95]*Mutation[Inversion]	0,4592
Taux de croisement[95]*Mutation[Swapping]	0,3022
Taux de croisement[95]*Mutation[SwapInvers]	0,0771
Croisement[PMX]*Mutation[Inversion]	0,0999
Croisement[PMX]*Mutation[Swapping]	0,0020
Croisement[PMX]*Mutation[SwapInvers]	0,1416
Croisement[OX]*Mutation[Inversion]	<,0001
Croisement[OX]*Mutation[Swapping]	<,0001
Croisement[OX]*Mutation[SwapInvers]	<,0001
Croisement[UX]*Mutation[Inversion]	<,0001
Croisement[UX]*Mutation[Swapping]	<,0001
Croisement[UX]*Mutation[SwapInvers]	<,0001
Taux de croisement[60]*(Génération-12500)	0,6322
Taux de croisement[65]*(Génération-12500)	0,0615
Taux de croisement[70]*(Génération-12500)	0,2157
Taux de croisement[75]*(Génération-12500)	0,1184
Taux de croisement[80]*(Génération-12500)	0,1104
Taux de croisement[85]*(Génération-12500)	0,2516
Taux de croisement[90]*(Génération-12500)	0,8667
Taux de croisement[95]*(Génération-12500)	0,4313
Croisement[PMX]*(Génération-12500)	0,0296
Croisement[OX]*(Génération-12500)	0,8828
Croisement[UX]*(Génération-12500)	0,0203
Mutation[Inversion]*(Génération-12500)	0,9740
Mutation[Swapping]*(Génération-12500)	0,2398
Mutation[SwapInvers]*(Génération-12500)	0,2270

Tab. C.4 – Effets des facteurs de la phase Préalable pour le codage direct

C.1.2.3 Observation des estimations avec l'outil Prediction profiler

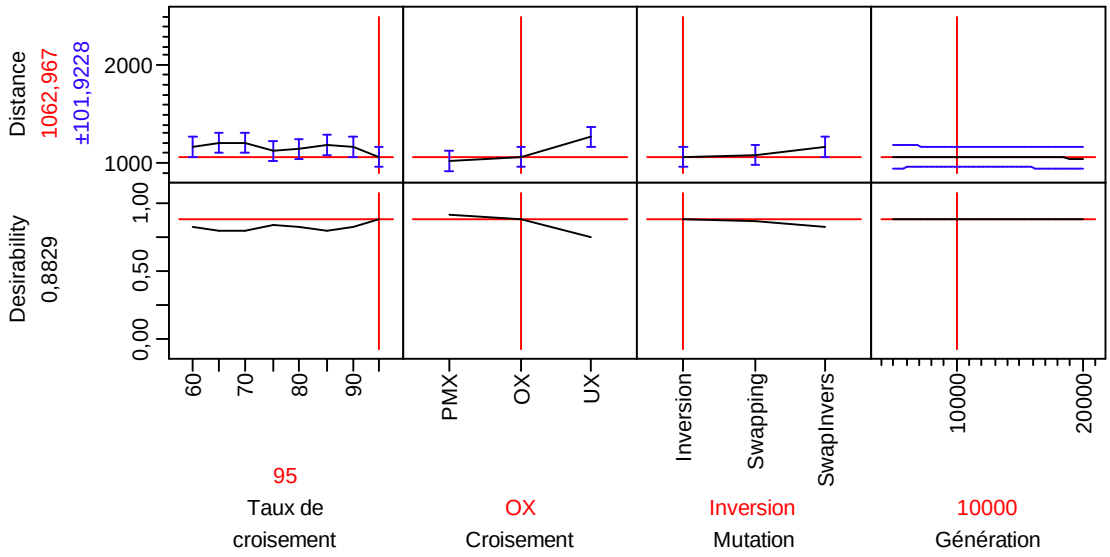


Fig. C.5 – Meilleur résultat de la phase Préalable du croisement OX pour le codage direct

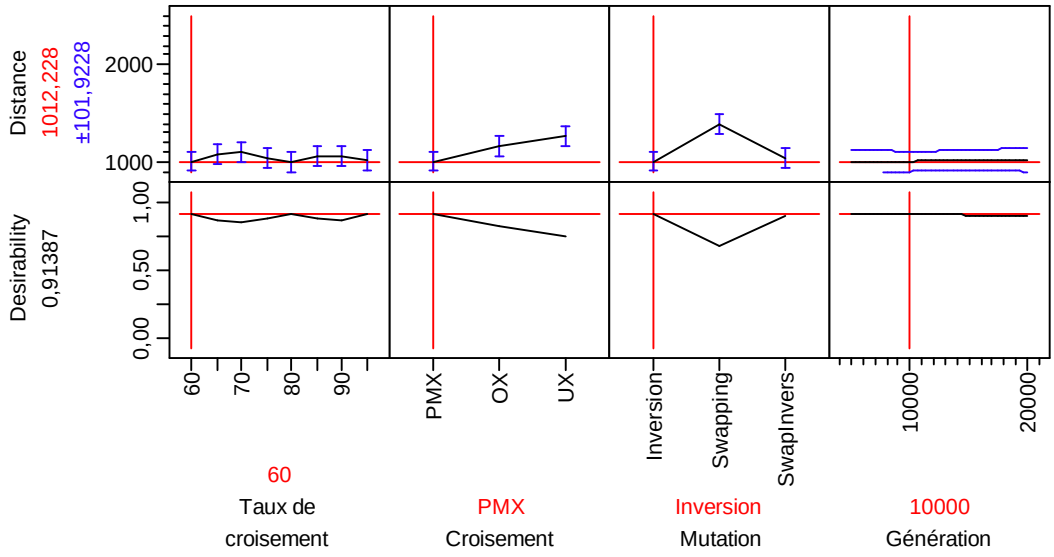


Fig. C.6 – Meilleur résultat phase Préalable du croisement PMX pour le codage direct

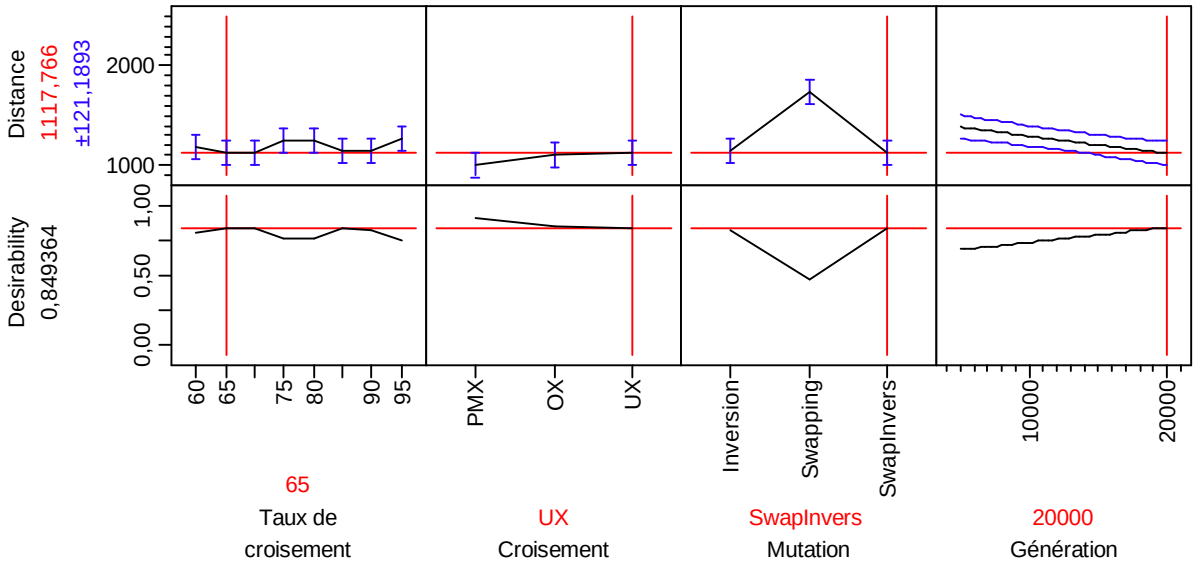


Fig. C.7 – Meilleur résultat phase Préalable du croisement UX pour le codage direct

C.1.2.4 Observation des estimations à l'aide de l'outil Surface profiler

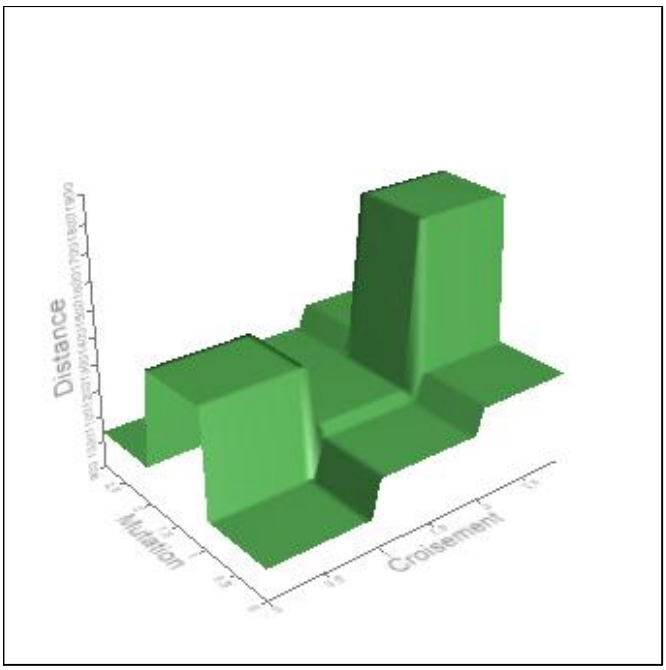


Fig. C.8 – Graphique 3D distance=f(croisement, mutation) de la phase Préalable pour le codage direct

C.2 Résultats de la phase Réglage

C.2.1 Résultats de la phase Réglage pour le codage indirect

C.2.1.1 Le plan des essais

160 essais supplémentaires sont ajoutés aux 288 essais de la phase Préalable. Ces essais correspondent à 10 exécutions de l'algorithme pour les 16 essais jugés les plus intéressants lors de la phase Préalable.

Taux de croisement	Croisement	Mutation	Génération	Distance
60	OX	Inversion	20000	1175,40
60	OX	Inversion	20000	1137,62
60	OX	Inversion	20000	1074,98
60	OX	Inversion	20000	1103,10
60	OX	Inversion	20000	1227,36
60	OX	Inversion	20000	1098,11
60	OX	Inversion	20000	1286,86
60	OX	Inversion	20000	1119,56
60	OX	Inversion	20000	1285,46
60	OX	Inversion	20000	1160,23
80	OX	Inversion	20000	1029,81
80	OX	Inversion	20000	1155,25
80	OX	Inversion	20000	1241,07
80	OX	Inversion	20000	1325,85
80	OX	Inversion	20000	1129,70
80	OX	Inversion	20000	1253,91
80	OX	Inversion	20000	1319,79
80	OX	Inversion	20000	1332,05
80	OX	Inversion	20000	1137,74
80	OX	Inversion	20000	1070,87
90	OX	Inversion	20000	1230,00
90	OX	Inversion	20000	1134,63
90	OX	Inversion	20000	1227,67
90	OX	Inversion	20000	1176,32
90	OX	Inversion	20000	1272,22
90	OX	Inversion	20000	1119,06
90	OX	Inversion	20000	1192,45
90	OX	Inversion	20000	1339,63
90	OX	Inversion	20000	1097,05
90	OX	Inversion	20000	1093,01
75	OX	Swapping	20000	1936,74
75	OX	Swapping	20000	1779,96
75	OX	Swapping	20000	1951,45
75	OX	Swapping	20000	1740,96
75	OX	Swapping	20000	1975,54
75	OX	Swapping	20000	1777,19
75	OX	Swapping	20000	2046,65
75	OX	Swapping	20000	1708,28
75	OX	Swapping	20000	1823,83
75	OX	Swapping	20000	1913,99
60	OX	SwapInvers	20000	1076,34
60	OX	SwapInvers	20000	1146,54
60	OX	SwapInvers	20000	1242,20
60	OX	SwapInvers	20000	1166,09
60	OX	SwapInvers	20000	1136,34
60	OX	SwapInvers	20000	1070,36
60	OX	SwapInvers	20000	1217,32
60	OX	SwapInvers	20000	1156,97
60	OX	SwapInvers	20000	1249,84
60	OX	SwapInvers	20000	1139,76
80	OX	SwapInvers	20000	2076,23
80	OX	SwapInvers	20000	2105,94
80	OX	SwapInvers	20000	2107,56
80	OX	SwapInvers	20000	2102,33
80	OX	SwapInvers	20000	2099,63
80	OX	SwapInvers	20000	2098,02
80	OX	SwapInvers	20000	2081,01
80	OX	SwapInvers	20000	2185,23
80	OX	SwapInvers	20000	2207,95
80	OX	SwapInvers	20000	2016,03
90	OX	SwapInvers	20000	2062,61
90	OX	SwapInvers	20000	2167,14
90	OX	SwapInvers	20000	2105,61
90	OX	SwapInvers	20000	1952,61
90	OX	SwapInvers	20000	2041,41
90	OX	SwapInvers	20000	2081,42
90	OX	SwapInvers	20000	2083,25
90	OX	SwapInvers	20000	2107,86
90	OX	SwapInvers	20000	2143,39
90	OX	SwapInvers	20000	2087,17
85	UX	Inversion	15000	1313,82
85	UX	Inversion	15000	1085,75
85	UX	Inversion	15000	1152,57
85	UX	Inversion	15000	1067,37
85	UX	Inversion	15000	1328,82
85	UX	Inversion	15000	1517,88
85	UX	Inversion	15000	1192,58
85	UX	Inversion	15000	1114,97
85	UX	Inversion	15000	1337,42
85	UX	Inversion	15000	1225,32
65	UX	Inversion	20000	1465,97
65	UX	Inversion	20000	1426,55
65	UX	Inversion	20000	1077,74
65	UX	Inversion	20000	1375,32
65	UX	Inversion	20000	1510,38
65	UX	Inversion	20000	1158,12
65	UX	Inversion	20000	1120,36
65	UX	Inversion	20000	1302,03
65	UX	Inversion	20000	1110,40
65	UX	Inversion	20000	1271,56
90	UX	Inversion	20000	1430,02
90	UX	Inversion	20000	1051,80
90	UX	Inversion	20000	1617,38
90	UX	Inversion	20000	1314,51
90	UX	Inversion	20000	1310,40
90	UX	Inversion	20000	1466,50

90	UX	Inversion	20000	1106,59
90	UX	Inversion	20000	1496,26
90	UX	Inversion	20000	1060,64
90	UX	Inversion	20000	1294,58
85	UX	Swapping	15000	1384,86
85	UX	Swapping	15000	1319,82
85	UX	Swapping	15000	1156,39
85	UX	Swapping	15000	1240,70
85	UX	Swapping	15000	1155,49
85	UX	Swapping	15000	1178,75
85	UX	Swapping	15000	1266,42
85	UX	Swapping	15000	2137,58
85	UX	Swapping	15000	1657,16
85	UX	Swapping	15000	1262,75
85	UX	SwapInvers	15000	1413,84
85	UX	SwapInvers	15000	1324,88
85	UX	SwapInvers	15000	1388,89
85	UX	SwapInvers	15000	1460,97
85	UX	SwapInvers	15000	1244,58
85	UX	SwapInvers	15000	1303,85
85	UX	SwapInvers	15000	1436,90
85	UX	SwapInvers	15000	1421,28
85	UX	SwapInvers	15000	1357,64
85	UX	SwapInvers	15000	1282,68
90	UX	SwapInvers	15000	1565,04
90	UX	SwapInvers	15000	1099,42
90	UX	SwapInvers	15000	1477,59
90	UX	SwapInvers	15000	1195,63
90	UX	SwapInvers	15000	1350,93
90	UX	SwapInvers	15000	1367,10
90	UX	SwapInvers	15000	1265,91
90	UX	SwapInvers	15000	1576,84
90	UX	SwapInvers	15000	1295,72
90	UX	SwapInvers	15000	1514,66
80	PMX	Inversion	20000	1496,87
80	PMX	Inversion	20000	1519,32
80	PMX	Inversion	20000	1417,39
80	PMX	Inversion	20000	1356,21
80	PMX	Inversion	20000	1699,51
80	PMX	Inversion	20000	1664,13
80	PMX	Inversion	20000	1718,10
80	PMX	Inversion	20000	1556,90
80	PMX	Inversion	20000	1519,16
80	PMX	Inversion	20000	1673,03
80	PMX	Swapping	20000	2932,81
80	PMX	Swapping	20000	2913,02
80	PMX	Swapping	20000	2815,74
80	PMX	Swapping	20000	2970,07
80	PMX	Swapping	20000	2961,77
80	PMX	Swapping	20000	2912,70
80	PMX	Swapping	20000	2885,93
80	PMX	Swapping	20000	2949,53
80	PMX	Swapping	20000	2928,38
80	PMX	Swapping	20000	2948,75
80	PMX	SwapInvers	20000	1569,82
80	PMX	SwapInvers	20000	1481,98
80	PMX	SwapInvers	20000	1671,01
80	PMX	SwapInvers	20000	1549,04
80	PMX	SwapInvers	20000	1492,88
80	PMX	SwapInvers	20000	1294,59
80	PMX	SwapInvers	20000	1560,88
80	PMX	SwapInvers	20000	1379,24
80	PMX	SwapInvers	20000	1749,89
80	PMX	SwapInvers	20000	1494,91

Tab. C.5 – Le plan d'expériences de la phase Réglage pour le codage indirect (448 essais)

C.2.1.2 Analyse des effets (avec l'outil Scaled Estimates)

Term	Prob>|t|
Taux de croisement[60]	0,3115
Taux de croisement[65]	0,8300
Taux de croisement[70]	0,5856
Taux de croisement[75]	0,6246
Taux de croisement[80]	0,0119
Taux de croisement[85]	0,1671
Taux de croisement[90]	0,0666
Taux de croisement[95]	0,9310
Croisement[PMX]	<,0001
Croisement[OX]	<,0001
Croisement[UX]	<,0001
Mutation[Inversion]	<,0001
Mutation[Swapping]	<,0001
Mutation[SwapInvers]	<,0001
Génération	<,0001
Taux de croisement[60]*Croisement[PMX]	0,8466
Taux de croisement[60]*Croisement[OX]	0,0773
Taux de croisement[60]*Croisement[UX]	0,1647
Taux de croisement[65]*Croisement[PMX]	0,4313
Taux de croisement[65]*Croisement[OX]	0,8025
Taux de croisement[65]*Croisement[UX]	0,5629
Taux de croisement[70]*Croisement[PMX]	0,2114
Taux de croisement[70]*Croisement[OX]	0,2985
Taux de croisement[70]*Croisement[UX]	0,8296
Taux de croisement[75]*Croisement[PMX]	0,3210
Taux de croisement[75]*Croisement[OX]	0,2115
Taux de croisement[75]*Croisement[UX]	0,8593
Taux de croisement[80]*Croisement[PMX]	0,0028
Taux de croisement[80]*Croisement[OX]	0,0037
Taux de croisement[80]*Croisement[UX]	0,9575
Taux de croisement[85]*Croisement[PMX]	0,7404
Taux de croisement[85]*Croisement[OX]	0,9619
Taux de croisement[85]*Croisement[UX]	0,6411
Taux de croisement[90]*Croisement[PMX]	0,4405

Term	Prob>|t|
Taux de croisement[90]*Croisement[OX]	0,0163
Taux de croisement[90]*Croisement[UX]	0,1338
Taux de croisement[95]*Croisement[PMX]	0,2870
Taux de croisement[95]*Croisement[OX]	0,4925
Taux de croisement[95]*Croisement[UX]	0,7018
Taux de croisement[60]*Mutation[Inversion]	0,0389
Taux de croisement[60]*Mutation[Swapping]	0,7055
Taux de croisement[60]*Mutation[SwapInvers]	0,0125
Taux de croisement[65]*Mutation[Inversion]	0,7387
Taux de croisement[65]*Mutation[Swapping]	0,1796
Taux de croisement[65]*Mutation[SwapInvers]	0,2991
Taux de croisement[70]*Mutation[Inversion]	0,4863
Taux de croisement[70]*Mutation[Swapping]	0,2875
Taux de croisement[70]*Mutation[SwapInvers]	0,7101
Taux de croisement[75]*Mutation[Inversion]	0,1223
Taux de croisement[75]*Mutation[Swapping]	0,3495
Taux de croisement[75]*Mutation[SwapInvers]	0,5068
Taux de croisement[80]*Mutation[Inversion]	0,0018
Taux de croisement[80]*Mutation[Swapping]	0,2670
Taux de croisement[80]*Mutation[SwapInvers]	0,0577
Taux de croisement[85]*Mutation[Inversion]	0,7824
Taux de croisement[85]*Mutation[Swapping]	0,9184
Taux de croisement[85]*Mutation[SwapInvers]	0,7048
Taux de croisement[90]*Mutation[Inversion]	0,0252
Taux de croisement[90]*Mutation[Swapping]	0,7393
Taux de croisement[90]*Mutation[SwapInvers]	0,0068

Term	Prob>|t|
Taux de croisement[95]*Mutation[Inversion]	0,5602
Taux de croisement[95]*Mutation[Swapping]	0,8003
Taux de croisement[95]*Mutation[SwapInvers]	0,7423
Croisement[PMX]*Mutation[Inversion]	<,0001
Croisement[PMX]*Mutation[Swapping]	<,0001
Croisement[PMX]*Mutation[SwapInvers]	<,0001
Croisement[OX]*Mutation[Inversion]	<,0001
Croisement[OX]*Mutation[Swapping]	0,5736
Croisement[OX]*Mutation[SwapInvers]	<,0001
Croisement[UX]*Mutation[Inversion]	<,0001
Croisement[UX]*Mutation[Swapping]	<,0001
Croisement[UX]*Mutation[SwapInvers]	<,0001
Taux de croisement[60]*(Génération-14732,1)	0,0190
Taux de croisement[65]*(Génération-14732,1)	0,9595
Taux de croisement[70]*(Génération-14732,1)	0,2115
Taux de croisement[75]*(Génération-14732,1)	0,9125
Taux de croisement[80]*(Génération-14732,1)	0,0109
Taux de croisement[85]*(Génération-14732,1)	0,5053
Taux de croisement[90]*(Génération-14732,1)	0,0178
Taux de croisement[95]*(Génération-14732,1)	0,9293
Croisement[PMX]*(Génération-14732,1)	0,0004
Croisement[OX]*(Génération-14732,1)	<,0001
Croisement[UX]*(Génération-14732,1)	0,6398
Mutation[Inversion]*(Génération-14732,1)	0,0806
Mutation[Swapping]*(Génération-14732,1)	0,7940
Mutation[SwapInvers]*(Génération-14732,1)	0,1491

Tab. C.6 – Effets des facteurs de la phase Réglage pour le codage indirect

C.2.1.3 Observation des estimations à l'aide de l'outil Surface profiler

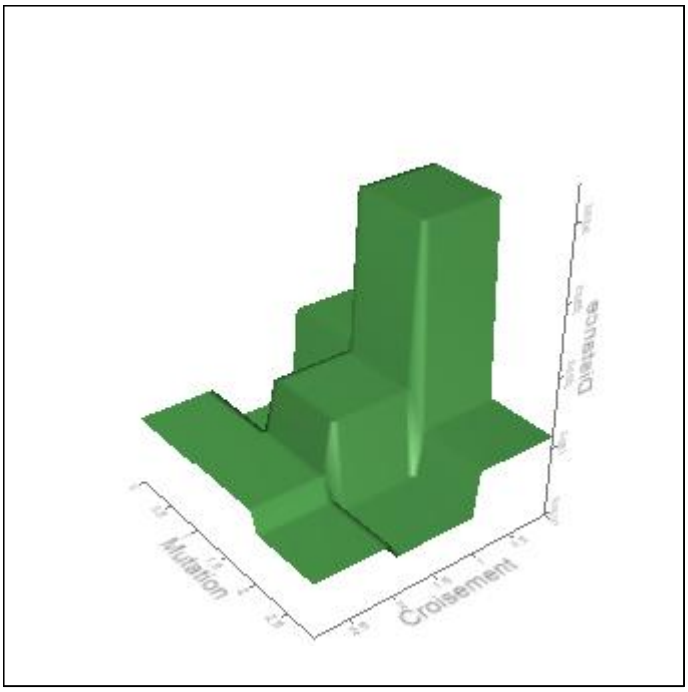


Fig. C.9 – Graphique 3D distance=f(croisement, mutation) de la phase Réglage pour le codage indirect

C.2.2 Résultats de la phase Réglage pour le codage direct

C.2.2.1 Le plan des essais

Au plus des 288 essais de l'analyse élémentaire, nous ajoutons les essais suivant qui sont les 16 meilleures essais produits de l'analyse élémentaire. Chaque essai a été fait 10 fois.

Taux de croisement	Croisement	Mutation	Génération	Distance
60	PMX	Inversion	10000	970,40
60	PMX	Inversion	10000	984,74
60	PMX	Inversion	10000	1004,10
60	PMX	Inversion	10000	1014,20
60	PMX	Inversion	10000	1021,50
60	PMX	Inversion	10000	1028,80
60	PMX	Inversion	10000	1040,40
60	PMX	Inversion	10000	1046,50
60	PMX	Inversion	10000	1066,80
95	PMX	Inversion	10000	965,21
95	PMX	Inversion	10000	985,34
95	PMX	Inversion	10000	989,86
95	PMX	Inversion	10000	1045,90
95	PMX	Inversion	10000	1053,60
95	PMX	Inversion	10000	1072,50
95	PMX	Inversion	10000	1097,40
95	PMX	Inversion	10000	1103,90
95	PMX	Inversion	10000	1121,00
85	PMX	Inversion	15000	978,10
85	PMX	Inversion	15000	999,40
85	PMX	Inversion	15000	1010,60
85	PMX	Inversion	15000	1022,78
85	PMX	Inversion	15000	1028,15
85	PMX	Inversion	15000	1032,05
85	PMX	Inversion	15000	1050,83
85	PMX	Inversion	15000	1068,79
85	PMX	Inversion	15000	1080,35
85	PMX	Inversion	15000	1085,55
95	PMX	Inversion	15000	973,71
95	PMX	Inversion	15000	978,56
95	PMX	Inversion	15000	1020,82
95	PMX	Inversion	15000	1039,43
95	PMX	Inversion	15000	1041,61
95	PMX	Inversion	15000	1085,80
95	PMX	Inversion	15000	1095,83
95	PMX	Inversion	15000	1111,83
95	PMX	Inversion	15000	1142,76
95	PMX	Inversion	15000	1170,59
60	PMX	Swapping	10000	1332,10
60	PMX	Swapping	10000	1332,00
60	PMX	Swapping	10000	1355,10
60	PMX	Swapping	10000	1371,70
60	PMX	Swapping	10000	1381,40
60	PMX	Swapping	10000	1391,00
60	PMX	Swapping	10000	1395,00
60	PMX	Swapping	10000	1401,00
60	PMX	Swapping	10000	1457,80
80	PMX	Swapping	5000	1172,90
80	PMX	Swapping	5000	1208,00
80	PMX	Swapping	5000	1240,80
80	PMX	Swapping	5000	1308,10
80	PMX	Swapping	5000	1328,40
80	PMX	Swapping	5000	1496,10
80	PMX	Swapping	5000	1500,30
80	PMX	Swapping	5000	1521,30
80	PMX	Swapping	5000	1565,10
80	PMX	Swapping	5000	1559,60
60	PMX	SwapInvers	10000	1018,69
60	PMX	SwapInvers	10000	1041,11
60	PMX	SwapInvers	10000	1052,04
60	PMX	SwapInvers	10000	1083,40
60	PMX	SwapInvers	10000	1099,58
60	PMX	SwapInvers	10000	1148,08
60	PMX	SwapInvers	10000	1148,54
60	PMX	SwapInvers	10000	1149,60
60	PMX	SwapInvers	10000	1158,19
85	PMX	SwapInvers	15000	956,54
85	PMX	SwapInvers	15000	1030,83
85	PMX	SwapInvers	15000	1060,93
85	PMX	SwapInvers	15000	1065,43
85	PMX	SwapInvers	15000	1076,03
85	PMX	SwapInvers	15000	1089,27
85	PMX	SwapInvers	15000	1144,61
85	PMX	SwapInvers	15000	1144,93
85	PMX	SwapInvers	15000	1153,46
85	PMX	SwapInvers	15000	1220,90
90	PMX	SwapInvers	10000	967,60
90	PMX	SwapInvers	10000	1000,10
90	PMX	SwapInvers	10000	1019,80
90	PMX	SwapInvers	10000	1024,40
90	PMX	SwapInvers	10000	1042,10
90	PMX	SwapInvers	10000	1044,00
90	PMX	SwapInvers	10000	1050,10
90	PMX	SwapInvers	10000	1068,10
90	PMX	SwapInvers	10000	1115,30
95	OX	Inversion	10000	1016,20
95	OX	Inversion	10000	1079,70
95	OX	Inversion	10000	1094,60
95	OX	Inversion	10000	1156,70
95	OX	Inversion	10000	1167,70
95	OX	Inversion	10000	1179,60
95	OX	Inversion	10000	1208,70
95	OX	Inversion	10000	1226,70
95	OX	Inversion	10000	1228,40
95	OX	Swapping	10000	1055,40
95	OX	Swapping	10000	1089,10

95	OX	Swapping	10000	1124,70
95	OX	Swapping	10000	1143,00
95	OX	Swapping	10000	1147,40
95	OX	Swapping	10000	1169,20
95	OX	Swapping	10000	1205,00
95	OX	Swapping	10000	1250,10
95	OX	Swapping	10000	1251,40
95	OX	Swapping	10000	1299,90
95	OX	SwapInvers	10000	1077,50
95	OX	SwapInvers	10000	1091,50
95	OX	SwapInvers	10000	1130,70
95	OX	SwapInvers	10000	1184,20
95	OX	SwapInvers	10000	1193,10
95	OX	SwapInvers	10000	1219,60
95	OX	SwapInvers	10000	1235,50
95	OX	SwapInvers	10000	1241,60
95	OX	SwapInvers	10000	1258,10
95	OX	SwapInvers	10000	1267,80
65	UX	Inversion	20000	1095,70
65	UX	Inversion	20000	1115,60
65	UX	Inversion	20000	1134,10
65	UX	Inversion	20000	1148,50
65	UX	Inversion	20000	1157,20
65	UX	Inversion	20000	1169,30
65	UX	Inversion	20000	1175,00
65	UX	Inversion	20000	1181,90
65	UX	Inversion	20000	1182,10
65	UX	Inversion	20000	1193,80
65	UX	Swapping	20000	1524,10
65	UX	Swapping	20000	1545,70
65	UX	Swapping	20000	1591,00
65	UX	Swapping	20000	1644,70
65	UX	Swapping	20000	1781,70
65	UX	Swapping	20000	1885,60
65	UX	Swapping	20000	1903,20
65	UX	Swapping	20000	1952,50
65	UX	Swapping	20000	2132,80
85	UX	Swapping	15000	1590,90
85	UX	Swapping	15000	1650,70
85	UX	Swapping	15000	1655,00
85	UX	Swapping	15000	1666,60
85	UX	Swapping	15000	1745,20
85	UX	Swapping	15000	1768,60
85	UX	Swapping	15000	1802,30
85	UX	Swapping	15000	1973,90
85	UX	Swapping	15000	2055,30
65	UX	SwapInvers	20000	1060,30
65	UX	SwapInvers	20000	1113,80
65	UX	SwapInvers	20000	1125,90
65	UX	SwapInvers	20000	1171,80
65	UX	SwapInvers	20000	1185,00
65	UX	SwapInvers	20000	1227,80
65	UX	SwapInvers	20000	1266,80
65	UX	SwapInvers	20000	1307,20
65	UX	SwapInvers	20000	1399,00

Tab. C.7 – Le plan d'expériences de la phase Réglage pour le codage direct (448 essais)

C.2.2.2 Analyse des effets (avec l'outil Scaled Estimates)

Term	Prob>|t|
Taux de croisement[60]	0,2254
Taux de croisement[65]	0,1887
Taux de croisement[70]	0,9440
Taux de croisement[75]	0,5171
Taux de croisement[80]	0,6639
Taux de croisement[85]	0,4897
Taux de croisement[90]	0,8015
Taux de croisement[95]	0,7204
Croisement[PMX]	<,0001
Croisement[OX]	<,0001
Croisement[UX]	<,0001
Mutation[Inversion]	<,0001
Mutation[Swapping]	<,0001
Mutation[SwapInvers]	<,0001
Génération	0,0005
Taux de croisement[60]*Croisement[PMX]	0,5379
Taux de croisement[60]*Croisement[OX]	0,5330
Taux de croisement[60]*Croisement[UX]	0,9048
Taux de croisement[65]*Croisement[PMX]	0,8270
Taux de croisement[65]*Croisement[OX]	0,6096
Taux de croisement[65]*Croisement[UX]	0,7404
Taux de croisement[70]*Croisement[PMX]	0,8700
Taux de croisement[70]*Croisement[OX]	0,9863
Taux de croisement[70]*Croisement[UX]	0,8573
Taux de croisement[75]*Croisement[PMX]	0,2658

Term	Prob>|t|
Taux de croisement[75]*Croisement[OX]	0,7256
Taux de croisement[75]*Croisement[UX]	0,4503
Taux de croisement[80]*Croisement[PMX]	0,2591
Taux de croisement[80]*Croisement[OX]	0,6516
Taux de croisement[80]*Croisement[UX]	0,5469
Taux de croisement[85]*Croisement[PMX]	0,9932
Taux de croisement[85]*Croisement[OX]	0,8591
Taux de croisement[85]*Croisement[UX]	0,8561
Taux de croisement[90]*Croisement[PMX]	0,9178
Taux de croisement[90]*Croisement[OX]	0,6959
Taux de croisement[90]*Croisement[UX]	0,7675
Taux de croisement[95]*Croisement[PMX]	0,6424
Taux de croisement[95]*Croisement[OX]	0,1865
Taux de croisement[95]*Croisement[UX]	0,5052
Taux de croisement[60]*Mutation[Inversion]	0,2726
Taux de croisement[60]*Mutation[Swapping]	0,9537
Taux de croisement[60]*Mutation[SwapInvers]	0,2949
Taux de croisement[65]*Mutation[Inversion]	0,5307
Taux de croisement[65]*Mutation[Swapping]	0,4242
Taux de croisement[65]*Mutation[SwapInvers]	0,8629
Taux de croisement[70]*Mutation[Inversion]	0,1655
Taux de croisement[70]*Mutation[Swapping]	0,8031
Taux de croisement[70]*Mutation[SwapInvers]	0,2546
Taux de croisement[75]*Mutation[Inversion]	0,2616
Taux de croisement[75]*Mutation[Swapping]	0,6220

Term	Prob>|t|
Taux de croisement[75]*Mutation[SwapInvers]	0,5281
Taux de croisement[80]*Mutation[Inversion]	0,6184
Taux de croisement[80]*Mutation[Swapping]	0,4399
Taux de croisement[80]*Mutation[SwapInvers]	0,8202
Taux de croisement[85]*Mutation[Inversion]	0,5988
Taux de croisement[85]*Mutation[Swapping]	0,3220
Taux de croisement[85]*Mutation[SwapInvers]	0,5903
Taux de croisement[90]*Mutation[Inversion]	0,5511
Taux de croisement[90]*Mutation[Swapping]	0,2657
Taux de croisement[90]*Mutation[SwapInvers]	0,0644
Taux de croisement[95]*Mutation[Inversion]	0,6735
Taux de croisement[95]*Mutation[Swapping]	0,2309
Taux de croisement[95]*Mutation[SwapInvers]	0,1118
Croisement[PMX]*Mutation[Inversion]	0,0297
Croisement[PMX]*Mutation[Swapping]	0,0006
Croisement[PMX]*Mutation[SwapInvers]	0,1655
Croisement[OX]*Mutation[Inversion]	<,0001
Croisement[OX]*Mutation[Swapping]	<,0001
Croisement[OX]*Mutation[SwapInvers]	<,0001
Croisement[UX]*Mutation[Inversion]	<,0001
Croisement[UX]*Mutation[Swapping]	<,0001
Croisement[UX]*Mutation[SwapInvers]	<,0001
Taux de croisement[60]*(Génération-12608,2)	0,0590
Taux de croisement[65]*(Génération-12608,2)	0,6796
Taux de croisement[70]*(Génération-12608,2)	0,1526
Taux de croisement[75]*(Génération-12608,2)	0,0880
Taux de croisement[80]*(Génération-12608,2)	0,0517
Taux de croisement[85]*(Génération-12608,2)	0,2196
Taux de croisement[90]*(Génération-12608,2)	0,8492
Taux de croisement[95]*(Génération-12608,2)	0,5659
Croisement[PMX]*(Génération-12608,2)	0,0088
Croisement[OX]*(Génération-12608,2)	0,8586
Croisement[UX]*(Génération-12608,2)	0,0152
Mutation[Inversion]*(Génération-12608,2)	0,2775
Mutation[Swapping]*(Génération-12608,2)	0,1824
Mutation[SwapInvers]*(Génération-12608,2)	0,8103

Tab. C.8 – Effets des facteurs de la phase Réglage pour le codage direct

C.2.2.3 Observation des estimations à l'aide de l'outil Surface profiler

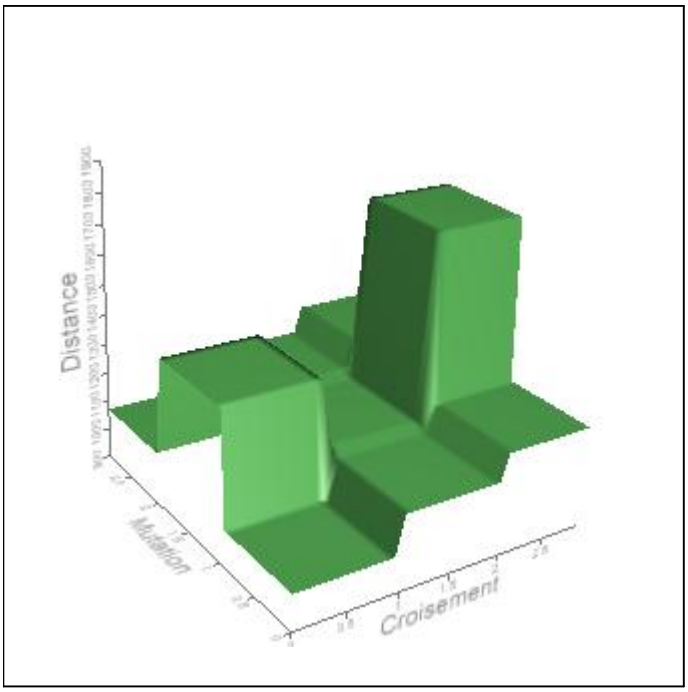


Fig. C.10 – Graphique 3D distance=f(croisement, mutation) de la phase Réglage pour le codage direct

Annexe D

Les résultats numériques des problèmes de VRP avec capacité de Christofides et Eilon (1969)

D.1 Résultats pour le codage direct

Benchmark	OX+I	OX+S	OX+SI	**PMX+I**	PMX+S	PMX+SI	UX+I	UX+S	UX+SI
E-n22-k4	413,133	413,769	404,511	**392,507**	400,333	406,17	398,09	411,06	406,588
E-n23-k3	648,725	638,177	645,06	**610,127**	632,777	627,209	623,72	655,67	626,287
E-n30-k4	586,068	582,486	576,042	573,079	632,559	577,549	579,09	584,25	**572,116**
E-n33-k4	1006,7	1035,71	1038,53	1037,07	1034,57	1022,58	**975,57**	999,31	984,869
E-n51-k5	632,351	647,162	639,844	**585,101**	659,37	587,379	638,25	715,75	601,455
E-n76-k7	924,473	939,517	924,899	**840,542**	1073,04	844,587	888,29	1197,4	888,66
E-n76-k8	980,235	968,347	987,31	917,568	1103,66	917,728	**884,33**	1173	912,507
E-n76-k10	1057,13	1102,82	1060,72	983,208	1057,47	**975,447**	1014	1363,2	1013,99
E-n76-k15	1300,09	1292,11	1308,97	1223,61	1304,54	**1209,23**	1374,7	1495,4	1318,76
E-n101-k8	1134,33	1159,95	1148,44	1083,48	1232,28	**1066**	1158,2	1741,5	1170,55
E-n101-k14	1467,66	1488,33	1444,41	1398,73	1455,92	**1369,74**	1461,4	1959,7	1515,36

Tab. D.1 –Les moyennes de 10 tests du programme de codage direct avec la sélection binaire pour reproduction (11 benchmarks).

Benchmark	OX+I	OX+S	OX+SI	**PMX+I**	PMX+S	PMX+SI	**UX+I**	UX+S	UX+SI	MRL
E-n22-k4	395,62	381,85	384,85	**377,43**	378,661	384,522	**377,43**	386,47	379,41	375
E-n23-k3	594,99	590,15	581,6	**569,88**	571,397	570,561	581,99	586,84	587,91	569
E-n30-k4	520,16	530,02	532,39	547,43	578,684	534,238	**509,89**	514,64	514,2	-
E-n33-k4	921,44	981,96	949,69	925,99	972,898	**886,727**	904,07	914,16	904,89	835
E-n51-k5	598,32	588,99	580,35	**542,18**	584,966	558,379	573,16	668,51	580,35	521
E-n76-k7	849,28	818,46	862,07	786,59	852,69	**784,753**	808,51	1047,7	786,9	683
E-n76-k8	921,96	876,73	920,6	848,28	1013,44	**786,766**	873,37	1048,5	848,28	735
E-n76-k10	989,57	1081,9	1003,6	954,77	988,179	**948,7**	965,4	1247	967,18	830
E-n76-k15	1248,9	1224,1	1242,1	1181	1239,53	**1156,54**	1234,4	1337,1	848,28	-
E-n101-k8	1007,6	966,13	1050,3	**956,3**	1008,65	976,225	1083,5	1284,8	1057,2	815
E-n101-k14	1356,9	1400,1	1365,4	1328,5	1366,85	**1249,09**	1268,4	1857,3	1330	1071

Tab. D.2 –Meilleures solutions entre les 10 tests du programme à codage direct avec sélection binaire pour reproduction (11 benchmarks) et meilleures solutions publiées.

D.1.1 Tableaux de comparaison entre la sélection aléatoire et la sélection binaire pour le codage direct

Benchmark	OX+I	OX+S	OX+SI	PMX+I	PMX+S	PMX+SI	UX+I	UX+S	UX+SI
E-n22-k4	binaire	aléatoire	binaire	binaire	binaire	aléatoire	binaire	binaire	aléatoire
E-n23-k3	aléatoire	aléatoire	aléatoire	aléatoire	aléatoire	binaire	aléatoire	aléatoire	aléatoire
E-n30-k4	binaire	binaire	binaire	binaire	aléatoire	binaire	binaire	aléatoire	aléatoire
E-n33-k4	binaire	aléatoire	aléatoire	binaire	aléatoire	aléatoire	aléatoire	aléatoire	aléatoire
E-n51-k5	aléatoire	aléatoire	aléatoire	binaire	aléatoire	aléatoire	aléatoire	aléatoire	binaire
E-n76-k7	aléatoire	aléatoire	aléatoire	binaire	binaire	binaire	aléatoire	aléatoire	aléatoire
E-n76-k8	binaire	binaire	aléatoire	binaire	aléatoire	aléatoire	binaire	aléatoire	binaire
E-n76-k10	binaire	aléatoire	aléatoire	aléatoire	aléatoire	aléatoire	aléatoire	aléatoire	aléatoire
E-n76-k15	aléatoire	aléatoire	aléatoire	binaire	aléatoire	binaire	aléatoire	aléatoire	binaire
E-n101-k8	binaire	binaire	binaire	aléatoire	binaire	aléatoire	aléatoire	aléatoire	binaire
E-n101-k14	aléatoire	aléatoire	binaire	binaire	aléatoire	binaire	binaire	aléatoire	binaire

Tab. D.3 –Sélection de la meilleure moyenne parmi toutes les moyennes obtenues avec un codage direct pour chaque benchmark et chaque combinaison d'opérateurs.

Benchmark	OX+I	OX+S	OX+SI	PMX+I	PMX+S	PMX+SI	UX+I	UX+S	UX+SI
E-n22-k4	aléatoire	binaire	aléatoire	aléatoire	aléatoire	aléatoire	binaire	aléatoire	aléatoire
E-n23-k3	aléatoire	binaire	aléatoire	binaire	aléatoire	binaire	aléatoire	aléatoire	aléatoire
E-n30-k4	binaire	binaire	binaire	aléatoire	aléatoire	aléatoire	binaire	aléatoire	aléatoire
E-n33-k4	aléatoire	aléatoire	aléatoire	binaire	aléatoire	binaire	aléatoire	binaire	binaire
E-n51-k5	aléatoire	binaire	aléatoire	binaire	aléatoire	aléatoire	aléatoire	aléatoire	aléatoire
E-n76-k7	aléatoire	binaire	aléatoire	aléatoire	binaire	binaire	binaire	aléatoire	binaire
E-n76-k8	binaire	binaire	aléatoire	binaire	aléatoire	binaire	aléatoire	aléatoire	binaire
E-n76-k10	aléatoire	aléatoire	aléatoire	aléatoire	aléatoire	binaire	aléatoire	aléatoire	aléatoire
E-n76-k15	aléatoire	aléatoire	aléatoire	binaire	aléatoire	binaire	=	=	binaire
E-n101-k8	binaire	binaire	binaire	binaire	binaire	aléatoire	binaire	aléatoire	binaire
E-n101-k14	binaire	aléatoire	aléatoire	binaire	aléatoire	binaire	binaire	aléatoire	binaire

Tab. D.4 –Sélection de la meilleure solution parmi toutes les solutions des algorithmes à codage direct pour chaque benchmark et chaque combinaison d'opérateurs.

D.2 Résultats du codage indirect

D.2.1 Tableaux des résultats de la sélection aléatoire:

Benchmark	OX+I	OX+S	OX+SI	PMX+I	PMX+S	PMX+SI	UX+I	UX+S	UX+SI
E-n22-k4	**380,52**	393,08	388,55	445,87	525,67	475,43	401,93	393,73	388,39
E-n23-k3	595,33	583,68	597,88	667,20	846,63	797,15	600,15	582,93	**569,32**
E-n30-k3	597,38	609,58	633,32	751,33	957,88	840,98	587,55	575,78	**571,73**
E-n30-k4	547,89	558,15	563,80	775,43	977,52	864,07	546,24	**544,07**	549,96
E-n33-k4	917,89	947,36	966,66	1020,20	1213,50	1159,22	942,10	965,38	**912,78**
E-n51-k5	759,46	812,80	834,14	890,17	1289,98	1227,68	792,78	**756,30**	757,87
E-n76-k7	999,77	1082,92	1048,37	1362,92	2031,60	1946,66	906,77	928,30	**884,89**
E-n76-k8	1072,11	1198,39	1157,47	1436,25	2088,63	1985,68	**998,09**	1112,41	1020,13
E-n76-k10	**955,56**	1730,39	1537,57	1232,34	2290,75	2176,77	1547,24	1964,62	1683,05
E-n76-k14	**1154,06**	1929,37	1425,91	1461,89	2576,71	2362,20	1762,01	1931,23	1770,58
E-n76-k15	**1139,63**	1359,07	1299,12	1295,71	2193,52	2131,94	1208,83	1599,32	1321,23
E-n101-k8	1299,94	1466,87	1415,76	1906,79	2844,06	2759,65	1274,22	1240,28	**1233,02**
E-n101-k14	1726,68	1837,36	1886,71	2124,66	3062,37	2917,36	**1604,53**	1851,08	1759,73

Tab. D.5 –Les moyennes de 10 tests du programme avec codage indirect, sélection aléatoire pour reproduction et méthode de découpage A (13 benchmarks).

Benchmark	OX+I	OX+S	OX+SI	PMX+I	PMX+S	PMX+SI	UX+I	UX+S	UX+SI	MRL
E-n22-k4	**375,28**	**375,28**	**375,28**	393,30	490,25	446,82	**375,28**	**375,28**	**375,28**	375
E-n23-k3	**568,56**	**568,56**	569,75	615,36	773,60	742,68	**568,56**	**568,56**	**568,56**	569
E-n30-k3	547,17	582,06	552,76	688,03	1004,76	715,87	553,77	**542,42**	546,82	534
E-n30-k4	513,76	525,83	514,00	674,40	1006,59	778,65	510,93	513,09	**508,63**	-
E-n33-k4	**879,69**	886,51	1030,5	1001,39	1139,21	1119,25	1002,0	1000,6	1002,2	835
E-n51-k5	**684,69**	751,95	753,73	847,96	1332,70	1173,44	1026,5	1134,4	1006,8	521
E-n76-k7	1002,92	1039,3	1009,8	1271,05	1835,39	1901,55	1020,1	1196,5	**842,59**	683
E-n76-k8	1006,86	1089,7	1080,2	1321,63	2029,09	1938,11	1088,8	1035,8	**1001,1**	735
E-n76-k10	**1002,29**	1580,3	1377,9	1124,44	2079,61	2121,39	1203,5	1878,5	1368,5	830
E-n76-k14	**1113,22**	1288,3	1264,8	1306,85	2354,72	2012,90	1269,2	1277,4	1138,6	1021
E-n76-k15	**1103,43**	1214,7	1216,1	1191,41	2098,44	2082,45	1152,6	1183,0	1214,7	-
E-n101-k8	1203,57	1270,1	1283,7	1780,20	2713,17	2662,02	**1040,8**	1055,3	1042,0	815
E-n101-k14	1649,31	1673,1	1710,6	1952,09	2862,40	2845,61	1302,6	1385,9	**1291,4**	1071

Tab. D.6 –Meilleures solutions entre les 10 tests du programme avec codage indirect, sélection aléatoire pour reproduction et méthode de découpage A (13 benchmarks) et meilleures solutions publiées.

Benchmark	OX+I	OX+S	OX+SI	PMX+I	PMX+S	PMX+SI	UX+I	UX+S	UX+SI
E-n22-k4	383,08	415,65	**379,60**	382,99	506,21	381,66	380,55	438,49	381,95
E-n23-k3	**570,67**	605,58	572,02	574,29	781,66	573,39	575,66	582,64	572,23
E-n30-k3	**548,37**	660,72	563,41	563,12	909,18	557,31	549,09	581,98	552,00
E-n30-k4	518,14	565,89	512,13	514,88	841,18	579,33	519,87	528,77	**511,51**
E-n33-k4	899,16	980,37	894,39	902,86	1181,28	915,99	**880,38**	925,83	892,15
E-n51-k5	613,28	1030,64	**601,32**	761,34	1305,90	742,55	695,79	1122,34	705,81
E-n76-k7	**809,65**	1159,24	809,94	1013,40	1994,62	1009,64	835,73	1024,58	873,55
E-n76-k8	**958,76**	1502,43	980,04	1294,38	2198,79	1541,02	983,69	1515,77	1130,25
E-n76-k10	1091,73	2464,81	1394,76	1407,78	2512,77	1676,09	**1064,67**	1137,72	1102,75
E-n76-k15	**1148,36**	1431,30	1399,22	1375,15	2120,00	2174,60	1217,76	1240,12	1180,38
E-n101-k8	1144,24	1842,74	**1128,50**	1539,14	2916,86	1512,02	1233,65	1294,49	1282,04
E-n101-k14	**1531,23**	2125,82	2002,86	2034,92	3225,40	3246,24	1588,35	1708,40	1578,43

Tab. D.7 –Les moyennes de 10 tests du programme avec codage indirect, sélection aléatoire pour reproduction et méthode de découpage B (12 benchmarks).

Benchmark	OX+I	OX+S	OX+SI	PMX+I	PMX+S	PMX+SI	UX+I	UX+S	UX+SI	MRL
E-n22-k4	**375,28**	383,52	**375,28**	**375,28**	463,66	**375,28**	**375,28**	391,63	**375,28**	375
E-n23-k3	**568,56**	569,75	**568,56**	**568,56**	703,24	**568,56**	**568,56**	**568,56**	**568,56**	569
E-n30-k3	537,46	604,59	538,79	543,79	857,85	857,85	**528,70**	563,32	538,79	534
E-n30-k4	507,89	533,50	**505,01**	505,68	713,33	506,94	508,11	508,81	**505,01**	-
E-n33-k4	853,98	911,95	867,05	872,47	1142,41	857,08	**846,53**	866,74	857,58	835
E-n51-k5	567,19	915,10	**557,13**	700,79	1190,91	655,16	624,19	769,04	640,75	521
E-n76-k7	**761,49**	1094,3	762,26	939,00	1886,33	951,79	781,85	888,49	771,41	683
E-n76-k8	896,54	1305,6	874,69	1167,27	2156,61	1195,08	**843,24**	1011,3	1039,3	735
E-n76-k10	**950,16**	2371,5	1215,5	1277,29	1623,22	1301,83	981,59	1028,9	983,06	830
E-n76-k15	**1093,81**	1288,5	1321,8	1333,73	2075,90	2093,03	1125,6	1172,6	1104,1	-
E-n101-k8	1029,81	1708,2	**981,59**	1356,21	2815,74	1294,59	1067,3	1155,4	1085,8	815
E-n101-k14	**1362,69**	1986,9	1453,6	1814,51	3138,71	2145,03	1449,4	1481,5	1399,6	1071

Tab. D.8 –Meilleures solutions entre les 10 tests du programme avec codage indirect, sélection aléatoire pour reproduction et méthode de découpage B (12 benchmarks) et meilleures solutions publiées.

D.2.2 Tableaux des résultats de la sélection binaire

Benchmark	OX+I	OX+S	OX+SI	PMX+I	PMX+S	PMX+SI	UX+I	UX+S	UX+SI
E-n22-k4	**385,52**	410,03	391,42	407,79	486,44	464,29	394,79	416,37	404,03
E-n23-k3	**592,21**	621,53	609,69	647,12	807,62	783,33	614,67	594,77	600,49
E-n30-k3	**587,50**	632,84	628,86	633,41	887,21	861,95	645,50	601,87	616,10
E-n30-k4	**521,43**	643,58	609,45	541,69	877,34	979,33	566,28	567,07	579,91
E-n33-k4	**919,32**	958,58	983,96	990,54	1161,08	1105,74	959,35	976,60	937,23
E-n51-k5	**677,73**	839,75	821,31	808,35	1218,59	1110,69	816,41	956,91	896,74
E-n76-k7	**926,43**	1173,70	1083,64	1202,28	1972,33	1761,61	1089,38	1135,83	1137,34
E-n76-k8	**841,45**	1249,00	1218,09	1099,70	1974,88	2029,40	1320,10	1344,33	1288,85
E-n76-k10	**940,41**	1827,22	1723,52	1268,75	2176,80	2257,98	1737,76	1991,85	1869,58
E-n76-k14	**1136,38**	2041,51	1620,09	1479,98	2437,33	2294,27	1766,90	2112,87	1864,54
E-n76-k15	**1154,30**	1448,48	1410,01	1334,03	2166,43	2212,88	1543,66	1729,06	1653,87
E-n101-k8	**1259,00**	1609,24	1468,85	1647,95	2771,25	2295,91	1427,51	1501,87	1498,41
E-n101-k14	**1351,94**	1959,34	2003,36	1811,83	2905,06	2945,54	2051,96	2113,74	2025,86

Tab. D.9 –Les moyennes de 10 tests du programme avec codage indirect, sélection binaire pour reproduction et méthode de découpage A (13 benchmarks).

Benchmark	OX+I	OX+S	OX+SI	PMX+I	PMX+S	PMX+SI	UX+I	UX+S	UX+SI	MRL
E-n22-k4	**375,28**	383,87	**375,28**	**375,28**	453,77	**375,28**	**375,28**	393,33	**375,28**	375
E-n23-k3	**568,56**	582,43	569,75	**568,56**	768,42	590,43	569,75	**568,56**	**568,56**	569
E-n30-k3	**535,80**	574,20	568,66	543,03	775,01	616,75	568,35	577,39	560,28	534
E-n30-k4	**508,35**	553,25	538,17	505,01	781,76	768,11	520,02	526,08	515,78	-
E-n33-k4	**848,85**	902,29	923,84	854,39	1118,95	950,69	907,82	884,65	880,95	835
E-n51-k5	**553,31**	747,00	752,22	623,72	1154,05	873,48	659,21	702,45	717,54	521
E-n76-k7	**743,76**	1008,81	981,24	923,92	1929,72	1350,73	935,79	965,80	948,76	683
E-n76-k8	**794,11**	1100,81	1041,30	980,54	1888,76	1501,04	1072,97	939,40	1031,88	735
E-n76-k10	**893,22**	1698,75	1613,59	1162,30	2080,87	2107,01	1544,56	1848,0	1481,31	830
E-n76-k14	**1104,8**	1365,78	1314,88	1314,88	2320,14	2021,65	1516,98	1798,0	1281,54	1021
E-n76-k15	**1087,2**	1164,63	1265,12	1228,01	2127,00	2134,35	1339,62	1565,0	1510,72	-
E-n101-k8	**991,19**	1417,19	1274,72	1339,79	2704,58	1825,36	1163,69	1161,5	1150,33	815
E-n101-k14	**1214,6**	1674,49	1778,94	1668,12	2836,94	2110,51	1562,84	1700,9	1577,46	1071

Tab. D.10 –Meilleures solutions entre les 10 tests du programme avec codage indirect, sélection binaire pour reproduction et méthode de découpage A (13 benchmarks) et meilleures solutions publiées.

Benchmark	OX+I	OX+S	OX+SI	PMX+I	PMX+S	PMX+SI	UX+I	UX+S	UX+SI
E-n22-k4	**379,52**	423,34	439,83	379,65	573,52	513,77	426,62	435,77	383,31
E-n23-k3	**570,15**	605,93	592,98	571,92	888,83	794,73	604,33	644,11	576,50
E-n30-k3	**548,58**	660,66	678,94	573,17	1042,65	906,49	627,86	625,90	558,61
E-n30-k4	525,68	573,43	579,07	**520,43**	958,11	870,97	570,93	572,87	521,09
E-n33-k4	**896,86**	1032,17	1020,49	913,47	1279,00	1180,99	1002,16	1015,14	899,09
E-n51-k5	**607,70**	1108,54	1001,66	799,11	1440,64	1324,18	1074,46	1114,07	689,19
E-n76-k7	**858,26**	1393,15	1384,79	1075,26	2203,02	2038,78	1191,62	1270,37	1001,03
E-n76-k8	**1000,84**	1635,75	1625,25	1335,57	2381,53	2192,01	1468,95	1775,15	1139,65
E-n76-k10	**1118,75**	1569,79	1398,57	1355,50	2522,89	2246,32	1504,29	1652,49	1171,11
E-n76-k15	**1149,35**	1382,98	1387,11	1363,36	2123,79	2189,81	1445,11	1479,68	1220,41
E-n101-k8	**1155,66**	2112,76	2124,39	1529,22	3119,06	2910,16	1638,60	1840,98	1318,83
E-n101-k14	**1664,42**	2237,85	2280,06	2035,22	3305,82	3224,15	2141,43	2246,50	1847,16

Tab. D.11 –Les moyennes de 10 tests du programme avec codage indirect, sélection binaire pour reproduction et méthode de découpage B (12 benchmarks).

Benchmark	OX+I	OX+S	OX+SI	PMX+I	PMX+S	PMX+SI	UX+I	UX+S	UX+SI	MRL
E-n22-k4	**375,28**	388,26	388,42	**375,28**	527,49	493,40	394,43	408,67	**375,28**	375
E-n23-k3	**568,56**	569,75	569,75	**568,56**	828,55	764,40	**568,56**	592,88	**568,56**	569
E-n30-k3	**538,79**	600,41	602,42	546,46	819,75	830,33	600,97	574,66	**538,79**	534
E-n30-k4	**505,23**	518,45	506,03	507,25	848,68	803,66	527,08	507,11	**505,23**	-
E-n33-k4	**857,25**	960,27	980,20	875,04	1202,18	1136,02	876,70	931,34	862,36	835
E-n51-k5	**570,49**	988,59	909,34	753,44	1239,14	1256,46	840,91	970,40	605,90	521
E-n76-k7	**813,38**	1226,57	1241,21	992,07	2109,79	1966,88	993,81	1037,26	854,45	683
E-n76-k8	**931,43**	1460,53	1475,91	1209,27	2139,12	2128,58	1027,63	1339,38	944,39	735
E-n76-k10	1009,81	1464,56	1042,01	1206,92	2459,51	1846,58	1169,19	1298,48	**1006,54**	830
E-n76-k15	**1109,03**	1213,47	1272,61	1260,81	2108,48	2109,63	1242,26	1303,66	1141,28	-
E-n101-k8	**1022,32**	1840,10	1936,59	1460,83	2941,79	2841,03	1362,19	1524,08	1087,67	815
E-n101-k14	1584,41	1982,24	2097,21	1895,98	3124,73	3100,06	1744,87	1709,79	**1524,54**	1071

Tab. D.12 –Meilleures solutions entre les 10 tests du programme avec codage indirect, sélection binaire pour reproduction et méthode de découpage B (12 benchmarks) et meilleures solutions publiée

Publications

- Mais Haj-Rachid, Christelle Bloch, Wahiba Ramdane Cherif, and Pascal Chatonnay. **Solving capacitated vehicle routing problem via different genetic operators.** *In META'10, The 3rd International Conference on Metaheuristics and Nature Inspired Computing*, Djerba Island, Tunisia, Octobrer 27 – 31, 2010.

- Mais Haj-Rachid, Wahiba Ramdane-Cherif, Pascal Chatonnay, and Christelle Bloch. **Comparing the performance of genetic operators for the vehicle routing problem**. In *MCPL 2010. Conf. on Management and Control of Production Logistics*, University of Coimbra, Portugal, September 8 -10, 2010.

- Mais Haj-Rachid, Christelle Bloch, Wahiba Ramdane Cherif, and Pascal Chatonnay**. Différentes opérateurs évolutionnaires de permutation: sélections, croisements et mutations.** Research Report RR2010-07, LIFC - Laboratoire d'Informatique de l'Université de Franche Comté, France, July 2010.

- Mais Haj-Rachid, Christelle Bloch, Wahiba Ramdane Cherif, and Pascal Chatonnay**. Vehicle routing problem: survey and methods of resolution.** Research Report RR2010-06, LIFC - Laboratoire d'Informatique de l'Université de Franche Comté, France, June 2010.

- Mais Haj-Rachid, Wahiba Ramdane-Cherif, Pascal Chatonnay, and Christelle Bloch. **A study of performance on crossover and mutation operators for vehicle routing problem**. In *ILS'10, 3rd int. Conf. on Information Systems, Logistics and Supply Chain*, Casablanca, Morocco, April 2010. Note: Proceedings on CD-ROM.

- Mais Haj-Rachid, Wahiba Ramdane Cherif, Christelle Bloch, and Pascal Chatonnay. **Nouvelle approche pour classifier les problèmes de tournées de Véhicules**. In *CIFMA'03, 3ème Congrès International Francophone de Mécanique Avancée, Conception, Fiabilité et Optimisation*, Faculté de Génie Mécanique, Aleppo, Syria, pages 38--46, April 2008.

- Mais Haj-Rachid, Wahiba Ramdane Cherif, Christelle Bloch, and Pascal Chatonnay. **Classification de problèmes de tournées de véhicules**. In *MOSIM'08, 7ème Conférence Internationale de MOdélisation et SIMulation*, Paris, France, pages 1149--1158, March 2008.

- Mais Haj-Rachid, Wahiba Ramdane Cherif, Christelle Bloch, and Pascal Chatonnay. **Proposition de notation pour les problèmes de tournées**. In *ROADEF'08, 9ème congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision*, Clermont-Ferrand, France, pages 63--77, February 2008.

- Mais Haj-Rachid. **Notation pour les problèmes de tournées de véhicules**. Présentation de 30 minutes à la 2ème journée de l'optimisation (Optimisation combinatoire dans les réseaux et les graphes), organisée conjointement par l'UTBM et l'UFC, January 2008.

Bibliographie

(Achuthan *et al.*, 1995) N.R. Achuthan, L. Caccetta, and S.P. Hill. A New Subtour Elimination Constraint for the Vehicle Routing Problem. *European Journal of Operations Research*, 91, pages 573-586, 1995.

(Aguilera *et al.*, 1996) L. M. Aguilera , Z. Binder , F. Hanada , J. J. Guimaraes Ramos. Non-identical parallel machines scheduling with sequence-dependent change over-costs in an industrial application. CESA'96 IMACS Multi-conference : computational engineering in systems applications, Lille , pages 559-564,1996.

(Alba et Dorronsoro, 2004) E. Alba, and B. Dorronsoro. Solving the Vehicle Routing Problem by Using Cellular Genetic Algorithms. *EvoCOP*, 11-20, 2004.

(Aldaihani et Dessouky, 2003) M. M. Aldaihani, and M. M. Dessouky. Hybrid Scheduling Methods for Paratransit Operations. *Computers and Industrial Engineering*, 45(1), pages 75-96, 2003.

(Alonso *et al.*, 2006) F. Alonso, M.J. Alvarez, and J.E. Beasley. A Tabu Search Algorithm for the Periodic Vehicle Routing Problem with Multiple Vehicle Trips and Accessibility Restrictions. *To appear in Journal of the OR Society*, 2006.

(Alvarenga *et al.*, 2007) G. B. Alvarenga, G.R. Mateus, and G.D. Tomi. A genetic and set partitioning two-phase approach for the vehicle routing problem with time windows. *Computers & Operations Research*, 34, 1561–1584, 2007.

(Ambrosini *et al.*, 2004) M. Ambrosini, T. Caruso, S. Foresti, and G. Righini. A GRASP for the pickup and delivery problem with rear loading. Information Technology Department Technical Report NO.65 , University of Milan, 2004.

(Anthony, 1965) R. N. Anthony. *Planning and Control Systems : a framework for analysis*. Harvard University Press, 1965.

(Archetti *et al.*, 2002) C. Archetti, A. Hertz, and M.G. Speranza. A Tabu Search Algorithm for the Split Delivery Vehicle Routing Problem. *Mathematics of Information Technology and Complex Systems*, 2002.

(Atmar, 1976) J. W. Atmar. *Speculation on the evolution of intelligence and its possible realization in machine form*. PhD thesis, New Mexico

State University, Las Cruces, NM, 1976.

(Augerat *et al.*, 1995) P. Augerat, J.M. Belenguer, E. Benavent, A. Corberán, D. Naddef, and G. Rinaldi. Computational Results with a Branch and Cut Code for the Capacitated. Vehicle Routing Problem. Research Report 949-M, Universite Joseph Fourier, Grenoble, France, 1995.

(Ayob et Kendall, 2009) M. Ayob, and G. Kendall. The optimisation of the single surface mount device placement machine in printed circuit board assembly: a Survey. *International Journal of Systems Science*, 40(6), 2009.

(Bäck, 1992), T. Back. Self adaptation in genetic algorithms. *In Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life, MIT Press, Cambridge, MA*, 263–271, 1992.

(Bäck, 1996) T. Bäck. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, New York, 1996.

(Bäck *et al.*, 1993) T. Bäck, G. Rudolph, and H.-P. Schwefel. Evolutionary programming and evolution strategies: Similarities and differences. *In D. B. Fogel and J. W. Atmar, editors, Proceedings of the Second Annual Conference on Evolutionary Programming, Evolutionary Programming Society, La Jolla, CA*, pages 11-22, 1993.

(Bäck *et al.*, 1997) T. Bäck, D. B. Fogel, and Z. Michalewicz. *Handbook of Evolutionary Computation*. Oxford University Press, 1997.

(Bäck *et al.*, 2000) T. Bäck, D. B. Fogel, and T. Michalewicz. *Evolutionary Computation 1: Basic Algorithms and Operators*. Institute of Physics Publishing, 2000.

(Baglin *et al.*, 2001) G. Baglin, O. Bruel, A. Garreau, M. Greif, and C. V. Delft. *Management industriel et logistique*, 3ème édition, 2001.

(Baker, 1987) J. E. Baker. Reducing bias and inefficiency in the selection algorithm. *In J. J. Grefenstette (ed), Proceedings of the International Conference on Genetic Algorithms*, pages14-21, 1987.

(Baker et Ayechew, 2003) B. M. Baker, and M. A. Ayechew. A genetic algorithm for the vehicle routing problem. *Computers and Operations Research*, 30(5), pages 787 - 800, 2003.

(Barbulescu *et al.*, 2004) L. Barbulescu, A. Howe, L. Whitley and M. Roberts. Trading places: How to schedule more in a multi-resource oversubscribed scheduling problem. *In Proceedings of the International Conference on Planning and Scheduling*, Whistler, CA, pages 227-234, 2004.

(Bart *et al.*, 2007) V. Bart, C. Dirk, D. Joost, and V.O. Dirk. Modeling sheet métal integrated production planning for laser cuting and air bending. *Key engineering Materials*, 344, pages 913-920, 2007.

(Baudet *et al.*, 1996) P. Baudet, C. Azzaro, L. Pibouleau, and S. Domenech. A genetic algorithm for batch chemical plant scheduling. *In Proc. Int. Congr. Chemical & Process Engineering*, pages 25-30,1996.

(Baum, 1986) E.B. Baum. Iterated descent: A better algorithm for local search in combinatorial optimization problems. Technical report, Caltech, Pasadena, CA, 1986.

(Baxter, 1981) J. Baxter. Local optima avoidance in depot location. *Journal of the Operational Research Society*, 32, pages 815-819, 1981.

(Beasley, 1983) J.E. Beasley. Route-first cluster-second methods for vehicle routing. *Omega*, 11, pages 403- 408, 1983.

(Bent et Hentenryck, 2001) R. Bent, and P. V. Hentenryck. A Two-Stage Hybrid Local Search for the Vehicle Routing Problem with Time Windows. Technical Report, CS-01-06, Brown University, 2001.

(Berger *et al.*, 2003) J. Berger, M. Barkaoui and O.Bräysy. A route-directed hybrid GA approach for the VRPTW. *Information Systems and Operational Research*, 41, 179-194, 2003.

(Besten *et al.*, 2001) M. L. den. Besten, T. Stützle, and M. Dorigo. Design of iterated local search algorithms: An example application to the single machine total weighted tardiness problem. *In Proceedings of EvoStim'01, Lecture Notes in Computer Science*, pages 441-452, Springer, Berlin, 2001

(Bianchi, 2000) L. Bianchi. Notes on Dynamic Vehicle Routing - The state of the art. Technical report, INDSIA-05-01, 20 December, 2000.

(Bjarnadóttir, 2004) Á.S. Bjarnadóttir. *Solving the Vehicle Routing Problem with Genetic Algorithms*. Thèse.Informatics and Mathematical Modelling, IMM. Technical University of Denmark, 2004.

(Blanton et Wainwright, 1991) J. Blanton and R. L. Wainwright. Multiple vehicle routing with time and capacity constraints using genetic algorithms. *In Proceedings of the Fourth International Conference on Genetic Algorithm*, pages 452-459, 1991.

(Blanton et Wainwright, 1993) J. L. Blanton. Jr, and R. L. Wainwright. Multiple Vehicle Routing with Time and Capacity Constraints Using Genetic Algorithms. *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 452-459, June 01, 1993.

(Blazewicz *et al.*, 1993) J. Blazewicz, K.H. Ecker, G. Schmidt, and J. Weglarz. *Scheduling in Computer and Manufacturing Systems.* Springer-Verlag, Berlin, 1993.

(Bloch et Manier, 1999) C. Bloch, and M.A. Manier. Notation and Typology for the Hoist Scheduling Problem. *IEEE International Conference on Systems, Man, and Cybernetics. Tokyo, Japan*, 4, pages 475–480, 1999.

(Blum et Roli, 2008) Blum, and A. Roli. Hybrid Metaheuristics: An Introduction.

Studies in Computational Intel- ligence (SCI) 114, 1–30, 2008.

(Bodin et Golden 1981) L. D. Bodin, and B. L. Golden. Classification in vehicle routing and scheduling. *Network*, 11(2), 97-108, 1981.

(Bodin *et al.*, 1983) L. D. Bodin, B. L. Golden, A. Assad, and M. Ball. Routing and scheduling of vehicles and crews: the state of the art. *Computers and Operation Research*, 10, pages 63-211, 1983.

(Boudia *et al.*, 2007) M. Boudia, M. A. Ould Louly, and C. Prins. A reactive GRASP and path relinking for a combined production-distribution problem. *Computers and Operation Research*, 2007.

(Boudia et Prins, 2009) M. Boudia, and C. Prins. A memetic algorithm with dynamic population management for an inte- grated production-distribution problem. *EJOR*, 2009.

(Boussedjra *et al.*, 2006) M. Boussedjra, C. Bloch, and A. El moudni. Apport d'une technique de diversification dans un algorithme de recherche de chemin intermodal. In Michel Gourgand and Fouad Riane, (eds), MoSIM'06, 6ème Conférence Francophone de Modélisation, Optimisation et Simulation des Systèmes : Défis et Opportunités, Rabat, Maroc, 2006.

(Bouzgarrou, 1998) M.E. Bouzgarrou. *Parallélisation de la méthode du "Branch and Cut" pour résoudre le problème du voyageur de commerce*. Thèse, Laboratoire de Modélisation et Calcul, Institut d'Informatique et de Mathématiques Appliquées de Grenoble, 1998.

(Boysen *et al.*, 2006) N. Boysen, M. Fliedner, and A. Scholl. A classification of assembly line balancing problems. *Wirtschaftswissenschaftliche Fakultät*, 2006.

(Boysen *et al.*, 2007) N. Boysen, M. Fliedner, and A. Scholl. Sequencing Mixed-Model AssemblyLines: Survey, Classification and Model Critique. *Friedrich-Schiller- University Jena, Department of Economics and Business Administration,* 2007.

(Brandao, 2006) J. Brandao. A new tabu search algorithm for the vehicle routing problem with backhauls. *European Journal of Operational Research*, 173, pages 540–555. , 2006.

(Bräysy, 2003) O. Bräysy, A Reactive Variable Neighborhood Search for the Vehicle-Routing Problem with Time Windows. INFORMS Journal on Computing, 15(4), pages 347-368, 2003

(Braÿsy *et al.*, 2004a) O. Bräysy, , W. Dullaert, and M. Gendreau. Evolutionary algorithms for the vehicle routing problem with time windows. *Journal of Heuristics*, 10(6), pages 587–611, 2004a.

(Braÿsy *et al.*, 2004b) O. Bräysy, G. Hasle, and W. Dullaert. A multi-start local search algorithm for the vehicle routing problem with time windows. *EJOR*, 159(3), pages 586–605, 2004b.

(Braÿsy et Gendreau, O. Bräysy, and M. Gendreau. Vehicle routing problem with time

2005a) windows – Part I: route construction and local search algorithms. *Transportation Science*, 39(1), pages 104–118, 2005a.

(Braÿsy et Gendreau, 2005 b) O. Bräysy, and M. Gendreau. Vehicle routing problem with time windows, Part II: Metaheuristics. *Transportation Science*, 39(1), pages 119–139, 2005b.

(Braÿsy *et al.*, 2005) O. Bräysy, W. Dullaert, and M. Gendreau. Evolutionary algorithms for the vehicle routing problem with time windows. Journal Heuristics, 10, pages 587–611, 2005.

(Braune *et al.*, 2005) R. Braune, S. Wagner, and M. Affenzeller. On the Analysis of Crossover Schemes for Genetic Algorithms Applied to the Job Shop Scheduling Problem. *Proceedings of the 9th World Multi-Conference on Systemics, Cybernetics and Informatics (WMSCI)*, (6), pages 236-241, 2005.

(Burgin, 1973) G. H. Burgin. Systems identification by quasilinear and evolutionary programming. *Journal of Cybernetics*, 3(2), pages 56-75, 1973.

(Calégary *et al.*, 1999) P. Calégary, G. Coray, A. Hertz, D. Kobler, and P. Kuonen. A taxonomy of evolutionary algorithms in combinatorial optimization. *Journal of Heuristics*, 5, pages 145–158, 1999.

(Cantù-Paz, 1997) E. Cant´u-Paz. A survey of parallel genetic algorithms. Report No 97003, Illinois Genetic Algorithms Laboratory, 1997.

(Cerny, 1985) V. Cerny. Thermodynamical approach to the travelling salesman problem. *Journal of Optimization Theory and Applications*, 45(1), pages 41-51, 1985.

(Chalasani et Motwani, 1999) P. Chalasani and R. Motwani. Approximating capacitated routing and delivery problems. *SIAM Journal on Computing*, 28(6), pages 2133-2149, 1999.

(Chan et Mercier, 1989) D. Chan, and D. Mercier. IC insertion: an application of the travelling salesman problem. *International journal of Production Research*, 27(10), pages 1837 – 1841, 1989.

(Chandra et Fisher, 1994) P. Chandra, and M.L. Fisher. Coordination of production and distribution planning. *EJOR*, 1994.

(Chen *et al.*, 2006) A.L. Chen, G.K. Yang, and Z.M. Wu. Hybrid discrete particle swarm optimization algorithm for capacitated vehicle routing problem. *Journal of Zhejiang University science A*, 7(4), pages 607-614, 2006.

(Cheng *et al.*, 2007) C. H. Cheng, Y.P. Gupta, W. H. Lee, and K.F. Wong. A TSP-based heuristic for forming machine groups and part families. *International Journal of Production Research*, 36(5), pages 1325 – 1337,1998.

(Chevrier *et al.*, 2008) R. Chevrier, E. Castex, D. Josselin, Philippe Canalda, and Pascal

	Chatonnay. Un algorithme Génétique pour le transport à la demande en convergence. RIG (Revue Internationales de Géomatique),18(2), 239-268, 2008. Note: Revue internationale en langue française et anglaise.
(Christiansen, 2007)	C. H. Christiansen. *Elements of Vehicle Routing under uncertainty*. PhD Thesis, Department of Business Studies, Aarhus School of Business, University of Aarhus. April, 2007.
(Christofiades et Eilon, 1969)	N. Christofiades and S. Eilon. An algorithm for the vehicle-dispatching problem. *Operations Research*, 20(3), 309-318, 1969.
(Christofides *et al.*, 1979)	N. Christofides, A. Mingozzi, P. Toth. The vehicle routing problem. *In: Christofides N, Mingozzi A, Toth P, Sandi C, editors. Combinatorial optimization. New York: Wiley*, pages 315–338, 1979.
(Christofides et Beasley, 1984)	N. Christofides, and J.E. Beasley. The Period Routing Problem. *Networks*, 14, 237 – 256, 1984.
(Clarke et Wright ,1964)	G. Clarke, and J.W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12, pages 568-581, 1964.
(Coello *et al*., 2007)	C. A. Coello Coello, G. B. Lamont, and D.A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Series: Genetic and Evolutionary Computation, Springer, 2007.
(Connolly, 1990)	D. Connolly. An improved annealing scheme for the QAP. *European Journal of Operational Research*, 46, pages 93-100, 1990.
(Cordeau et Laporte, 2002)	J.F. Cordeau, and G. Laporte. Tabu search heuristics for the vehicle routing problem. Technical Report G- 2002-15,Group for Research in Decision Analysis, 2002.
(Cordeau *et al.*, 2005)	J.-F. Cordeau, G. Laporte, D. Vigo, and M.W.P. Savelsbergh. Vehicle Routing, *Handbooks in Operations Research and Management Science*, 2005.
(Crispim et Brandao, 2005)	J. Crispim, and J. Brandao. Metaheuristics applied to mixed and simultaneous extensions of vehicle routing problems with backhauls. *Journal of the Operational Research Society*, 56(11), pages 1296–1302, 2005.
(Dantzig *et al.*, 1954)	G. Dantzig, R. Fulkerson, and S. Johnson. Solution of a large-scale traveling-salesman problem. *Operations Research*, (2), pages 393-410, 1954.
(Dantzig et Ramser, 1959)	G. Dantzig, and J. Ramser. The truck dispatching problem. *Management Science*, (6), pages 80–91, (1959).
(Davis,1991)	L. Davis. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, 1991.

(Dawkins, 1989) R. Dawkins. Chapitre: Memes: the new replicators, From: The Selfish Gene. (2nd edition), Oxford University Press, pages 368, 1989.

(Deb, 2002) K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley, 2002**.**

(DeJong, 1975) K. DeJong. An *analysis of the behaviour of a class of genetic adaptive systems*. PhD thesis, University of Michigan, USA, 1975.

(De Jong, 2007) K.A. De Jong. Parameter Setting in EAs : a 30 Year Perspective. *Chapter in Parameter Setting in Evolutionary Algorithms, F.G. Lobo, C.F. Lima, and Z. Michalewicz (eds)*, (54), pages 1–18, 2007.

(Dell'Amico *et al.*, 2006) M. Dell'Amico, M. Monaci, C. Pagani, and D. Vigo. Heuristic approaches for the Fleet Size and Mix Vehicle Routing Problem with Time Windows. *Working paper, University of Modena, Italy*, 2006.

(Deneubourg *et al.*, 1990) J.L. Deneubourg, S. Aron, S. Goss, and J.M. Pasteels. The Self-organizing Exploratory Pattern of the Argentine Ant. *Insect Behavior*, (3), pages 159-168, 1990.

(Desrochers *et al..*, 1990) M. Desrochers, J. K. Lenstra, and M. W. P. Savelsbergh. A classification scheme for vehicle routing and scheduling problems. *European Journal of Operational Research*, 46(3), pages 322-332, 1990.

(Dorigo, 1992) M. Dorigo. *Optimization, Learning and Natural Algorithms*. PhD thesis, Politecnico di Milano, Italie, 1992.

(Dorigo *et al.*, 1996) M. Dorigo, V. Maniezzo, and A. Colorni. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics--Part B* 26(1), pages 29-41, 1996.

(Dorigo *et al.*, 1999) M. Dorigo, G. Di Caro, and L.M. Gambardella. Ant Algorithms for Discrete Optimization. Artificial Life, 5(2), 1999.

(Dorigo et Stützle, 2004) M. Dorigo, and T. Stützle. Ant Colony Optimization. Bradford Book, 2004

(Eglese *et al.*, 2005) R.W. Eglese , Z. Fu, and L. Li . A tabu search heuristic for the open vehicle routing problem . *Journal of the Operational Research Society*, 56 (3), pages 267–274, 2005.

(Eiben et Ruttkay, 1997) A. E. Eiben and Z. Ruttkay. Constraint satisfaction problems. *In Handbook of Evolutionary Computation, T. B¨ack, D. Fogel, and M. Michalewicz, Eds. Institute of Physics Publishing Ltd, Bristol*, UK,1997.

(Eiben et Smith, 2003) A. E. Eiben, and J. E. Smih. *Introduction to Evolutionary Computing*. Springer, 2003

(Eilon *et al.*, 1971) S. Eilon, C.D.T Watson-Gandy, and N. Christofides. Distribution Management: Mathematical Modelling and Practical Analysis. Griffin, London, 1971.

(El Fallahi *et al.*, 2006) A. El Fallahi, C.Prins, and R.Wolfler Calvo. Un algorithme mémétique pour le problème de tournées de véhicules multi-compartiments. *6er Conférence de Modélisation et SIMulation – MOSIM'06*, 2006.

(Erbao et Mingyong, 2009) C. Erbao, and L.Mingyong . A hybrid differential evolution algorithm to vehicle routing problem with fuzzy demands. *Journal of computational and Applied Mathmatic*, 231(1), pages 302-310, 2009.

(Erschler et Grabot, 2001) J. Erschler, and B. Grabot. *Organisation et gestion de la production*. Hermès, Lavoisier, 2001.

(Eymery, 1998) P. Eymery. Enjeux de la logistique = The stakes of the logistics. *Journal Techniques de l'ingénieur, L'Entreprise industrielle* ISSN 1282-9072 . vol. AGL2, pages A9020.1 - A9020.8, 1998.

(Falkenauer, 1998) E. Falkenauer. *Genetic algorithms und grouping problems*. Wiley, Chichester, 1998.

(Feo et Resende, 1989) T.A. Feo, and M.G.C. Resende. A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8, pages 67-71, 1989.

(Feo *et al.*, 1994), T.A. FeoEO, M.G.C. Resende, and S.H. Smith. A greedy randomized adaptive search procedure for maximum independent set. Operations Research, 42, pages 860-878, 1994.

(Feo et Resende, 1995) T.A. Feo and M.G.C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6, pages 109-133, 1995.

(Ferreira, 2006) C. Ferreira. *Gene Expression Programming: Mathematical Modeling by an Artificial Intelligence*. 2nd Edition, Springer-Verlag, Germany, 2006.

(Festa et Resende, 2002) P. Festa, and M. G. C. Resende. GRASP: An annotated bibliography. *In C. C. Ribeiro and P. Hansen (eds), Essays and Surveys on Metaheuristics*, pages 325–367, Kluwer Academic Publishers, 2002.

(Fisher, 1995) M.L. Fisher. *Vehicle Routing*. Handbooks in Operations Research and Management Science 8, North Holland, Amsterdam, pages 1–33, 1995.

(Fisher et Jaikumar, 1981) M.L. Fisher, and R. Jaikumar. A generalized assignment heuristic for the vehicle routing problem. *Networks*, 11, pages 109-124, 1981.

(Fleszar *et al.*, 2009) K. Fleszar, I. H. Osman, and K. S. Hindi. A variable

neighbourhood search algorithm for the open vehicle routing problem. *EJOR*, 195(3), 2009.

(Fleurent et Ferland, 1994) C. Fleurent, and J. A. Ferland. Genetic hybrids for the quadratic assignment problem. In P. Pardalos, and H.Wolkowicz, , (eds.), Quadratic Assignment and Related Problems 16, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, pages 173-187, 1994.

(Fogel, 1962) L. J. Fogel. Toward inductive inference automata. *In Proceedings of the International Federation for Information Processing Congress, Munich*, pages 395–399, 1962.

(Fogel, 1994) D. B. Fogel. An introduction to simulated evolutionary optimization. *IEEE Trans. Neural Netw*, 5(1), pages 3–14, 1994.

(Fogel, 1995) D. B. Fogel. A Comparison of Evolutionary Programming and Genetic Algorithms on Selected Constrained Optimization Problems. *Simulation*, 64(6), pages 399-406, 1995.

(Fogel *et al.*, 1966) L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial Intelligence through Simulated Evolution*. Wiley, New York,1966.

(Foster et Ryan, 1976) B.A. Foster, and D.M. Ryan. An integer programming approach to the vehicle scheduling problem. *Operations Research*, 21, pages 361-384, 1976.

(Fourer *et al.*, 2003) R Fourer, D. Gay, B. Kernighan. *AMPL: A Modeling Language for Mathematical Programming*. Brooks/Cole -Thomson Learning, Pacific Grove, CA, 2003.

(Francis et Smilowitz, 2006) P. Francis, and K. Smilowitz. Modeling Techniques for Periodic Vehicle Routing Problems. *Conference Transportation Research Board*, 2006

(Freisleben et Merz, 1996) B. Freisleben, and P. Merz. New genetic local search operators for the travelling saleman problem. *Lecture Notes in Computer Science* 1141, pages 890-899, Springer-Verlag, 2006.

(Freitas, 2002). A. Freitas. *Data mining and knowledge discovery with evolutionary algorithms.* Springer, 2002 .

(Fu *et al.*, 2003) Z. Fu, R.W. Eglese, and L. Li. A tabu search heuristic for the open vehicle routing problem. Technical Report 2003/042, Lancaster University Management School, Lancaster, United Kingdom, 2003.

(Gajpal et Abad, 2009) Y. Gajpal , and P. Abad. An ant colony system (ACS) for vehicle routing problem with simultaneous delivery and pickup. *Computers and Operations Research*, 36 (12), pages 3215-3223, December, 2009.

(Galasso, 2007) F. Galasso. *Aide à la planification dans les chaînes logistiques en présence de demande Flexible*. Thèse INP Toulouse, France, 2007.

(Gambardella *et al.*, 2003) L. M. Gambardella, N. Casagrande, R. Mantemanni, F. Oliverio, and A.E. Rizzoli. Ant Colony Optimization applied to industrial vehicle routing problems. *ECCO, XVI Conference of the European Chapter on Combinatorial Optimisation, Molde, Norway*, 5 - 7, June 2003.

(Gaskell, 1967) T. Gaskell. Bases for vehicle fleet scheduling. *Operational Research Quarterly* , 18, pages 281-295, 1967.

(Gendreau *et al.*, 1994) M. Gendreau, A. Hertz, and G. Laporte .A tabu search heuristic for the vehicle routing problem. *Manag. Sci.*, 40, pages 1276-1290, 1994.

(Gendreau *et al.*, 1996) M. Gendreau, G. Laporte, and R. Seguin. Stochastic Vehicle Routing. European *Journal of Operational Research*, 1996.

(Gendreau *et al.*, 1997) M. Gendreau, G. Laporte, and J.-Y. Potvin. Vehicle routing: Modern heuristics. *J. K. Lenstra, and E. H. L. Aarts (eds). Local Search in Combinatorial Optimization. John Wiley and Sons, Chichester, UK*, pages 311–336, 1997.

(Gendreau *et al.*, 1999) M. Gendreau, G. Laporte, C. Musaraganyi, and E.D. Taillard. A Tabu Search Heuristic for the Heterogeneous Fleet Vehicle Routing Problem. *Computers and Operations Research*, 26, pages 1153–1173,1999.

(Gendreau *et al.*, 2002) M. Gendreau, G. Laporte, and J.-Y. Potvin. Metaheuristics for the capacitated VRP. *P. Toth, and D. Vigo (eds). The Vehicle Routing Problem. SIAM Series on Discrete Mathematics and Applications, SIAM, Philadelphia*, 129–154. 2002.

(Gendreau *et al.*, 2003) M.Gendreau, G. Laporte, J.-Y. Potvin, and F. Semet. Métaheuristiques pour le problème des tournées de véhicules. *M. Pirlot, and J. Teghem (eds). Résolution de Problèmes de RO par les Métaheuristiques. Hermès, Paris*, pages 49–70, 2003.

(Gendreau *et al.*, 2008) M. Gendreau , J.-Y. Potvin, O. Bräysy, G. Hasle, and A. Løkketangen. Metaheuristics for the vehicle routing problem and extensions:A categorized bibliography. *B. L. Golden, S. Raghavan, and E. Wasil (eds). The Vehicle Routing Problem- Latest Advances and New Challenges. Springer, New York*, pages 143–170, 2008.

(Génin, 2003) P. Génin. *Planification tactique robuste avec usage d'un APS. Proposition d'un mode de gestion par plan de référence*. Thèse de Doctorat, École des Mines de Paris, 2003.

(Génin *et al.*, 2005) P. Génin, S. Lamouri, and A. Thomas. Planification industrielle et ses limites. *Techniques de l'ingénieur, chapitres de livres*, 2005.

(Ghiani *et al.*, 2003) G. Ghiani, F. Guerriero , G. Laporte, and R. Musmanno. Real-Time Vehicle Routing : Solution Concepts, Algorithms and Parallel Computing Strategies. *European Journal of OR*, 151, pages 1–11, 2003.

(Giard et Mendy, 2006) V. Giard, and G. Mendy. Amélioration de la synchronisation de la production sur une chaîne logistique. *Revue Française de Gestion Industrielle*, 25(1), 2006.

(Gillett et Miller, 1974) B.E. Gillett, and L.R. Miller. A heuristic algorithm for the vehicle dispatch problem. *Operations Research*, 22, pages 340-349, 1974.

(Glover, 1977) F. Glover. Heuristics for integer programming using surrogate constraints. *Decision Sciences*, 8, pages 156-166, 1977.

(Glover, 1986) F. Glover. Future paths for integer programming and links to artificial intelligence. *Comput. Oper. Res*, 13, pages 533–549, 1986.

(Glover *et al.*, 1992) F. Glover, E. Taillard and D. De Werra. A User's Guide to Tabu search. Rapport technique N ORWP 92101, department de Mathématiques, Ecole Polytechnique Féderal de Lausane, Switzerland, 1992.

(Glover, 1998) F. Glover. A template for scatter search and path relinking. *In J.K. Hao, E.Lutton, E. Ronald, M. Schoenauer, and D. Snyers (Eds.). Artificial Evolution'97, Lecture Notes in Computer Sciences*, 1363, pages 3-51, Spinger-Verlag, 1998.

(Glover *et al.*, 2000) F. Glover, A. Løkketangen, and D.L. Woodruff. Scatter search to generate diverse MIP solutions. *In M. Laguna, and J.L.G. Velarde (Eds.). Computing Tools for Modeling Optimization and Simulation*, pages 299.320, Kluwer, Boston, 2000.

(Goldberg, 1989) D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley,1989.

(Goldberg et Lingle, 1985) D.E. Goldberg and R. Lingle. Alleles, loci and the traveling salesman problem. *In J.J. Grefenstette, editor, Proceedings of the First International Conference on Genetic Algorithms, Lawrence Erlbaum, Hillsdale, NJ*, pages 154-159, 1985.

(Golden *et al.*, 1977) B.L. Golden, T.L. Magnanti, and H.Q. Nguyen. Implementing vehicle routing algorithms. *Networks*, 7, pages 113-148, 1977.

(Golden *et al.*, 1982) B.L. Golden, A. Assad, L. Levy and F. Gheysens. The fleet size and mix routing problem. Management Science and Statistics . *Working paper, University of Maryland at college park*, N 82-020, 1982.

(Golden *et al.*, 1995) B.L. Golden, I.M. Chao, and E. Wasil. An Improved Heuristic for the Period Vehicle Routing Problem. *Networks*, 25-44, 1995.

(Golden *et al.*, 1998) B.L. Golden, E.A. Wasil, J.P. Kelly, and I.M. Chao. The impact of metaheuristics on solving the vehicle routing problem: algorithms, problem sets, and computational results. *In: Crainic TG, Laporte G, editors. Fleet management and logistics. Dordrecht: Kluwer*, pages 33–56, 1998.

(Golden *et al.*, 2008) B. L. Golden, S. Raghavan, and E. Wasil. The Vehicle Routing Problem- Latest Advances and New Challenges. *Springer, New York*, pages 143–170, 2008.

(Goupy et Creighton, 2009) J. Goupy, and L. Creighton. Introduction aux plans d'expériences : avec applications. Jacques Goupy, Lee Creighton. - 3e édition. - Paris : Dunod : L'Usine nouvelle- IX-334 pages, 2009.

(Gourgand *et al.*, 2007) M. Gourgand, N. Grangeon, D. Lemoine, and S. Norre . Couplage de modèles pour une planification tactique robuste: Application à la planification dans un bloc opératoire. *CPI'2007, 5ème Colloque International Conception et Production Intégrées, Rabat, Maroc*, 2007.

(Grabot et Letouzey, 2000) B. Grabot, and A. Letouzey. Short-term manpower management in manufacturing systems: new requirements and DSS prototyping. *Computers in Industry*, 43(1), pages 11-29, 2000.

(Graham *et al.*, 1979) R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G. Rinnooy Kan. Optimization and Approximation in Deterministic Sequencing and Scheduling Theory:A Survey. *Annals of Discrete Mathematics*, 5, pages 287–326, 1979.

(Grefenstette, 1986) J. Grefenstette. Optimization of control parameters for genetic algorithms. *IEEE Transactions on Systems, Man and Cybernetics, SMC*, (16), 122–128, 1986.

(Gueguen, 1999) C. Gueguen. *Méthodes de résolution exacte pour les problèmes de tournées de véhicules*. Thèse de Doctorat, Laboratoire Productique Logistique, Ecole Centrale Paris, 1999.

(Hansen et Mladenovic, 1999) P. Hansen, and N. Mladenovic. An introduction to variable neighborhood search. *In S. Vob, S. Martello, I. H. Osman and C. Roucairol (eds), Metaheuristic : Advances and Trends in Local Search Paradigms for Optimization, Kluwer*, pages 422–458, 1999.

(Hao *et al.*, 1999) J-K Hao, P. Galinier, and M. Habib. Méthaheuristiques pour l'optimisation combinatoire et l'affectation sous contraintes. *Revue d'Intelligence*, 1999.

(Hayari *et al.*, 2003) N. Hayari, M.A. Manier, C. Bloch, and A. El Moudni. Un Algorithme Evolutionniste pour le problème de Tournées Sélectives avec Contraintes de Fenêtres de Temps. *Conférence de Modélisation et Simulation : Organisation et conduite d'activités dans l'industrie et les services -MOSIM'03, Toulouse, France*, 2003.

(Hippolyte *et al.*, 2007) J.L. Hippolyte, C. Bloch, P. Chatonnay, C. Espanet, and D. Chamagne, and G. Wimmer. Tuning an Evolutionary Algorithm with Taguchi Methods. *In Richard Chbeir, Youakim Badr, Ajith Abraham, Dominique Laurent, and Fernando Ferri, editors, CSTST'08, 5th Int. Conf. on Soft Computing as Transdisciplinary Science and Technology*, pages 265-272, 2008.

(Ho et Ji, 2003) W. Ho, and P. Ji. Component scheduling for chip shooter machines: a hybrid genetic algorithm approach. *Computers and Operations Research*, 30, pages 2175–2189, 2003.

(Ho et Ji, 2004) W. Ho, and P. Ji. A hybrid genetic algorithm for component sequencing and feeder arrangement. *Journal of Intelligent Manufacturing*, 15, pages 307–315, 2004.

(Ho *et al.*, 2008) W. Ho, G.T.S. Ho, P. Ji, and H.C.W. Lau. A hybrid genetic algorithm for the multi-depot vehicle routing problem. *Engineering Applications of Artificial Intelligence*, 21(4), pages 548-557, 2008.

(Hoff et Løkketangen, 2006) A. Hoff, and A. Løkketangen. Creating lassosolutions for the traveling salesman problem with pickup and delivery by tabu search. *Central European Journal of Operations Research*, 14, pages 125-140, 2006.

(Holland, 1975) J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, 1975.

(Housroum *et al.*, 2005) H. Housroum, T. Hsu, R. Dupas, and G. Goncalves. Une approche génétique "en ligne" pour la gestion de tournées dynamique. *Conférence MHOSI'2005, Méthodologies et Heuristiques pour l'Optimisation des Systèmes Industriels*, 24-26 Avril, 2005.

(Huang *et al.*, 2003) G. Q. Huang, J. S .K. Lau, and K. L. Mak. The impacts of sharing production information on supply chain dynamics: a review of the literature.International Journal of Production Research, 41(7), pages 1483-1517, 2003.

(Hwang 2005) H-S Hwang. An integrated distribution routing model in multi-supply center system. *International Journal of Production Economics*, 98(2), pages 136-142, 2005.

(Jason et Konstantinos, 2002) G. D. Jason, and G. M. Konstantinos . An experimental study of benchmarking functions for Genetic Algorithms. *International Journal Computer Math*, 79(4), pages 403–416, 2002.

(Javel, 2004) G. Javel. *Organisation et gestion de la production: cours avec exercices corrigés*. 3e édition, Paris : Dunod, 2004.

(Jeong *et al.*, 1997) E-Y. Jeong, S-C. Oh , Y-K. Yeo , K-S. Chang, J-Y. Chang, and KS. Kim. Application of traveling salesman problem (TSP) for decision of optimal production sequence. *Korean Journal of Chemical Engineering*, 14(4), pages 1997.

(Jih et Hsu, 2004) W. R. Jih, and Y. J. Hsu, A Family Competition Genetic Algorithm for the Pickup and Delivery Problems with Time Window. *Bulletin of the College of of Engineering, N.T.U.*, No. 90, pages 89–98, 2004.

(Jog et Gucht, 1987) P. Jog, and G. Gucht. Parallelisation of probabilistic sequential search algorithms. *Proc. Of ICGA'87, San Mateo: Morgan*

Kaufmann, pages 170-176, 1987.

(Jog *et al.*, 1991) P. Jog, J. Suh, and G. Gucht. Parallel genetic algorithms applied to the traveling salesman problem. *SIAM Journal on Optimization*, 1(4), pages 515-529, 1991.

(Johnson et McGeoch, 1997) D. S. Johnson, and L. A. Mcgeoch. The travelling salesman problem: a case study. *In Local Search in Combinatorial Optimization, E. Aarts and J. Lenstra, Eds. Wiley, New York*, pages 215-310, 1997.

(Kilby *et al.*, 1997) P. Kilby, P. Prosser, and P. Shaw. Guided local search for the vehicle routing problem. *Proceeding of the 2nd International Conference on Mataheuristics (MIC97), Sophia-Antipolis, France, pages 21-24, 1997.*

(Kilby *et al.*, 2000) P.J Kilby., P. Prosser, and P. Shaw. A comparison of traditional and constraintbased heuristic methods on vehicle routing problems with side constraints. *Journal of Constraints*, 5(4), pages 389-414, 2000.

(Kirkpatrick *et al.*, 1983) S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220 (4598), pages 671-680, 1983.

(Kolen et Pesch, 1994) A.W. J. Kolen, and E. Pesch. Genetic local search in combinatorial optimization. *Discrete applied mathematics and combinatorial operations research and computer science*, 48(3), pages 273–284, 1994.

(Kontoravdis et Bard, 1995) G. Kontoravdis, and J.F. Bard. Improved heuristics for the vehicle routing problem with time windows. *ORSA Journal on Computing*, 7, 1995.

(Koza, 1992) J.R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press. ISBN 0-262-11170-5, 1992.

(Ku et Mak, 1998) K. W. C. Ku, and M. W. Mak. Empirical analysis of the factors that affect the Baldwin effect. *In A. E. Eiben, T. Bäck, , M. Schonauer, , and H.-P. Schwefel(eds). Parallel Problem Solving from Nature . PPSN V, Berlin, Springer*, pages 481-490, 1998.

(Kumar et Zhonghui, 2003) R. Kumar, and L. Zhonghui. Optimizing the operation sequence of a chip placement machine using TSP model. IEEE transactions on electronics packaging manufacturing, 26(1), pages 14-21, 2003.

(Kursawe, 1992) F. Kursawe. Evolution strategies for vector optimization. *In G-H Tzeng, and P-L Yu (eds). Preliminary Proc. Tenth Int'l Conf. Multiple Criteria Decision Making, National Chiao Tung University, Taipei*, pages 187-193, 1992.

(Kursawe, 1993) F. Kursawe. Evolution strategies: Simple models of natural processes?. *Revue Internationale de Systëmique*, 7(5), pages 627-

642, November, 1993.

(Labadi *et al*., 2008) N. Labadi, C. Prins, and M. Reghioui. A memetic algorithm for the vehicle routing problem with time windows. *RAIRO-Operations research*, Revue SCI, 42(3), pages 415-431, 2008.

(Lacomme *et al*., 2005) P. Lacomme, C. Prins, and W. Ramdane-Cherif. Evolutionary algorithms for periodic arc routing problems. *European Journal of OR*, 165, pages 535-553, 2005.

(Lacomme et Prins, 2006) P. Lacomme, and C. Prins. Algorithmes de découpage pour les problème de tournées de véhicules. *6e Conférence Francophone de MOdélisation et SIMulation - MOSIM'06, Maroc*, 3 - 5 avril, 2006.

(Laguna *et al*., 1994) M. LAGUNA, T.A. FEO, H.C. ELROD, A greedy randomized adaptive search procedure for the two-partition problem. Operations Research 42 : 677-687, 1994.

(Laguna, 2002) M. Laguna. Scatter Search. *In: P.M. Pardalos, and M.G.C. Resende (eds). Handbook of Applied Optimization, Oxford University Press,* pages 183–193, 2002.

(Land et Doig, 1960) A. H. Land, and A. G. Doig. An Automatic Method of Solving Discrete Programming. Problems. *Econometrica*, 28, pages 497-520, 1960.

(Laporte, 1992) G. Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59, pages 345-358, 1992b.

(Laporte et Osman, 1995) G. Laporte, and I.H. Osman. Routing Problems: A Bibliography . *Annals of Operations Research*, 61, pages 227-262, 1995.

(Laporte *et al*., 1986) G. Laporte , H. Mercure, and Y. Nobert . A exact algorithm for the asymmetrical capacitated vehicle routing problem . *Networks*, 16, pages 33–46, 1986.

(Laporte et Nobert, 1987) G. Laporte, and Y. Norbert. Exact algorithms for the vehicle routing problem. *Annals of Discrete Mathematics*, 31, pages 147-184, 1987

(Laporte *et al*., 2000) G. Laporte, M. Gendreau, J.-Y. Potvin, and F. Semet. Classical and modern heuristics for the vehicle routing problem. *International Transaction in Operational Research*, 7(4–5), pages 285-300, 2000.

(Lau *et al*., 2003) C. Lau, M. Sim, Kwong, and M. Teo. Vehicle routing problem with time windows and a limited number of vehicles. *European Journal of Operational Research*, 148(3), pages 559-569, 2003.

(Le Bouthillier 2000) A. Le Bouthillier. *Modélisation UML pour une architecture cooperative appliquée au problème de tournées de véhicules avec fenêtres de temps*. Mémoire de maîtrise, Université de Montréal,

Département d'informatique et de recherche opérationnelle, 2000.

(Lee *et al*. 1993) H.L. Lee, and C. Billington. Material management in decentralized supply chain. *Operations Research*, 41(5), pages 835-847, 1993.

(Lemoine, 2008) D. Lemoine. Modèles génériques et méthodes de résolution pour la planification tactique mono-site et multi-site. Thèse, Université Blaise Pascal, Clermont-Ferrand, 2008.

(Li et Lim, 2001) H. Li, and A. Lim. A metaheuristic for solving the Pickup und Delivery Problem with Time Windows. *In: IEEE Computer Society (ed) Proceedings of the 13th IEEE International Conference on Tools with Artificial Intelligence (ICTAI),* pp 160–167. IEEE Computer Society, Los Alamitos, CA, 2001.

(Libralao *et al.*, 2005) G. L. Libralao, F. C. Pereira, T. W. Lima, and A. C. B. Delbem. Node-Depth Encoding for Evolutionary Algorithms Applied to Multi-vehicle Routing Problem. *International conference on Innovations in Applied Artificial Intelligence*, pages 557-559, 2005 .

(Lin,1965). S. Lin. Computer solutions of the traveling salesman problem. *Bell System Technical Journal*, 44, pages 2245-2269, 1965.

(Ling *et al*., 2002) Q. Ling, H. Wen-Jing, H. Shell-Ying, and W. Han. Scheduling and routing algorithms for AGVs: a survey. *Int. j. prod. Res*, 40(3), pages 745-760, 2002.

(Liu et Shen, 1999) F.H. Liu, and S.Y. Shen. A Method for VRP with multiple vehicle types and time windows. *Journal of the Operational Research Society*, 50(7), pages 721-732, 1999.

(Liu *et al*., 2008) S. Liu, J-M. Pinto, L-G. Papageorgiou. A TSP-based MILP Model for Medium-Term Planning of Single-Stage Continuous Multiproduct Plants. *Ind. Eng. Chem. Res*, 47, pages 7733–7743, 2008.

(Lundy et Mees, 1986) S. Lundy, and A. Mees. Convergence of an annealing algorithm. *Mathematical Programming, 34, pages 111-124, 1986.*

(Louis *et al*., 1999) S. Louis , X. Yin, and Z. Yuan. Multiple vehicle routing with time windows using genetic algorithms. *Evolutionary Computation*, 3, pages 1804–1808, 1999.

(Manier et Bloch, 2003) M.A. Manier, and C. Bloch. A classification for Hoist Scheduling Problems. *International Journal of Flexible Manufacturing Systems.* 15(1), pages 37- 55, Kluwer Academic Publishers, ISSN 0920-6299, 2003.

(Martin et Otto, 1992) O. Martin, S.W. Otto, and E.W. Felten. Large-step Markow chains for the TSP: Incorporating local search heuristics. *Operations Research Letters*, 11, pages 219-214, 1992.

(Mathias et Whitley, 1992) K. Mathias, and D. Whitley. Genetic operators, the fitness landscape and the traveling salesman problem. *In R. Männer and B.*

Manderick B. Manderick, editors, Parallel Problem Solving from Nature, 2, Amsterdam, pages 219-228, 1992. Elsevier Science Publishers.

(Mentzen *et al*., 2001) J. T. Mentzer, W. Dewitt, J.S. Keebler, S. Min, N.W. Nix, C.D. Smith, Z.G. Zacharia. Defining supply chain management. *Journal of business logistics Management*, 22 (2), pages 1-25, 2001.

(Mendoza *et al*., 2009) J. E. Mendoza, B.Castanier, C. Guéret, A.L. Medaglia, and N.Velasco. Constructive heuristics for the multi-compartment vehicle routing problem with stochastic demands. Technical report, Universidad de los Andes, 2009.

(Merz, 2000) P. Merz. *Memetic algorithms for combinatorial optimization problems: Fitness landscapes and effective search strategies*. PhD thesis, Department of Electrical Engineering and Computer Science, University of Siegen, Germany, 2000.

(Merz et Freisleben, 1997) P. Merz, and B. Freisleben. Genetic Local Search for the TSP: New Results. *International Conference on Evolutionary Computation (ICEC 97), N. F. F. Ebecken (eds), Wit Press/Computational Mechanics Publications, Boston*, 1997.

(Mester et Bräysy, 2007) D. Mester, and O. Bräysy. Active-guided evolution strategies for large-scale capacitated vehicle routing problems. *Computers and Operations Research*, 34(10) , Pages 2964-2975, 2007.

(Metropolis *et al*., 1953) W. Metropolis, A. Roenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equation of the state calculations by fast computing machines. *Journal of Chemical Physics*, 21, pages 1087-1092, 1953.

(Michalewicz, 1994) Z. Michalewicz . *Genetic Algorithms + Data Structures = Evolution Programs*. Berlin : Springer-Verlag, seconde édition, 1994.

(Michalewicz et Michalewicz, 1997) Z. Michalewicz, and M. Michalewicz. Evolutionary computation techniques and their applications. *In Proceedings of the IEEE International Conference on Intelligent Processing Systems, Beijing, China. Institute of Electrical & Electronics Engineers, Incorporated*, pages 14–24, 1997.

(Miller, 1995) D.L. Miller. A matching based exact algorithm for capacitated vehicle routing problems. *ORSA Journal on Comupting*, 7, pages 1–9, 1995.

(Miller, 2001) T. Miller. *Hierarchical operations and supply chain planning*, Springer, 2001.

(Miller et Goldberg, 1995) B.L. Miller, and D. E. Goldberg. Genetic Algorithms, Tournament Selection, and the Effects of Noise. *Complex System*, 9(3), 193-212, 1995.

(Mingozzi, 2005) A.Mingozzi. The multi-depot periodic vehicle routing problem. *Book Chapter (Abstraction, Reformulation and Approximation)*, pages 347-350, 2005.

(Mladenovic et Hansen, 1997) N. Mladenovic, and P. Hansen. Variable neighborhood search. *Computers and Operations Research*, 24, pages 1097-1100, 1997

(Monkman *et al*., 2008) S. K. Monkman, D. J. Morrice b, J. F. Bard, A production scheduling heuristic for an electronics manufacturer with sequence-dependent setup costs. *European Journal of Operational Research*, 187, pages 1100–1114, 2008

(Morris *et al*., 1998) G. M. Morris, D. S. Goodsell, R. S. Halliday, R. Huey, W.E. Hart, , R. K. Belew, and A. J. Olson. Automated docking using a Lamarckian genetic algorithm and an empirical binding free energy function. *Journal of Computational Chemistry*, 19(14), 1998.

(Moscato, 1989) P. Moscato. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Technical Report 05.20 02.50 87.10, California Institute of technologie, Pasadena, CA 91125, U.S.A., 1989.

(Mosheiov, 1998) G. Mosheiov. Vehicle routing with pick-up and delivery: Tour-partitioning heuristics. *Computers and Industrial Engineering*, 34, Pages 669–684, 1998.

(Mülhenbein, 1993) H. Mühlenbein. *Evolutionary Algorithms Theory and applications*. Wiley, 1993.

(Mühlenbein et Schlierkamp-Voosen, 1993) H. Mühlenbein, and D. Schlierkamp-Voosen. Predictive Models for the Breeder Genetic Algorithm, continuous Parameter Optimization. *Evolutionary computation*, 25-49, 1993.

(Mülhenbein *et al*., 1988) H. Mülhenbein, M. Gorges-Schleuter, and O. Krämer. Evolution algorithms in combinatorial optimization. *Parallel Computing*, 7, pages 65-85, 1988.

(Najera, 2007) A.G. Najera .A *Genetic Algorithm for the Capacitated Vehicle Routing Problem*. School of Computer Science University of Birmingham, February 10, 2007

(Nanry et Barnes, 2000) W.P. Nanry, and J.W. Barnes. Solving the Pickup und Delivery Problem with TimeWindows using Reactive Tabu Search. *Transportation Research Part* B, 34, 107–121, 2000.

(Nelson *et al*., 1985) M.D. Nelson, K.E. Nygard, J.H. Griffin, and W.E. Shreve. Implementation techniques for the vehicle routing problem. *Computers and Operations Research*, 12, pages 273-283, 1985.

(Oliver *et al*., 1987) I. Oliver, D. Smith., and J Holland. A Study of Permutation Crossover Operators on The Traveling Salesman Problem. *Proc. 2nd Int. Conf. on Genetic Algorithms and Theirs Applications*, 1987.

(Ombuki *et al.*, 2002) B.M. Ombuki, M. Nakamura, and M. Osamu. A Hybrid Search

Based on Genetic Algorithms and Tabu Search for Vehicle Routing. Technical Report ,Brock University ,Department of Computer Science, St. Catharines,Ontario, Canada, 2002.

(Ombuki *et al*., 2006) B.M. Ombuki, B.J. Ross, and F. Hanshar. A Multi- Objective Genetic Algorithm Approach to the Vehicle Routing Problem with Time Windows. *Applied Intelligence*, 24(1), pages 17-30, 2006.

(Osman, 1993) I. H. Osman. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research*, 41, pages 421-451, 1993.

(Ozdemir, 2002) M. Ozdemir. *Evolutionary computing for feature selection and predictive data mining*. Thesis, Rensselaer Polytechnic Institute, Troy, New York, 2002.

(Padberg et Rinaldi, 1987) M. Padberg, and G. Rinaldi. Optimization of a 532 city symmetric traveling salesman problemby branch-and-cut. *Operations Research Letters*, pages 1–7, 1987.

(Paessens, 1988) H. Paessens. The savings algorithm for the vehicle routing problem. *European Journal of Operational Research*, 34, pages 336-344, 1988.

(Pankratz, 2005) G. Pankratz. A Grouping Genetic Algorithm for the Pickup and Delivery Problem with Time Windows. *OR Spectrum*, 27, pages 21-24, 2005.

(Pareto, 1896) V. Pareto. *Cours d'économie politique*. Université de Lusanne, 1896.

(Parragh *et al*., 2006a) S. N. Parragh, K.F. Doerner, and R. F. Hartl. A survey on pickup and delivery models Part I: Transportation between customers and depot. *Working paper, Chair of Production and Operations Management, University of Vienna*, 2006a.

(Parragh *et al*., 2006b) Parragh S. N., K.F. Doerner and R. F. Hartl, A survey on pickup and delivery models Part II: Transportation between pickup and delivery Locations. *Working paper, Chair of Production and Operations Management, University of Vienna*, 2006b.

(Pereira *et al*., 2002) F. Pereira , J. Tavares , P. Machado, and E. Costa. GVR : a New Genetic Representation for the Vehicle Routing Problem. *Artificial Intelligence and Cognitive Science : 13th Irish Conference Proceedings*, pages 95–102, 2002.

(Pitsoulis et Resende, 2002) L. Pitsoulis, and M. G. C. Resende. Greedy randomized adaptive search procedures. *In P. M. Pardalos and M. G. C. Resende (eds), Handbook of Applied Optimization*, pages 168–181, Oxford University Press, 2002.

(Poirier *et al*., 2001) C. Poirier, and S.E. Reiter. *La Supply Chain*. Dunod, 2001.

(Polacek *et al*., 2004) M. Polacek, R. F. Hart, K. Doerner, and M. Reimann. A Variable

Neighborhood Search for the Multi Depot Vehicle Routing Problem with Time Windows. *Journal of heuristics*, 10(6), 2004.

(Portmann, 1996) M-C. Portmann. Genetic algorithm and scheduling: a state of the art and some propositions. In Proceedings of the workshop on production planning and control, Mons, Belgium, pages I-XIV, 1996.

(Potvin, 1996) J.Y. Potvin. Genetic Algorithms for the Traveling Salesman Problem. *Annals of Operations Research*, 63, pages 339-370, 1996

(Potvin et Bengio, 1996) J.Y. Potvin, and S. Bengio. The vehicle routing problem with time windows--Part II: genetic search. *INFORMS. Journal on Computing*, 8, pages 165-72, 1996.

(Potvin *et al*., 1996) J.Y. Potvin, C. Duhamel, and F. Guertin. A genetic algorithm for vehicle routing with backhauling. *Applied Intelligence*, 6, pages 345-55, 1996.

(Potvin et Thangiah, 1999) J.Y. Potvin, and S. R. Thangiah. Vehicle routing through simulation of natural processes. *L. C. Jain, and N. M. Martin, (eds). Fusion of Neural Networks, Fuzzy Sets and Genetic Algorithms-Industrial Applications. CRC Press, Boca Raton, FL*, pages 141–168, 1999.

(Potvin, 2009) J.Y. Potvin. Evolutionary Algorithms for Vehicle Routing. *INFORMS Journal on Computing* 21(4), pages 518–548, 2009.

(Prins, 2004) C. Prins, A simple and effective evolutionary algorithm for the Vehicle Routing Problem. *Computers and Operations Research*, 31(12), pages 1985-2002, 2004.

(Prins et Bouchenoua, 2004) C. Prins, and S. Bouchenoua. A Memetic Algorithm Solving the VRP, the CARP and General Routing Problems with Nodes, Edges and Arcs. *Studies in fuzziness and soft computing*, 166, pages 65-86, 2004.

(Radcliffe et Surry, 1994) N. J. Radcliffe, and P. D. Surry. Formal Memetic Algorithms. *In Lecture Notes in Computer Science, New York: Springer-Verlag*, pages 1-16, 1994.

(Ralphs, 2003) T. K. Ralphs . Parallel branch and cut for capacitated vehicle routing. *Parallel Computing archive*, 29 (5), pages 607–629, 2003.

(Ramdane Cherif, 2002) W. Ramdane Cherif. *Problèmes d'Optimisation en Tournées sur Arcs*. Thèse de l'Université de Technologie de Troyes, France, 2002.

(Rechenberg, 1965) I. Rechenberg. *Cybernetic Solution Path of an Experimental Problem*. Royal Aircraft Establishment Library Translation, 1965.

(Rechenberg, 1973) I. Rechenberg. Evolution strategie. *Optimierung technischer systeme nach prinzipien der biologischen evolution, Frommann-Holzboog*, 1973.

(Reeves, 1993) C. R. Reeves. *Modern Heuristic Techniques for Combinatorial Problems*. Blackwell Scientific Publishing, Oxford, England, 1993.

(Reeves et Rowe, 2002) C. R. Reeves, and J. E. Rowe. Genetic Algorithms: Principles and Perspectives . *A Guide to GA Theory. Kluwer Academic Publishers, Boston (USA)*, 2002.

(Rego *et al*., 1994) C. Rego, and C. Roucairol. Le Problème de Tournées de Véhicules : Etude et Résolution Approchée. Technical Report, inria, Février, 1994.

(Rego *et al*., 1996) C. Rego, and C. Roucairol, .A Parallel Tabu Search Algorithm using Ejection Chains for the Vehicle Routing Problem. *In I.H. Osman and J.P. Kelly (eds), MetaHeuristics: Theory and Applications. Kluwer, Boston*, 1996.

(Reinelt, 1994) G. Reinelt. *The Traveling Salesman: Computational Solutions for TSP Applications*. Springer, New York, 1994.

(Renaud *et al*., 1996) J. Renaud, F.F. Boctor, and G. Laporte. An Improved Petal Heuristic for the Vehicle Routeing Problem. *The Journal of the Operational Research Society*, 47, pages 329-336, 1996.

(Repoussis *et al*., 2010) P. P. Repoussis, C. D. Tarantilis, O. Bräysy, and G. Ioannou. A hybrid evolution strategy for the open vehicle routing problem .*Computers and Operations Research*, 37(3), Pages 443-455, 2010.

(Resende et Ribeiro, 2003) M. G. C. Resende, and C. C. Ribeiro. Greedy randomized adaptive search procedures. *In F. Glover and G. Kochenberger, editors, Handbook of Metaheuristics*, pages 219–249, Kluwer Academic Publishers, 2003.

(Rinnooy Kan, 1976) A.H.G. Rinnooy Kan. *Machine Scheduling Problems: Classification, Complexity and Computations*. Nijhoff, The Hague, 1976.

(Rizzoli *et al*., 2007) A .E. Rizzoli, R. Montemanni, E. Lucibello, and L .M. Gambardella . Ant colony optimization for real-world vehicle routing problems. *Swarm Intelligence1*,(2), pages 135-151, 2007.

(Rochat, et Taillard, 1995) Y. Rochat, and E.D. Taillard. Probabilistic Diversification and Intensification in Local Search for Vehicle Routing. *Journal of Heuristics*, 1, 147-167, 1995.

(Rodriguez-Tello *et al.*, 2005) E. Rodriguez-Tello, J. Hao, and J. Torres-Jimenez. A comparison of memetic recombination operators for the minla problem. *MICAI 6*, 392–403, 2005.

(Ropke et Pisinger, 2006) S. Ropke and D. Pisinger. A unified heuristic for a large class of vehicle routing problems with backhauls. *European Journal of OR*, 171, 750-775, 2006.

(Rota, 1998) K. Rota. *Coordination temporelle de centres gérant de façon autonome des ressources: Application aux chaînes logistiques*

intégrées en aéronautique. Thèse de doctorat de l'ENSAE, 1998.

(Rota *et al*., 2001) K. Rota, C. Thierry and G. Bel. *La maîtrise des flux* (coordonateur : J.P.Campagne) : Chapitre 5 "Gestion des flux dans les chaînes logistiques (Supply Chain Management)", Ouvrage Hermès : Traité Systèmes pour l'ingénieur, Ed. Hermès, 2001.

(Ryan *et al*., 1993) D. M. Ryan, C. Hjorring, and F. Glover. Extensions of the petal method for vehicle routing. *Journal of Operational Research Society*, 44, pages 289-296, 1993.

(Schaffer *et al.*, 1989) J. Schaffer, R. Caruana, L. Eshelman, and R. Das. A study of control parameters affecting online performance of genetic algorithms for function optimization. *Proceedings of the third international conference on Genetic Algorithms,* 51–60,1989.

(Scheuerer et Wendolsky, 2006) S. Scheuerer, and R. Wendolsky. A scatter search heuristic for the capacitated clustering problem. *European Journal of Operational Research*,169(2), pages 533-547, 2006.

(Schoenauer et Michalewicz, 1997) M. Schoenauer and Z. Michalewicz. Evolutionary Computation. *Journal Control and Cybernetics*, 26, pages 307-338, 1997.

(Schwefel, 1977) H. P. Schwefel. *Numerishe Optimierung Von Computer-modellen mittels der evolution strategie. Interdisciplinary Systems research*, (26), 1977.

(Schwefel, 1981) H.P. Schwefel. Numerical Optimization of Computer Models. John Wiley & Sons, Inc., New York, NY, 1981.

(Schwefel, 1984) H-P. Schwefel. Evolution strategies: A family of non-linear optimization techniques based on imitating some principles of organic evolution. *Annals of Operations Research*, 1, pages 165-167, 1984.

(Schwefel, 1987) H-P. Schwefel. Collective phenomena in evolutionary systems. *In P. Checkland and I. Kiss (eds). Problems of Constancy and Change - the Complementarity of Systems Approaches to Complexity, Papers presented at the 31st Annual Meeting of the Int'l Soc. for General System Research 2. Budapest,* pages 1025-1033. *Int'l Soc. for General System Research,* 1987.

(Schwefel et Rudolph, 1995) H-P Schwefel, and G. Rudolph. Contemporary Evolution Strategies. Proceedings of the Third European Conference on Advances in Artificial Life, pages 893-907, 1995.

(Sevaux et Dauzere-Peres, 2000) M. Sevaux, and S. Dauzere-Peres. A Genetic Algorithm to Minimize the Weighted Number of Late Jobs on a Single Machine. *Working Paper, CNRS, UMR 8530, LAMIH/S*, 2000.

(Shapiro, 1998) J. Shapiro. Bottom-up versus top-down approaches to supply chain modelling . *In Quantitative Models for Supply Chain Management , Kluwer Academic Publishers, Dordrecht,* pages 739-759, 1998.

(Shapiro, 1999) J. Shapiro. On the connections among activity based costing,

optimization models for strategic decision support and the resource-based view of the firm. *European Journal of Operational Research,* 118 (2), pages 295-314, 1999.

(Shen *et al*., 2006) Z. Shen, F. Ordonez, and M.M. Dessouky. The minimum unmet demand stochastic vehicle routing problem. *Working paper 2006-07, University of Southern California*, 2006.

(Siarry *et al*., 2003) P. Siarry, J. Dréo, and A. Pétrowski. *Métaheuristiques pour l'optimisation difficile*. 355 pages, 2003.

(Solomon, 1987) M.M. Solomon. Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. *Operations Research*, 35, pages 254-265, 1987.

(Sörensen, 2003) K. Sorensen. *A framework for robust and flexible optimization using metaheuristics with applications in supply chain design*. PhD Thesis University of Antwerp, 2003.

(Spears *et al.*, 1993) W. M. Spears, K. A. D. Jong, T. Bäck., D. B. Fogel., and H. De Garis. An overview of evolutionary computation. *In Proceedings of the European Conference on Machine Learning (ECML- 93), P. B. Brazdil, Ed.667, Springer Verlag, Vienna, Austria*, pages 442–459,1993.

(Stadtler et Kilger, 2000) H. Stadtler, and C. Kilger. *Supply Chain Management and Advanced Planning*. Springer: Berlin, 2000.

(Stützle et Hoos, 2000) T. Stützle and H.H. Hoos. MAX MIN Ant System. *Future Generation Computer Systems*, 16, pages 889-914, 2000.

(Suerie, 2005) C. Suerie. Time Continuity in Discrete Time Models : New Approaches for Production Planning in Process Industries. *Lecture Notes in Economics and Mathematical Systems, Springer Berlin Heidelberg*, 2005.

(Syswerda, 1989) G. Syswerda. Uniform crossover in genetic algorithms. *J. D. Schaffer, ed. Proc. Third Internat. Conf. Genetic Algorithms, Morgan Kaufmann, San Mateo, CA*, 2–9, 1989.

(Syswerda, 1991) G. Syswerda. Schedule optimization using Genetic Algorithms. *Handbook of Genetic Algorithms, L. Davis (ed)*, 332-349, 1991.

(Taillard, 1993) E. D. Taillard. Parallel iterative search methods for vehicle routing problems. *Networks*, 23, pages 661-673, 1993.

(Taillard *et al*., 1997) E. D. Taillard, P. Badeau, M. Gendreau, F. Guertin, and J.-Y. Potvin. .A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Science*, 31, pages 170-186, 1997.

(Taillard *et al*., 1998) E. D. Taillard, M. Gambardella, M., Gendreau, and J.-Y. Potvin. Programmation à mémoire adaptative. Technical report IDSIA-79-97, IDSIA, Lugano, 1997. Published in: Calculateurs Parallèles,

Réseaux et Systèmes Répartis 10, pages 117-140, 1998.

(Talbi, 2002) E.G. Talbi. A taxonomy of hybrid metaheuristics. *Journal of Heuristics*, 8, pages 541.564, 2002.

(Talbi *et al*., 1998) E-G. Talbi, Z. Hafidi, J-M. Geib. Parallel adaptive tabu search for large optimization problems. *(à paraître) dans Parallel Computing*,1998.

(Tan *et al*., 2000) K. C. Tan, L. H. Lee, Q. L. Zhu, and K. Ou. Heuristic Methods for Vehicle Routing Problem with Time Windows. Artificial Intelligent in Engineering, pages 281-295, 2000.

(Tan, 2001) K.C. Tan. A framework of supply chain management literature. *European Journal of Purchasing and Supply Management*, 7, pp 39-48, 2001.

(Tang *et al*., 2010) J. Tang, J. Zhang, and Z Pan. A scatter search algorithm for solving vehicle routing problem with loading cost. *Expert Systems with Applications*, 37(6), pages 4023-4724, 2010.

(Tavares *et al*., 2003) J. Tavares, P. Machado, F.B. Pereira, and E. Costa. On the inuence of gvr in vehicle routing. *In Proceedings of the 2003 ACM Symposium on Applied Computing*, pages 753{758, New York, NY, USA, 2003.

(Tayur *et al*., 1999) S. Tayur, R. Ganeshan, and M. Magazine.Quantitative models for supply chain management. *Kluwer Academic Publishers*, 1999

(Thangiah *et al*., 1993) S. R. Thangiah , R.Vinayagamoorty , and A.V. Gubbi. Vehicle Routing and Time Deadlines Using Genetic and Local Algorithms. Proceedings of the 5th International Conference on Genetic Algorithms, pages 506-515, 1993.

(Thangiah, 1995) S. R. Thangiah. Vehicle routing with time windows using genetic algorithms. *In: L. Chambers (eds). Application handbook of genetic algorithms: new frontiers, vol. II. Boca Raton, FL: CRC Press*, pages 253-77. 1995.

(Theys *et al*., 2010) C. Theys, O. Bräysy, W. Dullaert, and B. Raa. Production, manufacturing and logistics using a TSP heuristic for routing order pickers in warehouses. *European Journal of Operational Research*, 200(3), pages 755-763, 2010.

(Thierens, 2009) D. Thierens. Adaptive Operator Selection for Iterated Local Search. *In T. Stützle, M. Birattari and H.H. Hoos (Eds.), Proceedings of the Second international Workshop on Engineering Stochastic Local Search Algorithms*, pages 140-144, Springer, 2009.

(Toth et Vigo, 2002a) P. Toth, and D. Vigo. The vehicle routing problem. *SIAM Monographs on Discrete Mathematics and Applications*, pages 129–154, 2002.

(Toth et Vigo, 2002b) P. Toth, and D. Vigo. Models, relaxations and exact approaches for the capacitated vehicle routing problem. *Discrete Applied*

Mathematics, 123, 487-512, 2002.

(Tounsi et Ouis, 2008) M. Tounsi, S. Ouis. An Iterative local-search framework for solving constraint satisfaction problem. *Applied Soft Computing*, 8 , Pages 1530-1535, 2008.

(Tsutsui *et al.*, 1997) S. Tsutsui, Y. Fujimoto, and A. Ghosh. Forking GAs : GAs with search space division schemes. Evolutionary Computation, 5(1), 61–80, 1997.

(Ulder *et al*., 1991) N.L.J. Ulder, E.H.L. Aarts, H.J. Bandelt, P.J.M.V. Laarhoven, and E. Pesch. Genetic local search algorithms for the traveling salesman problem. *Lecture Notes in Computer Science*, 496, 1991.

(Vacic et Sobh, 2002) V. Vacic, and T. M. Sobh. Vehicle Routing Problem with Time Windows. . *Working paper, Department of Computer Science and Engineering, University of Bridgeport, USA*, 2002.

(Valenzuela, 1995) C. L.Valenzuela. *Evolutionary Divide and Conquer : a novel genetic approach to the TSP*. These, Imperial College of Science, Department of Computing, University of London, England, 28 June, 1995.

(Verhoeven et Aarts, 1995) M.G.A. Verhoeven, and E.H.L. Aarts. Parallel local search. *Journal of Heuristics*, 1(1), pages 43-65, 1995.

(Vianna *et al*., 1999) D. S. Vianna, L.S. Ochi, and L. M.A. Drummond. A Parallel Hybrid Evolutionary Metaheuristic for the Period Vehicle Routing Problem. *IPPS/SPDP Workshops*, 183-191, 1999.

(Vose, 1999) M. Vose. *The Simple Genetic Algorithm: Foundations and Theory*. Complex Adaptive Systems. MIT Press, 1999.

(Vollman *et al*., 1997) T. E. Vollman, W.L. Berry, and D.C.Whybark. *Manufacturing planning and control systems* .4th edition, New York,1997.

(Voudouris et Tsang, 1995) C. Voudouris, and E. Tsang. Guided Local Search. Rapport technique CSM-247, Department of Computer Science, University of Essex, 1995.

(Voudouris et Tsang, 1999) C. Voudouris, and E.P.K. Tsang. Guided local search and its application to the travelling salesman problem. European Journal of Operational Research, 113, pages 469-499, 1999.

(Wade et Salhi, 2004) A. Wade and S. Salhi. An ant system algorithm for the mixed vehicle routing problem with backhauls. Metaheuristics: computer decision-making, Kluwer Academic Publishers, Norwell, MA, 2004.

(Waltz, 2004) R. Waltz. Knitro 4.0. *User manual, Ziena Optimization, Inc., Evanston, IL, USA,. October* 2004.(Available online at http://www.ziena.com/documentation.html).

(Womack *et al*., 1992) J. P. Womack, D. T. Jones, and D. Roos. Le système qui va

changer le monde : une analyse des industries automobiles mondiales dirigée par le Massachusetts Institute of Technology. *Dunod*,1992.

(Whitley, 1989) D. Whitley. The GENITOR Algorithm and Selective Pressure. *Proc 3d International Conf on Genetic Algorithms, Morgan Kaufmann.* Pages 116-121, 1989.

(Whitley, 1994) D. Whitley. A Genetic Algorithm Tutorial. *Statistics and Computing*, (4), pages 65-85, 1994 .

(Whitley *et al.*, 1991) D. Whitley, T. Starkweather, and D. Shaner. The Traveling Salesman and Sequence Scheduling: Quality Solutions Using Genetic Edge Recombination. *Handbook of Genetic Algorithms. L. Davis(ed), Van Nostrand Reinhold, New York*, 350-372,1991.

(Wineberg et Chen, 2004) M. Wineberg, and J. Chen. The shifting balance genetic algorithm as more than just another island model GA. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), pages 318–329, 2004.

(Wren, 1971) A. Wren. *Computers in Transport Planning and Operation*. Ian Allan, London. 1971.

(Wren et Holliday, 1972) A.Wren, and A. Holliday. Computer scheduling of vehicles from one or more depots to a number of delivery points. *Operations Research Quarterly*, 23, pages 333-344, 1972.

(Xu *et al.*, 1996) J. Xu, and J. P. Kelly. A network flow-based tabu search heuristic for the vehicle routing problem., *Transportation Science*, 30, pages 379-393, 1996.

(Yang et Zhuang, 2010) J. Yang, and Y. Zhuang. An improved ant colony optimization algorithm for solving a complex combinatorial optimization problem .*Applied Soft Computing*, 10(2), pages 653-660, 2010.

(Yellow, 1970) P. Yellow. A computational modification to the savings method of vehicle scheduling. *Operational Research Quarterly*, 21, pages 281-283, 1970.

(Yu *et al.*, 2009) B. Yu, Z. Z. Yang, and B.Yao. An improved ant colony optimization for vehicle routing problem. *Revue / Journal European journal of operational research*, 196(1), pages 171-176 ,2009.

(Zhan-feng *et al.*, 2009) W. Zhan-feng, D. Hai-lian, H. Ji-chao, and L.Guang-xia. An Improved Genetic Algorithm for Vehicle Routing Problem of Non-full Load. *Third International Symposium on Intelligent Information Technology Application, iita*, V 2, pages 173-175, 2009.

(Zhang et Tang, 2009) X. Zhang, and L. Tang . A new hybrid ant colony optimization algorithm for the vehicle routing problem. *Revue / Journal Pattern recognition letters*, 30(9) , pages 848-855, 2009.

(Zhao *et al*., 2009) F. Zhao , S. Li , J. Sun , and D. Mei. Genetic algorithm for the one-commodity pickup-and-delivery traveling salesman problem. *Computers and Industrial Engineering,* 56(4), pages 1642-1648, 2009.

(Zhen et Zhang, 2009) T. Zhen, and Q. Zhang. A Hybrid Metaheuristic Algorithm for the Multi-depot Vehicle Routing Problem with Time Windows. *NSWCTC '09, Wuhan, Hubei*, V 2, pages 798 – 801, 2009.

(Zhu, 1999) K.Q. Zhu. *Heuristic Methods For Vehicle Routing Problem with Time Windows*. PhD thesis, National University of Singapore, Departement of Electrical Engineering, May, 1999.

(Zhu, 2000) K. Q. Zhu. A New Genetic Algorithm for VRPTW. *International Conference on Artificial Intelligence, Las Vegas, USA*, 2000.

Résumé

Le VRP (ou problème de tournées de véhicules) est l'un des problèmes d'optimisation combinatoire les plus étudiés car il a de multiples applications en planification industrielle. La littérature associée est très volumineuse et très riche, en variantes de problèmes et en approches de résolution. Face à un problème réel, il est difficile d'identifier la classe de problème à laquelle il appartient, de recenser tous les travaux correspondants, et de déterminer le type de méthode de résolution le plus approprié.

Cette thèse étudie la faisabilité d'un projet destiné à faciliter ces démarches, en s'intéressant plus particulièrement aux approches de résolution évolutionnaires. Il repose sur trois éléments : une notation des variantes de VRP, un recensement d'opérateurs évolutionnaires de la littérature, et la construction d'une base de règles liant les variantes de problèmes à l'efficacité des opérateurs évolutionnaires. L'objectif est de guider la conception d'un algorithme en fonction des caractéristiques du problème, en proposant les opérateurs qui ont la plus grande probabilité d'être efficaces.

Appliquer la notation proposée à plusieurs articles montre qu'elle permet à chacun de classifier les travaux de manière précise dans la littérature, et d'identifier ainsi plus facilement les approches et résultats comparables aux siens.

La méthodologie expérimentale proposée pour déduire des règles liant les caractéristiques d'un problème à l'efficacité d'opérateurs évolutionnaires est illustrée en considérant 3 types de croisement et 3 types de mutation. Cette étude montre qu'il est possible d'estimer quels éléments de l'algorithme ont un impact détectable sur les performances, et d'établir des relations entre les choix de conception de l'algorithme (par exemple un type de codage et un type de croisement), ou entre l'instance de problème et l'efficacité des opérateurs.

Mots-clés: Planification industrielle, problème de tournées de véhicules (VRP), classification, algorithmes évolutionnaires.

Abstract

Solving Vehicle Routing Problems (VRP) have been some of the most studied problems in combinatorial optimisation because they have many applications in the field of industrial planning. The related literature is quite large and diversified both in terms of variants of the problems and in terms of solving approaches. Identifying which class of problems a given real-world problem belongs to, in order to gather related works and determine the most relevant resolution method, is a difficult task.

The present thesis constitutes a feasibility study of a project to make these tasks easier, privileging evolutionary solving approaches. This project relies on three essential bases: a notation of the variants of VRP, a compilation of evolutionary operators from the literature, a set of rules linking VRP variants to evolutionary operators according to their efficiency. The objective is to find guidelines to design a solving algorithm according to the characteristics of the problem by identifying the subset of operators showing the greater estimated efficiency.

Putting the proposed notation into practice using several papers demonstrated that anyone using this notation can classify accurately papers from the literature and can recognise easily approaches and results that are similar to their own.

The experimental methodology proposed to infer rules linking the characteristics of a problem to the efficiency of evolutionary operators is illustrated by considering three types of crossover and three types of mutation. This study confirms that it is possible to determine which elements of an algorithm have a discernable impact on the performance. It reveals relationships between choices in the design of the algorithm (for instance the choice of a type of coding and a type of crossover) or between the variant of problem and the efficiency of the operators.

Keywords: Industrial planning, Vehicle Routing problem, Classification, Evolutionary algorithms.