MODÉLISATION ET RÉSOLUTION DE PROBLÈMES DE DISTRIBUTION ET D'ORDONNANCEMENT

THÈSE N° 1517 (1996)

PRÉSENTÉE AU DÉPARTEMENT DE MATHÉMATIQUES

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES TECHNIQUES

PAR

Yves-Nicolas ROCHAT

Ingénieur mathématicien diplômé EPF

originaire de l'Abbaye (VD)

acceptée sur proposition du jury:

Prof. A. Hertz, directeur de thèse
Dr A. Donzel, corapporteur
Prof. G. Finke, corapporteur
Prof. G. Laporte, corapporteur
Prof. F.-L. Perret, corapporteur
Prof. D. de Werra, corapporteur

Lausanne, EPFL
1996

À mes parents,
À Claire.

REMERCIEMENTS

Je remercie M. le Professeur Alain Hertz qui, par sa rigueur scientifique et sa disponibilité, m'a prodigué de précieux conseils et apporté son soutien constant tout au long de la rédaction de cet ouvrage. Mes remerciements vont également à M. le Professeur Dominique de Werra qui m'a initié à la Recherche Opérationnelle et plus particulièrement au domaine de la distributique.

Que MM. les membres du jury trouvent ici l'expression de ma très vive reconnaissance pour avoir accepté de juger mon travail.

Je remercie également la société SCITEC S.A. des nombreux conseils et encouragements qu'elle m'a donnés lors de l'étude des problèmes d'ordonnancement relatifs à l'optimisation des traitements analytiques d'échantillons dans un laboratoire robotisé.

Que MM. Jean-Claude Favre, Jean-Richard Rochat et Éric Taillard soient chaleureusement remerciés d'avoir relu très attentivement cette thèse.

Ma gratitude s'adresse également à mes collègues et notamment MM. Yves Mottet, Frédéric Semet et Éric Taillard avec lesquels j'ai eu le plaisir de collaborer scientifiquement.

Finalement, je tiens à exprimer ma reconnaissance à toutes celles et ceux qui m'ont apporté leur aide.

RÉSUMÉ

L'une des préoccupations majeures des entreprises dont le fonctionnement est tributaire du transport de leurs produits est d'améliorer leur système de distribution. Dans la première partie de cette thèse, nous montrons comment l'utilisation conjointe des mathématiques appliquées et de l'informatique permet de réduire significativement les coûts inhérents à la distribution de biens. Dans ce but, nous proposons des nouvelles méthodes de résolution du problème d'élaboration de tournées de véhicules. En plus d'être performantes, ces méthodes sont flexibles (elles tiennent compte d'une grande variété de contraintes) et robustes (elles peuvent s'appliquer à des problèmes très divers sans qu'il faille fixer avec précision leurs paramètres de contrôle). Nous avons ainsi pu résoudre un problème réel complexe en combinant une heuristique simple et rapide avec une méthode basée sur les techniques de recherche avec tabous. Un autre algorithme que nous proposons est une méthode évolutive qui se trouve à la croisée entre la méta-heuristique Tabou et les algorithmes génétiques. En plus des avantages précités et à l'inverse de la plupart des heuristiques développées à ce jour, cette nouvelle méthode possède l'atout de pouvoir être parallélisée aisément avec un nombre de processeurs ne dépendant pas de la taille du problème étudié.

Le second type de problèmes abordé dans cette thèse est celui relatif à l'optimisation de traitements d'échantillons dans un laboratoire robotisé d'analyses chimiques. Des études de marché récentes ont montré un intérêt croissant pour ces problèmes d'ordonnancement complexes qui, à notre connaissance, ne pouvaient pas être résolus à l'aide des méthodes développées à ce jour. Dans la seconde partie de cet ouvrage, nous proposons donc une heuristique constructive rapide ainsi qu'un algorithme génétique qui sont capables de résoudre efficacement ce nouveau type de problèmes. Le cas que nous avons étudié est une généralisation des problèmes d'ordonnancement liés à la gestion d'un environnement robotisé. Nos méthodes, qui ont été réellement implantées dans la pratique, sont souples d'emploi et peuvent gérer un grand nombre de contraintes. Plus précisément, nous tenons compte de la gestion d'une ressource non statique telle qu'un robot, de ressources capables de traiter simultanément plusieurs échantillons, des attentes (éventuellement bornées) entre les opérations d'un même échantillon ainsi que d'une possible souplesse quant aux temps de traitement des échantillons sur les ressources.

ABSTRACT

A major concern of many industrial enterprises is the improvement of their distribution system. In the first part of this thesis, we show how the use of tools from applied mathematics and computer science helps to reduce distribution costs significantly. For this purpose, we propose new methods for solving the vehicle routing problem. These methods are not only efficient but also flexible (they take into account a large variety of constraints) and robust (they can be applied to very different problems, without the need to determine precisely their control parameters). Combining a simple and fast heuristic procedure with a method based on Tabu search techniques makes it possible to solve complex real life problems. Another algorithm developed in this thesis is an evolutionary method at the crossroads of the Tabu meta-heuristic and genetic algorithms. In addition to the above advantages and as opposed to most of today's heuristics, an interesting feature of this new method is the fact that it can be easily codable on parallel CPU's. The number of processors can be chosen independently from the size of the problem.

The second type of problem addressed in this thesis is the optimization of sample processing in a robotized analytical system. Recent market studies have shown a growing interest in these complex scheduling problems, which, to our knowledge, cannot be solved with currently available methods. In the second part of this work, we describe both a fast constructive heuristic and a genetic algorithm to efficiently solve scheduling problems in a robotized environment. Our methods, which have already been used in practice, are flexible and can handle a large number of constraints. More precisely, we take into account the management of a non-static resource such as a robot, of resources able to simultaneously process several samples, of (possibly bounded) waiting times between operations on a single sample, and of flexibility in the samples processing times on the resources.

TABLE DES MATIÈRES

INTRODUCTION GÉNÉRALE

Cette thèse est consacrée à la modélisation et à la résolution de problèmes issus des domaines de la distributique et de l'ordonnancement. Chacune des deux parties qui composent cet ouvrage contient une introduction spécifique au type de problèmes qui y sera étudié.

Dans la première partie, nous développerons de nouveaux modèles flexibles pour l'élaboration de tournées de véhicules. Ils pourront tenir compte de multiples contraintes telles que la capacité volumique et la charge utile des camions, l'accessibilité de la clientèle, les heures de visite chez les clients, la durée totale des tournées ou les pauses des chauffeurs. Sur la base de ces modèles, l'objectif de cette première partie est de développer des algorithmes efficaces et robustes qui puissent résoudre des problèmes complexes de distribution, par exemple ceux provenant du monde industriel.

La seconde partie de cette thèse traitera des problèmes d'ordonnancement qui surviennent lorsqu'il faut gérer un laboratoire robotisé d'analyses chimiques. Nous présenterons des méthodes constructives et un algorithme évolutif pour résoudre des problèmes de ce type et notamment une application chimique réelle : la préparation d'échantillons pour l'identification des bactéries par leurs acides gras membranaires.

PARTIE 1

ÉLABORATION DE TOURNÉES DE VÉHICULES

Introduction à la distributique

Après avoir assisté à l'essor remarquable de toutes les techniques qui concourent à une automatisation efficace de la production – *la productique* –, nous observons que l'une des préoccupations majeures des entreprises industrielles actuelles est l'amélioration de leurs chaînes logistiques afin de pouvoir distribuer le plus efficacement possible leurs produits. Les sommes consacrées à cette distribution sont considérables. À titre d'exemple, Owoc et Sargious [Owo92] citent une étude canadienne indiquant que la proportion des coûts de transport par rapport au coût total de production est d'environ 12 %. De plus, ces coûts sont généralement supérieurs à ceux rattachés à la publicité et à la promotion des produits. Mentionnons encore que le montant des biens échangés par transport routier en 1989 entre le Canada et les États-Unis s'élève à 122 milliards de dollars [Cap91].

Ces constats témoignent de l'importance capitale que revêtent des problèmes tels que la conception et la gestion d'un système de distribution ou la réduction des coûts de livraison des produits finis. Si l'on a pu diminuer les coûts de production par une automatisation et une gestion intégrée, une utilisation conjointe et intelligente des mathématiques appliquées et de l'informatique devrait aussi permettre de réduire davantage les coûts de distribution. C'est là l'enjeu de *la distributique*.

La distributique est la science qui a pour but de concevoir et gérer efficacement des systèmes de distribution. Mais, si l'on décide d'englober dans le terme distribution tout ce qui a trait à la diffusion et à la circulation de biens de production, de personnes, de ressources ou d'informations, on remarque très vite qu'il faudrait introduire une classification des domaines de la distributique, tant le champ des activités qui y sont incluses est vaste. Alors, en général, on définit la distributique comme étant « *l'ensemble des techniques visant à effectuer des transports de biens ou de personnes d'une manière aussi performante que possible* ».

Dans cette première partie de la thèse, nous nous intéresserons plus particulièrement au cas le plus fréquent des problèmes de la distributique : le problème de l'élaboration de tournées de véhicules (PTV) pour lequel nous distinguons deux variantes.

La première, le *problème académique*, est la plus classique et la plus simple des deux variantes car elle ne prend en compte qu'un très petit nombre de contraintes. Cette version a été largement étudiée et de nombreux articles scientifiques ont été écrits sur ce sujet au cours des 30 dernières années. Malheureusement, la plupart des modèles décrits dans cette abondante littérature simplifient de manière excessive les problèmes issus de la réalité, ce qui rend leur utilisation dans la pratique peu courante.

La seconde variante, le *problème réel*, est plus complexe car elle prend en compte une grande variété de contraintes : capacité volumique et charge utile des camions, durée totale des tournées, flotte hétérogène de véhicules, accessibilité de la clientèle, heures de visite chez les clients, pauses des chauffeurs. L'élaboration de tournées de distribution et/ou de ramassage est un problème qui apparaît très fréquemment dans la vie de tous les jours. Le premier exemple qui nous vient à l'esprit est celui d'une chaîne de grands magasins qui distribue ses produits à un ensemble de succursales éparpillées sur un certain territoire. Mais après quelques instants de réflexion nous imaginons des systèmes de distribution des plus variés : distribution des quotidiens dans des cassettes à journaux, récolte de la monnaie dans les automates pour tickets de bus, livraison de travaux photographiques et ramassage de films à développer, transport scolaire, etc.

La multitude de contraintes à prendre en compte pour le problème réel sont de types très variés, par exemple économiques, techniques ou matériels. De nos jours, lorsque nous traitons des problèmes provenant du monde industriel, nous devons de plus en plus fréquemment faire face à de telles situations pour lesquelles nous ne disposons pas encore de méthodes suffisamment flexibles et efficaces.

L'originalité de la première partie de cette thèse tient donc essentiellement dans la prise en compte de telles contraintes dans le développement de nouveaux modèles et algorithmes pour le problème de l'élaboration de tournées de véhicules sous contraintes de *fenêtres de temps* (PTVFT). Cette terminologie désigne le cas où les livraisons chez les clients ne peuvent se dérouler que dans des plages horaires bien précises et fixées par les clients. Ce type de situations est très fréquent lorsque nous traitons des problèmes réels, par exemple lors de l'approvisionnement en produits frais des magasins ou lors de la distribution aux grandes surfaces de vente. Dans ces deux cas, les magasins doivent fixer les heures de livraisons de leurs commandes afin de pouvoir préparer leurs rayons de vente pour leur clientèle et traiter avec l'ensemble de leurs fournisseurs qui sont généralement fort nombreux.

1. Présentation générale du problème

1.1 L'optimisation combinatoire

En plus de leur indéniable intérêt pratique, les algorithmes d'optimisation de tournées de véhicules ont une importance théorique fondamentale. Le problème de base du PTV (un seul véhicule de capacité infinie, pas de contraintes sur les heures de livraison) consiste à trouver une tournée de longueur minimum qui visite tous les clients (en passant une et une seule fois par chacun de ceux-ci) pour finalement revenir à son point de départ, le dépôt. En dépit de sa formulation élémentaire, ce fameux problème, connu sous le nom du *Problème du Voyageur de Commerce* (PVC), est difficile à résoudre dès que le nombre de clients est important. Ce problème, décrit pour la première fois par Menger [Men32] en 1932 et dont Dantzig et al. [Dan54] ont entrepris la première étude systématique en 1954, est au coeur même d'une branche importante des mathématiques discrètes : *l'optimisation combinatoire*. Cette dernière se définit généralement comme étant l'étude mathématique qui conduit à un arrangement, une sélection ou un groupement adéquat d'un ensemble d'objets distincts donné [Law76].

Un problème P d'optimisation combinatoire, comme le PVC ou le PTV, exprimé en termes de minimisation se définit de la manière suivante :

$$\boxed{P: \min_{s \in S} f(s)}$$

S est un ensemble discret de *solutions admissibles* (c'est-à-dire de solutions qui vérifient l'ensemble des contraintes du problème P), appelé *l'espace des solutions admissibles* de P. Nous désignons par f une fonction quelconque, appelée la *fonction objectif* de P, qui associe à chaque solution $s \in S$ une valeur f(s) réelle ou entière. Le problème P consiste à déterminer une solution $s^* \in S$ qui soit de valeur $f(s^*)$ minimum (i.e. $\nexists\, s \in S \mid f(s) < f(s^*)$). Dans une telle situation, nous parlerons de solution *optimale* et de problème résolu de façon optimale. Cette formulation n'est pas restrictive car la recherche d'une solution qui maximise f(s) peut se modéliser en utilisant l'égalité $\max f(s) = -(\min -f(s))$.

La recherche de méthodes de résolution performantes pour le PVC et le PTV a été à l'origine d'importants développements en optimisation combinatoire et en programmation mathématique. Ainsi, l'étude, l'analyse, la mise au point et l'implantation d'algorithmes exacts ou approchés de plus en plus efficaces ont permis de résoudre des problèmes de plus en plus complexes.

1.2 Description du problème

Dans la version académique classique du PTV, un ensemble de clients avec des demandes connues doivent être desservis par une flotte homogène de véhicules de capacité donnée à partir d'un unique centre de distribution ou dépôt. L'objectif est de déterminer une séquence de livraison pour chaque véhicule de sorte que : la capacité maximale du véhicule ne soit pas dépassée, chaque client soit desservi par un seul véhicule et la distance totale parcourue par la flotte soit la plus petite possible. Une légère variante de ce problème, rencontrée souvent dans la littérature, consiste à imposer une limite supérieure à la durée d'une tournée. Nous parlerons alors du PTV avec contrainte de durée (PTVD) alors que pour la première description nous ferons référence au PTV avec contrainte de capacité (PTVC). Ces deux types de contraintes sont souvent considérées comme les *contraintes classiques* du PTV. Nous parlerons du PTVCD lorsque ces deux contraintes sont prises en compte simultanément.

Le PTV, même dans sa version classique, est un problème d'optimisation combinatoire fascinant car il est simple à décrire mais extrêmement difficile à résoudre de façon optimale. Une autre caractéristique remarquable de ce problème est que sa version de base peut être étendue, par l'adjonction de nouvelles contraintes, de manière quasi infinie. Ces nouvelles variations du problème de base ne sont pas simplement des formulations mathématiques sans intérêt mais ont souvent une signification pratique réelle.

L'une des plus importantes d'entre elles considère le cas où les livraisons chez l'ensemble (ou une partie) des clients ne sont autorisées que dans des plages horaires – *fenêtres de temps* – bien précises, par exemple les heures d'ouverture d'un magasin. Dans cette situation, on parlera du PTV avec contraintes de fenêtres de temps (PTVFT). Il s'agit de la restriction pour le PTV qui est la plus fréquemment imposée mais il en existe bien d'autres. En effet, elle fait partie d'un ensemble de restrictions, *les contraintes spéciales*, parmi lesquelles on peut noter :

- collecte de marchandise chez certains clients,
- non-accessibilité d'une partie de la flotte chez certains clients,

- relations de précédence entre les lieux de livraisons,
- priorité de certaines commandes,
- flotte hétérogène de véhicules,
- durée de travail des chauffeurs,
- etc.

Un inventaire détaillé de ces contraintes spéciales peut être consulté dans l'article de Ronen [Ron88]. Dans la suite de ce travail, nous allons nous concentrer sur le PTVFT que ce soit dans sa version académique (sans adjonction de contraintes) ou réelle (prise en compte de certaines contraintes spéciales).

En présence de fenêtres de temps, les coûts totaux d'élaboration d'un plan de tournées de véhicules n'incluent pas seulement la distance totale parcourue et les coûts liés à l'utilisation des véhicules, mais aussi les frais engendrés par les temps d'attente qui surviennent lorsqu'un camion arrive trop tôt chez un client. Dans le PTVFT, il faut tenir compte des aspects aussi bien spatiaux que temporels des mouvements des véhicules.

1.3 Notations

Nous considérons un graphe orienté $G = (X, U)$ où $X = \{x_0, \ldots, x_i, \ldots, x_n\}$ est un ensemble de sommets qui représente les clients qui doivent être desservis, x_0 étant le dépôt. $U = \{(x_i, x_j) \mid i \neq j, 0 \leq i, j \leq n\}$ est un ensemble d'arcs qui dénote les trajets entre chaque paire de clients. À chaque trajet (x_i, x_j) on associe deux valeurs non négatives c_{ij} et t_{ij} qui représentent respectivement la distance du trajet le plus court du client x_i au client x_j (coût du trajet) et sa durée. Chaque client x_i, caractérisé par sa localisation géographique, a passé une commande d'un poids total q_i ($q_0 = 0$) qui doit lui être livrée au cours d'une plage horaire (fenêtre de temps) fixée à l'avance. Celle-ci se définit généralement à l'aide d'une borne minimale d_i et d'une borne maximale f_i qui désignent respectivement les limites de temps inférieure et supérieure entre lesquelles le début du service chez le client x_i doit avoir lieu. Lorsqu'il n'y a pas de restriction sur les heures d'ouverture et de fermeture du dépôt, nous considérerons que $d_0 = 0$ et $f_0 = \infty$.

Une flotte de véhicules identiques de capacité Q est basée au dépôt. Le nombre m de camions nécessaires pour fournir une solution admissible au PTVFT n'est généralement pas connu a priori. C'est une variable du problème dont la valeur est comprise entre une borne inférieure m_{inf} et une borne supérieure m^{sup}. Lorsque $m_{inf} = 1$ et $m^{sup} = n$ la flotte n'est pas limitée et quand $m_{inf} = m^{sup}$, la flotte est dite *fixe*. Dans la plupart des cas réels, la flotte n'est pas fixe car il est toujours possible de faire appel à un

transporteur privé lorsque les camions basés au dépôt ne suffisent pas pour desservir la totalité des commandes. Il est alors d'usage courant d'associer un coût fixe d'utilisation à chaque véhicule et de chercher à minimiser la variable m. Notons de plus, qu'une borne inférieure pour m_{inf} est donnée trivialement par la relation suivante :

$$m_{inf} \geq \left\lceil \frac{\sum_{i=0}^{n} q_i}{Q} \right\rceil$$

Chaque véhicule v_i, i = 1, ..., m, effectue une tournée T_i que nous représenterons à l'aide d'un vecteur de clients comme suit :

$$\boxed{T_i = (x_{i_0}, x_{i_1}, \ldots, x_{i_j}, x_{i_{j+1}}, \ldots, x_{i_{n_i+1}})}$$

où x_{i_0} et $x_{i_{n_i+1}}$ désignent le dépôt et n_i le nombre de clients, différents du dépôt, desservis par le véhicule v_i.

Pour chaque client x_i sont définis un temps d'accès a_i ($a_0 = 0$), un temps d'attente éventuel w_i ainsi qu'un temps de service s_i ($s_0 = 0$). De plus, l'heure d'arrivée chez le client x_i est désignée par H_i et l'heure de début du service chez ce même client par h_i. Si un véhicule se déplace directement du client x_i vers le client x_j, nous avons donc : $H_j = h_i + s_i + t_{ij} + a_j$. Trois situations différentes peuvent alors se présenter. Tout d'abord lorsque le véhicule arrive à temps chez le client x_j :

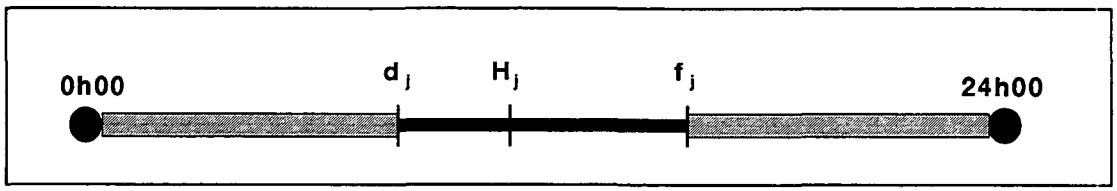


Figure 1.1 : L'heure d'arrivée chez le client $x_j \in [d_j, f_j]$.

Si le camion arrive trop tôt chez le client x_j, le chauffeur devra attendre un temps égal à w_j avant de pouvoir décharger la commande q_j :

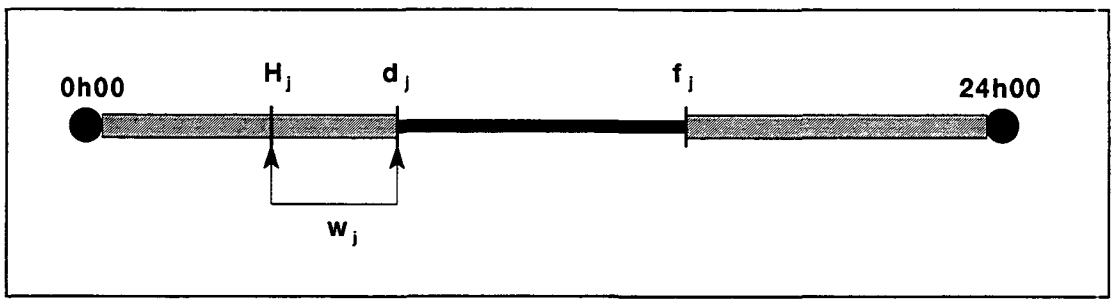


Figure 1.2 : Attente chez le client x_j car $H_j < d_j$.

Dans ces deux premières situations, nous avons $h_j = \max(d_j, H_j)$ et $w_j = \max(0, d_j - H_j)$. La dernière situation se présente lorsque le camion arrive trop tard chez le client x_j et qu'une violation de la contrainte de fenêtres de temps se produit :

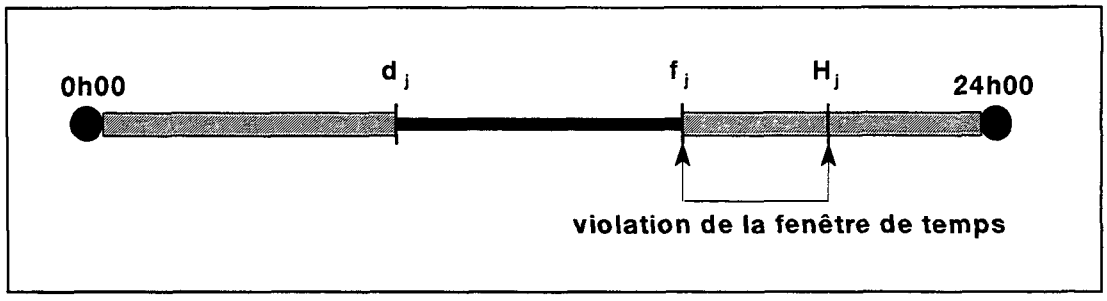


Figure 1.3 : Violation de la contrainte de fenêtres de temps.

La valeur de la violation est égale à $\max(0, H_j - f_j)$. De telles violations sont interdites dans la solution que nous fournirons au responsable des transports.

Finalement, notons que certains arcs (x_i, x_j) appartenant à U peuvent être éliminés en tenant compte des contraintes temporelles $(d_i + s_i + t_{ij} + a_j > f_j)$, des contraintes de capacité $(q_i + q_j > Q)$ ou d'autres considérations.

Le PTVFT consiste alors à déterminer un ensemble de tournées minimisant la distance totale parcourue par les véhicules qui les effectuent de sorte que :

1. chaque tournée commence et se termine au dépôt,
2. chaque client de $X \setminus \{x_0\}$ est desservi une seule fois et par un seul véhicule,
3. le poids total des commandes transportées par chaque véhicule ne dépasse pas Q,
4. la durée de chaque tournée n'excède pas une borne supérieure D_{max},
5. l'heure du début de la livraison chez un client x_i appartient à l'intervalle de temps $[d_i, f_i]$.

1.4 Formulation mathématique

Le PVTFT peut être vu comme un problème constitué de deux sous-problèmes qui sont d'une part, la détermination d'une affectation optimale des clients aux véhicules et d'autre part, la recherche de la tournée optimale pour chaque véhicule. Bien qu'il existe trois formulations de base pour le PVTFT (problème de partitionnement, problème de flot et problème d'affectation généralisé), nous allons présenter ci-dessous seulement la dernière d'entre elles car elle met clairement en évidence les deux sous-problèmes du PVTFT et de plus elle a permis le développement du célèbre algorithme exact de

Fisher & Jaikumar [Fis81]. Cette formulation, qui ne tient pas compte de la contrainte de durée maximale d'une tournée, a l'avantage de ne pas requérir une flotte de véhicules identiques. Elle utilise trois types de variables :

$$x_{ijk} = \begin{cases} 1 \text{ ssi l'arc } (x_i, x_j) \text{ est utilisé par le véhicule k} \\ 0 \text{ sinon} \end{cases}$$

$$z_{ik} = \begin{cases} 1 \text{ ssi le client } x_i \text{ est desservi par le véhicule k} \\ 0 \text{ sinon} \end{cases}$$

h_i = heure de début de service chez le client x_i

Le problème s'exprime alors de la façon suivante :

$$(1) \quad \text{Min} \sum_{k=1}^{m} \sum_{i=0}^{n} \sum_{j=0}^{n} c_{ij} \cdot x_{ijk}$$

sous contraintes

$$(2) \quad \sum_{k=1}^{m} z_{ik} = \begin{cases} m, & i = 0 \\ 1, & i = 1, \ldots, n \end{cases}$$

$$(3) \quad \sum_{i=0}^{n} q_i \cdot z_{ik} \leq Q \quad (k = 1, \ldots, m)$$

$$(4) \quad z_{ik} \in \{0,1\} \quad (i = 0, \ldots, n ; k = 1, \ldots, m)$$

$$(5) \quad \sum_{i=0}^{n} x_{ijk} = z_{jk} \quad (j = 0, \ldots, n ; k = 1, \ldots, m)$$

$$(6) \quad \sum_{j=0}^{n} x_{ijk} = z_{ik} \quad (i = 0, \ldots, n ; k = 1, \ldots, m)$$

$$(7) \quad x_{ijk} = 1 \Rightarrow h_i + s_i + t_{ij} + a_j \leq h_j \quad (i, j = 0, \ldots, n ; k = 1, \ldots, m)$$

$$(8) \quad d_i \leq h_i \leq f_i \quad (i = 0, \ldots, n)$$

$$(9) \quad x_{ijk} \in \{0,1\} \quad (i, j = 0, \ldots, n ; k = 1, \ldots, m)$$

Tableau 1-1 : Formulation mathématique du PTVFT.

La fonction objectif (1) consiste à minimiser la distance totale parcourue par les véhicules. Les contraintes (4) et (9) sont les contraintes d'intégralité du modèle.

Lorsqu'elles sont respectées, les contraintes (2), (5) et (6) indiquent que chaque client doit être visité par un seul véhicule et une et une seule fois. La contrainte (3) signale que la capacité d'un véhicule ne peut pas être dépassée. Les contraintes (7) et (8) sont les contraintes de fenêtres de temps. Les sous-tours sont éliminés par la contrainte (7) qui peut être remplacée par la contrainte linéaire suivante [Sol83, Des83] :

$$(10)\quad h_i + s_i + t_{ij} + a_j - h_j \le T_{ij} \cdot (1 - \sum_k x_{ijk}) \quad (i, j = 0, \ldots, n\,;\, k = 1, \ldots, m)$$

$$\text{avec} \quad T_{ij} = f_i + s_i + t_{ij} + a_j - d_j$$

C'est une généralisation de la contrainte classique d'élimination de sous-tours pour le PVC proposée par Miller et al. [Mil60]. Si l'arc (x_i, x_j) est utilisé dans la solution alors x_{ijk} vaudra 1 pour un certain k et nous aurons bien $h_i + s_i + t_{ij} + a_j \le h_j$. Dans le cas contraire, x_{ijk} vaudra 0 pour tous les k et la contrainte (10) devient :

$$h_i + s_i + t_{ij} + a_j - h_j \le T_{ij} = f_i + s_i + t_{ij} + a_j - d_j$$

$$\Leftrightarrow \quad h_i - h_j \le f_i - d_j \quad \Leftrightarrow \quad d_j - h_j \le f_i - h_i$$

Comme $d_i \le h_i \le f_i \quad \forall\, i = 0, \ldots, n$, la dernière inégalité est toujours vérifiée.

Cette formulation mathématique fait clairement apparaître deux problèmes bien connus en optimisation combinatoire. Il s'agit d'une part, du problème d'affectation généralisé (contraintes (2), (3) et (4)) et d'autre part, du problème du voyageur de commerce avec fenêtres de temps (PVCFT). En effet, pour des z_{ik} fixés qui vérifient (2) et (4), les contraintes (5), (6), (8), (9) et (10) reviennent à résoudre m PVCFT indépendants. Pour un véhicule k donné, les variables x_{ijk} définissent un circuit hamiltonien, c'est-à-dire un circuit passant par chaque sommet une et une seule fois, qui respecte les contraintes horaires.

2. État actuel de la recherche

Le PTV joue un rôle central en distributique et constitue depuis longtemps un domaine de recherche privilégié en recherche opérationnelle. Ce problème a été étudié intensivement durant les 30 dernières années et de multiples algorithmes de résolution exacts et heuristiques ont vu le jour. Des informations complètes et précises sur ces méthodes sont présentées dans les articles de Christofides et al. [Chr79], Bodin et al. [Bod83], Christofides [Chr85a], Laporte et Nobert [Lap87] et Golden et Assad [Gol88]. Pour un survol d'ensemble plus récent des algorithmes de résolution du PTV, il faut se référer à Laporte [Lap92], Osman [Osm93b] et Fisher [Fis95]. Signalons, de plus, la parution d'une publication récente [Lap95] qui regroupe une bibliographie de 500 références sur les problèmes du voyageur de commerce, des tournées de véhicules, du postier chinois et du postier rural.

En ce qui concerne le PTVFT, il faut noter que c'est une importante généralisation du PTV qui n'a attiré que récemment l'attention des chercheurs en recherche opérationnelle. En effet, alors qu'au début des années 1980 il n'existait quasiment aucun algorithme permettant la résolution de tels problèmes, nous sommes aujourd'hui en possession de plusieurs méthodes exactes ou approchées efficaces pour le PVTFT académique. Le lecteur intéressé est invité à consulter les articles de Solomon et Desrosiers [Sol88b], Desrochers et al. [Des88] et Desrosiers et al. [Des94] pour connaître l'état de l'art dans ce domaine. Il reste cependant bien des défis à relever pour ce problème. L'un des plus importants étant de concevoir un algorithme flexible et performant pour le PTVFT lorsqu'il est issu du monde industriel, avec la multitude de contraintes additionnelles que cela implique.

2.1 Difficulté du problème

Malgré les études approfondies précédentes, le PTV est encore loin d'être résolu de façon optimale. En effet, une étude de Lenstra et Rinnooy Kan [Len81] montre que ce problème appartient à la classe des problèmes NP-difficiles c'est-à-dire qu'il n'existe pas à ce jour un algorithme exact pouvant le résoudre en un temps qui soit une fonction

polynomiale de sa taille n. Ce constat est d'autant plus vrai lorsqu'on ajoute la contrainte de fenêtres de temps au PTV classique. Les travaux de Savelsbergh [Sav85] et Solomon [Sol86] montrent en effet que le PTVFT est fondamentalement plus difficile que le PTV. C'est-à-dire que non seulement le PTVFT est NP-difficile mais de plus, la recherche d'une solution admissible est un problème NP-complet [Gar79] lorsque la taille de la flotte de véhicules est fixe.

2.2 Les algorithmes exacts

Selon une étude de Laporte et Nobert [Lap87], les méthodes exactes pour le PTV peuvent êtres classifiées en trois grandes catégories.

2.2.1 Les algorithmes d'énumération implicite

Ces méthodes constructives consistent à créer graduellement une solution à l'aide d'un arbre d'énumération implicite (branch and bound). Elles sont, en général, assez flexibles et peuvent prendre en compte un nombre important de contraintes. De par leur nature, plus le nombre de contraintes est grand, meilleure est leur efficacité car le développement de l'arbre d'énumération implicite se restreint. Mentionnons les travaux de Christofides et al. [Chr81a, Chr81b] et ceux de Laporte et al. [Lap86].

2.2.2 La programmation dynamique

Les premiers à proposer l'utilisation de la programmation dynamique pour le PTV furent Eilon et al. [Eil71] en 1971. Pendant longtemps, cette approche ne fut pas directement utilisée de par le nombre considérable de calculs qu'elle engendrait. Des résultats n'avaient été obtenus que pour des problèmes de très petite taille (n = 10 à 25), jusqu'à ce que Christofides [Chr85b] présente des résultats encourageants pour le PTV classique. En utilisant des méthodes de relaxation d'états [Chr81a], il a pu résoudre sans difficulté tout problème pour des tailles allant jusqu'à 50.

2.2.3 La programmation linéaire en nombres entiers

Balinski et Quandt [Bal64] furent parmi les premiers à proposer une formulation de partitionnement d'ensemble pour le PTV. Ce modèle requiert généralement un nombre astronomique de variables si bien que pour le résoudre on utilise des techniques de génération de colonnes qui ne considèrent qu'un nombre restreint de variables

simultanément. Cette approche a été utilisée avec succès par Desrosiers et al. [Des84], Agarwal et al. [Aga89] et Desrochers et al. [Des92].

Toujours dans cette catégorie d'algorithmes, un célèbre modèle de flux de véhicules a été développé pour le PTVFT par Fisher [Fis78] et Fisher et Jaikumar [Fis81]. Une formulation plus compacte pour le PTV avec contraintes de capacité et de durée est décrite par Laporte et al. [Lap85], lesquels ont résolu des problèmes jusqu'à des tailles de 60 en utilisant un algorithme de relaxation de contraintes.

2.2.4 Utilité des algorithmes exacts

La dernière méthode présentée s'avère plus efficace pour des problèmes faiblement contraints alors que pour des problèmes fortement contraints, les autres algorithmes exacts sont plus appropriés.

Mêmes si les algorithmes exacts favorisent une meilleure compréhension de la structure du PTV, ils ne peuvent en général pas résoudre des problèmes de grande taille ($n > 50$) si bien que leur utilité dans la pratique reste limitée.

Dans l'état actuel des recherches, les méthodes de résolution exactes se limitent à la considération des contraintes classiques ou d'une contrainte de type classique et une de type spécial. Mentionnons toutefois que les méthodes exactes peuvent être arrêtées dans leur recherche de l'optimum par tout un ensemble de critères, allant du plus simple au plus perfectionné. Nous avons alors un algorithme d'*optimisation incomplète* dont les performances en termes de temps de calcul sont améliorées sans toutefois offrir la garantie que la solution obtenue soit optimale. N'importe quel algorithme d'énumération implicite connu peut être utilisé. Christofides et al. [Chr79] décrivent une telle procédure d'exploration directe d'un arbre, basée sur un algorithme exact dû à Christofides [Chr76].

2.3 Les heuristiques

Pour quantité de problèmes réels d'élaboration de tournées de véhicules, et plus généralement pour des problèmes complexes d'optimisation (soit par leur structure soit par leur taille), la recherche de la solution optimale est exclue. Les principales raisons sont le nombre astronomique de solutions à passer en revue et le temps de calcul

prohibitif que nécessiterait une méthode exacte de résolution. Afin de pouvoir malgré tout traiter de tels problèmes, des méthodes approximatives de résolution, appelées *heuristiques*, doivent être employées. Celles-ci fournissent une ou plusieurs solutions du problème considéré, si possible proche de l'optimum, en un temps de calcul limité. Malheureusement, elles n'offrent pas la garantie de trouver la solution optimale du problème étudié même si, pour certaines d'entre elles, des garanties d'écart par rapport à l'optimum peuvent êtres obtenues. Par conséquent, la garantie d'optimalité présente dans les algorithmes exacts est échangée contre l'espoir d'obtenir rapidement une solution de bonne qualité. Ce type de méthodes est décrit en détail dans l'article de Silver et al. [Sil80].

Une des principales raisons pour lesquelles les chercheurs se sont intéressés aux heuristiques pour le PTV est leur aptitude à fournir des solutions qui tiennent compte simultanément des contraintes classiques et de plusieurs contraintes spéciales. De telles méthodes sont de plus, par nature, aisément implantables sur des micro-ordinateurs et permettent, si elles ont été judicieusement élaborées, d'obtenir une solution de bonne qualité en un temps de calcul raisonnable. Par exemple, les heuristiques pour le PTV qui utilisent des règles simples de construction d'une solution nécessitent un temps de calcul qui est en général polynomial par rapport à n et le degré du polynôme ne dépasse pas 3.

Les heuristiques développées au fil des ans pour le PTV, pour lequel il existe une multitude de variantes, sont nombreuses et des plus diverses. Une classification de celles-ci est proposée par Christofides et al. dans [Chr79]. Dans les paragraphes suivants, nous décrirons trois familles distinctes d'heuristiques dans lesquelles peuvent se répartir la plupart des méthodes approchées pour le PTV. Si aucune précision n'est apportée quant au nombre m de véhicules disponibles au dépôt, nous considérerons dans ce qui suit que cette valeur est une variable des problèmes étudiés qui n'est pas bornée supérieurement.

2.3.1 Les heuristiques simples de première génération

2.3.1.1 Les heuristiques constructives

À un pas donné du processus d'élaboration des tournées de véhicules à l'aide d'une heuristique constructive, une tournée incomplète existe. Celle-ci se développera lors des étapes suivantes de l'algorithme, pour atteindre sa configuration finale si une solution admissible du problème existe. La construction d'une telle tournée se déroule en considérant les commandes qui ne sont pas encore desservies les unes après les autres.

Ces méthodes peuvent être soit séquentielles soit parallèles. Dans le premier cas, les différentes tournées sont construites successivement alors que dans le second, plusieurs d'entre elles peuvent se développer simultanément.

Une des méthodes les plus connues qui appartient à cette catégorie a été proposée par Clarke et Wright [Cla64]. Son principal avantage est de pouvoir tenir compte d'un grand nombre de contraintes alors que ses deux défauts majeurs sont d'une part, une tendance à produire des tournées périphériques et d'autre part, un temps de calcul important. Afin de remédier à ces inconvénients, différentes variantes ont été proposées par Gaskell [Gas67], Yellow [Yel70] et Paessens [Pae88]. Mentionnons encore la méthode de Mole et Jameson [Mol76] qui est une généralisation de la procédure de Clarke et Wright car elle envisage l'extension de la tournée en construction par l'insertion d'un nouveau client à n'importe quelle position de la tournée et non plus uniquement à l'une de ses extrémités. Pour conclure le survol des heuristiques constructives, citons les travaux de Dror et Levy [Dro86] et ceux de Desrochers et Verhoog [Des91] qui proposent une heuristique basée sur la résolution d'un problème de couplage de poids maximum, capable de gérer une flotte hétérogène de véhicules.

2.3.1.2 Les heuristiques à deux phases

Pour la grande majorité des méthodes à deux phases, la première étape consiste à affecter des groupes de clients à des véhicules et la seconde à déterminer pour chaque véhicule, à l'aide des algorithmes connus pour le PVC, la meilleure tournée passant par l'ensemble des clients qui lui ont été assignés. L'algorithme de *balayage*, dont l'origine remonte aux travaux de Wren [Wre71] et Wren et Holliday [Wre72], appartient à cette classe. Sa version la plus couramment utilisée est due à Gillett et Miller [Gil74]. Cette méthode comporte toutefois un défaut important caractérisé par le fait que la condition de base à son utilisation est de considérer la position des clients à desservir à l'aide de leurs coordonnées polaires. En effet, deux clients proches en coordonnées polaires pourraient très bien être éloignés en distance réelle. Cette situation s'envisage facilement dans un pays montagneux comme la Suisse où deux clients seraient situés dans deux vallées voisines.

Une méthode du même style, mais plus perfectionnée, est celle de Christofides et al. [Chr79] qui a été par la suite affinée par Toth [Tot84].

Des heuristiques, où l'ordre des deux phases présentées ci-dessus est inversé, ont été proposées par Beasley [Bea83] et Haimovich et Rinnooy Kan [Hai85]. Typiquement, ces méthodes résolvent d'abord un PVC et ensuite décomposent la solution obtenue en tournées admissibles. Mais, outre le fait que cette façon de procéder est moins naturelle que la précédente, les résultats obtenus sont généralement moins bons.

Nous ne pouvons pas terminer le survol de cette classe d'heuristiques à deux phases sans mentionner la méthode d'affectation généralisée de Fisher et Jaikumar [Fis81] qui a connu un grand succès et a été utilisée dans de nombreux cas pratiques. Cet algorithme, qui se décompose naturellement en deux sous-problèmes classiques à savoir un problème d'affectation généralisé et plusieurs PVC, possède l'avantage de toujours fournir une solution admissible. De plus, il converge en théorie vers la solution optimale bien que les temps de calcul requis puissent être très longs. C'est la raison pour laquelle on considérera généralement une solution approchée obtenue après un certain nombre d'itérations.

2.3.2 Les heuristiques d'amélioration

Ce sont des méthodes itératives séquentielles qui examinent l'espace des solutions d'un problème en se déplaçant, à chaque itération, d'une solution à une autre. Cet examen se poursuit jusqu'à ce qu'un critère d'arrêt soit satisfait. Une telle méthode est utilisée afin d'améliorer une solution connue (la solution initiale), obtenue soit aléatoirement soit par n'importe lequel des algorithmes décrits précédemment. C'est la raison du nom donné à ce type de méthodes.

L'utilisation des heuristiques itératives suppose que l'on puisse définir, pour toute solution $s \in S$, un ensemble N(s) de *solutions voisines* de s, c'est-à-dire proches de s dans un certain sens. Cet ensemble N(s), appelé *voisinage* de s, est bien évidemment contenu dans S. Dans le contexte du PTV, cet ensemble est généralement défini comme étant l'ensemble des plans de transport qu'il est possible d'obtenir en échangeant n_1 client(s) d'une certaine tournée de la solution actuelle s avec n_2 client(s) d'une autre tournée de s. Pratiquement, les valeurs de n_1 et n_2 vérifient les conditions suivantes : $0 \leq n_1 \leq 1$ et $0 \leq n_2 \leq 1$. Par conséquent, les échanges possibles consistent à permuter deux clients se trouvant sur des tournées différentes ou à déplacer un client d'une tournée à une autre. Ainsi dans ce dernier cas, deux solutions du PTV sont voisines si et seulement si elles ne diffèrent que dans l'affectation et la position d'un unique client.

Nous citerons deux grandes classes de ces heuristiques dans les sections suivantes.

2.3.2.1 Les heuristiques simples d'amélioration

Cette première classe d'heuristiques regroupe les méthodes connues sous les noms de méthodes de descente et méthodes d'échanges de type r–opt.

2.3.2.1.1 Méthodes de descente

Ces méthodes comptent parmi les heuristiques d'amélioration les plus simples pour traiter des problèmes d'optimisation combinatoire. La formulation générique d'une méthode de descente est la suivante :

Choisir une solution initiale $s_0 \in S$;

continuer := vrai ; k := 0 ;

Tant que continuer **faire**

Choisir une solution $s_{k+1} \in N(s_k)$ telle que $f(s_{k+1}) \leq f(s)\ \forall\ s \in N(s_k)$;

Si $f(s_{k+1}) < f(s_k)$

alors k := k + 1

sinon continuer := faux ;

Tableau 2-1 : La méthode de descente.

Bien que ce type d'approches fournisse des solutions de qualité acceptable par rapport à son temps d'exécution généralement court, deux défauts majeurs en limitent son efficacité :

- La recherche du meilleur voisin dans le voisinage N(s) peut, suivant la taille et la structure de ce dernier, être un problème très coûteux en terme de temps de calcul et parfois aussi difficile que le problème original.
- Ce type de méthodes s'arrête dès qu'un optimum local du problème considéré est atteint. De plus, ce dernier n'est pas forcément de bonne qualité.

2.3.2.1.2 Méthodes d'échanges de type r–opt

Ces méthodes d'amélioration ont été proposées par Lin [Lin65]. Cette classe d'algorithmes fut initialement développée pour le PVC, mais elle s'applique également à tout problème combinatoire dont la solution consiste en une permutation de n objets.

Pour le PTV, cette méthode s'applique à tour de rôle sur chacune des tournées constituant une solution du problème étudié. Son application à une de ces tournées peut se décrire de la façon suivante :

(0) Choisir une tournée T d'un plan de transport admissible comme solution initiale.

(1) Enlever r arcs de T et reconnecter les r chaînes restantes de toutes les façons possibles. Dès qu'une reconnection quelconque produit une tournée T' de coût moindre, considérer celle-ci comme nouvelle solution initiale et répéter l'étape (1). Si la tournée T ne peut pas être améliorée, l'algorithme s'arrête.

Tableau 2-2 : Méthode d'échanges de type r–opt pour le PTV.

Comme le nombre de solutions à considérer est de l'ordre de n^r puisqu'il y a $\binom{n}{r}$ façons d'enlever r arcs et r! façons de reconnecter les chaînes restantes, on se limite en général à r = 2 ou 3. Cet algorithme a été amélioré par Lin et Kernighan [Lin73] et par Or [Or76] qui proposent une procédure réduite d'ordre n^2 donnant en pratique d'aussi bons résultats qu'un algorithme 3–opt (Golden et Stewart [Gol85]). Des travaux plus récents concernant ce type de méthodes ont été réalisés par Savelsbergh [Sav85] et Thompson [Tho88]. Mentionnons aussi une nouvelle technique prometteuse, appelée *GENIUS*, proposée par Gendreau et al. [Gen92].

Lorsque les méthodes présentées dans cette section sont appliquées au PTVFT, il faut cependant s'assurer en tout temps que les solutions retenues demeurent admissibles par rapport à la contrainte des fenêtres de temps.

2.3.2.2 Les heuristiques intelligentes d'amélioration

Le principal défaut des heuristiques simples d'amélioration est qu'elles sont incapables de progresser au-delà du premier optimum local rencontré, ce qui limite considérablement leur puissance. Pour remédier à cette situation, des nouvelles techniques, présentées dans ce qui suit, ont été élaborées au cours des quinze dernières années.

2.3.2.2.1 Recuit simulé

Cette méthode d'améliorations successives tire son nom d'une analogie avec un principe de thermodynamique appliqué au processus de recuit d'un matériau. Son origine remonte aux travaux de Metropolis et al. [Met53]. Pour amener un matériau à un état solide d'énergie aussi faible que possible, on commence par le chauffer à haute température de sorte que les particules soient distribuées aléatoirement dans l'état liquide. Puis, afin d'éviter des minima locaux d'énergie, on diminue la température T par petits paliers successifs. Le refroidissement doit se dérouler aussi lentement que possible afin que le système atteigne son état d'équilibre à l'intérieur d'un palier de température donné. A haute température, on peut atteindre tous les états mais, à mesure que le système refroidit, le nombre de possibilités diminue et le processus finit par converger vers un état stable où l'on suppose qu'il a atteint un niveau d'énergie minimum. A la température T, un nouvel état du système est obtenu en considérant un déplacement infinitésimal aléatoire d'une particule quelconque. Ce mouvement provoque une modification de l'énergie E du système de ΔE. Le déplacement d'une particule est accepté si $\Delta E < 0$ (l'énergie du système diminue) sinon on ne l'accepte qu'avec une probabilité non nulle $Pr(\Delta E,T)$ définie comme suit :

$$Pr(\Delta E,T) = e^{\frac{-\Delta E}{k_b \cdot T}}, \text{ où } k_b \text{ est la constante physique de Boltzman}$$

Metropolis et al. ont montré qu'en réitérant un tel choix d'acceptation d'un nouvel état, le système converge vers un état d'équilibre thermique.

Kirkpatrick et al. [Kir83] et indépendamment Cerny [Cer85] ont été les premiers à se rendre compte, 30 ans plus tard, que ce procédé pouvait être adapté pour la résolution de problèmes d'optimisation combinatoire en remplaçant simplement la fonction énergie par la fonction objectif. La méthode du recuit simulé était née. L'idée à la base de cette méta-heuristique, qui la différencie des méthodes de descente, est de pouvoir se diriger de la solution courante s vers une solution voisine s' de moins bonne qualité avec une probabilité non nulle. Le but de cette idée est de tenter d'échapper aux optima locaux de la fonction objectif. La valeur de cette probabilité dépend de la détérioration $\Delta f = f(s') - f(s)$ de la fonction objectif et d'un paramètre T, apparenté à la température, qui tend vers 0 à mesure que le processus évolue. Les modifications graduelles de la température suivent une règle de refroidissement par paliers bien précise. Lorsqu'un certain nombre *nbiter_palier* d'itérations ont été effectuées, la température du prochain palier est déterminée à l'aide de la règle suivante qui est souvent utilisée en pratique :

$$T_{p+1} = \alpha \cdot T_p, \quad p \in N, 0 < \alpha < 1$$

T_0 est la température initiale du système et α, très proche de 1, le coefficient de refroidissement du système. Le voisinage de s se réduit à un seul élément, généré aléatoirement de façon uniforme dans N(s). La meilleure solution trouvée au cours de l'heuristique est stockée dans la variable s*. L'algorithme se décrit comme suit :

```
Choisir une solution initiale s ∈ S ; s* := s ;
p := 0 ;                          { compteur du nombre de paliers }
nouveau_palier := vrai ;          { variable booléenne qui indique s'il faut effectuer un
                                    nouveau cycle d'itérations }
Tp := Tini ;                      { Tini est la température initiale du système }
Tant que nouveau_palier faire
    nbiter := 0 ;                 { compteur d'itérations pour un palier }
    nouveau_palier := faux ;
    Tant que (nbiter < nbiter_palier) faire
        nbiter   := nbiter + 1 ;
        Générer aléatoirement de façon uniforme une solution s' ∈ N(s) ;
        Δf := f(s') – f(s) ;
        Si Δf < 0
            alors   s := s' ; nouveau_palier := vrai ;
            sinon   Pr(Δf, Tp) := exp(–Δf/Tp) ;
                    Générer z uniformément dans l'intervalle ]0,1[ ;
                    Si z < Pr(Δf,Tp)
                        alors   s := s' ; nouveau_palier := vrai ;
        Si f(s) < f(s*)
            alors s* := s ;
    p   := p + 1 ;
    Tp  := α·Tp-1 ;
```

Tableau 2-3 : Le recuit simulé.

Il est important de noter que la performance du recuit simulé dépend étroitement de la règle de refroidissement utilisée. Un des avantages de cette méthode est de posséder des résultats théoriques de convergence. Hajek [Haj85], puis Faigle et Schraeder [Fai88] dans une généralisation de ce résultat, ont établi un résultat de convergence en probabilité vers une solution optimale du problème considéré si la suite T_p des températures remplit certaines conditions. Comme celles-ci sont généralement

irréalisables en pratique et très coûteuses en terme de temps de calcul, nous utilisons de préférence la règle de refroidissement décrite précédemment avec une valeur de α égale à 0.93. Une telle méthode a été appliquée au PTV par Osman [Osm91, Osm93a].

2.3.2.2.2 La méthode de recherche avec tabous

La méthode de recherche avec tabous (RT), dont l'origine peut remonter à 1977 [Glo77], a été formalisée et développée assez récemment par Glover [Glo86] et, indépendamment, par Hansen [Han86] sous le nom anglais de *steepest ascent mildest descent heuristic*. Par la suite, la RT a été, entre autres, abondamment expérimentée et perfectionnée à la chaire de Recherche Opérationnelle du Département de Mathématiques de l'École Polytechnique Fédérale de Lausanne. Cette méthode séquentielle est en fait une méta-heuristique basée sur des critères et des principes généraux. Leurs rôles consistent à contrôler et à diriger le processus interne de recherche au travers de l'espace S des solutions admissibles du problème d'optimisation combinatoire étudié afin de lui permettre de progresser au-delà de l'obstacle des optima locaux.

À l'inverse de la méthode du recuit simulé qui fait appel à des techniques probabilistes pour atteindre cet objectif, la RT utilise la notion de mémoire. L'idée principale, inspirée de techniques de recherche issues du domaine de l'intelligence artificielle, consiste à garder en mémoire un historique du processus de recherche et à utiliser cette information pour le diriger efficacement dans l'espace des solutions.

Comme dans la plupart des algorithmes itératifs de recherche, le moteur du processus de recherche est défini par les mécanismes d'évaluation de l'ensemble N(s) des solutions voisines de la solution courante s. Ceux-ci doivent impérativement être efficaces pour que la méthode le soit aussi. Lorsque la taille du voisinage n'est pas trop grande, la recherche pourra faire un examen exhaustif de celui-ci. Mais de façon générale, seul un sous-ensemble $V(s) \subseteq N(s)$ de solutions voisines sera examiné. En effet, une façon d'accélérer l'examen d'un voisinage, qui est un processus coûteux du point de vue du temps de calcul et des ressources informatiques, est de réduire sa taille.

À l'opposé des méthodes séquentielles présentées précédemment, la RT décrit N(s) à l'aide d'un ensemble M(s) de transformations locales applicables à la solution s. N(s) se définit comme l'ensemble des solutions admissibles s' qui peuvent être atteintes depuis s par l'application d'un seul mouvement m appartenant à M(s). L'application de m à s

sera notée $s \oplus m$ et on aura $N(s) = \{s' \in S \mid s' = s \oplus m, m \in M(s)\}$. Le principal avantage de l'expression de N(s) en termes de mouvements est que généralement l'ensemble M(s) se caractérise assez facilement. Un mouvement pour le PVC pourrait être, par exemple, un échange 2–opt.

À chaque pas de la méthode, N(s) est examiné en procédant par énumération ou par échantillonnage selon les problèmes étudiés. Comme dans le cas du recuit simulé, des détériorations de la fonction objectif sont autorisées pour éviter que le processus ne s'enlise dans un optimum local. En effet, au lieu de sélectionner le voisin donnant la plus grande amélioration de la fonction objectif comme dans une méthode de descente classique on choisit, en l'absence d'un tel voisin, celui qui donne la plus petite détérioration de la fonction objectif. Le voisin qui répond à ces critères est appelé le *meilleur voisin* de la solution courante s. Un tel choix a pour but de permettre au processus de recherche d'atteindre et d'explorer de nouvelles régions de l'espace S qui pourraient contenir des solutions meilleures que celles déjà visitées.

Cette politique de choix d'un voisin rend donc la RT moins sensible aux optima locaux, mais introduit malheureusement un cas de figure désagréable : le risque de cyclage du processus de recherche. En effet, lorsque le processus de recherche se déplace d'un optimum local s du problème étudié vers une solution voisine s' de moins bonne qualité, le risque existe qu'il retourne en s à l'itération suivante (voir Figure 2.1).

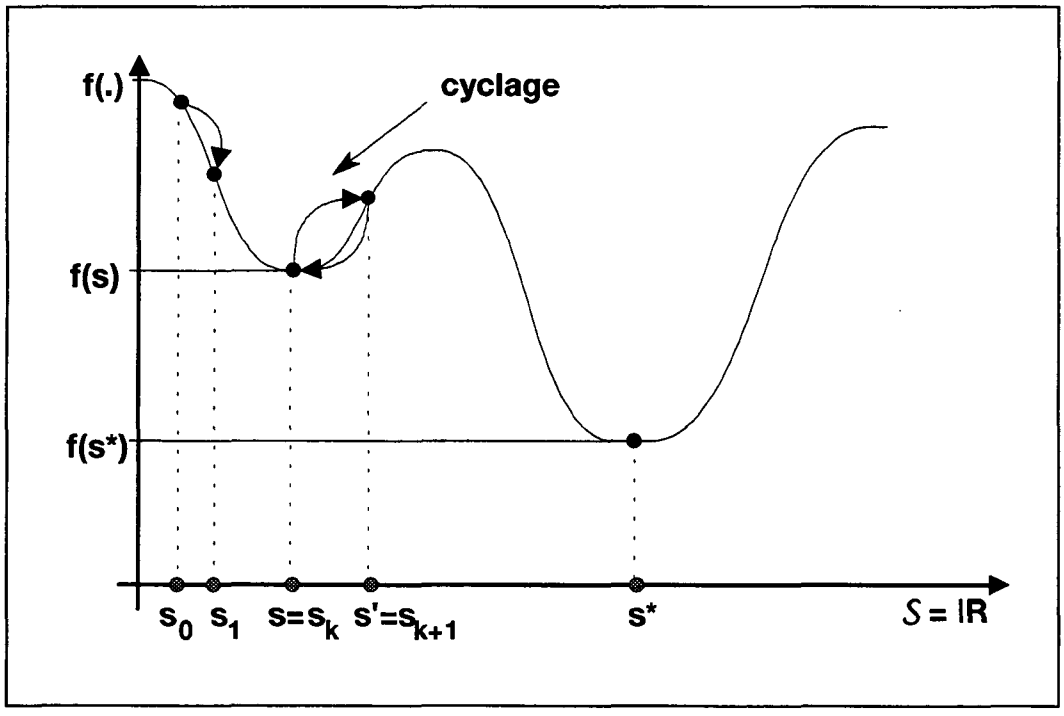


Figure 2.1 : Le phénomène de cyclage de la RT.

Pour tenter d'éviter un tel phénomène, le processus de recherche a l'interdiction pendant plusieurs itérations de reconsidérer des solutions qu'il a déjà visitées précédemment. C'est dans ce cas de figure qu'intervient la notion de mémoire à court terme dans la RT. Une approche efficace et facile à mettre en œuvre pour implanter cette idée est d'introduire des restrictions sur les mouvements qui peuvent être réalisés à chaque étape. Les mouvements interdits sont déclarés *tabous* et placés dans une liste LT appelée *liste de tabous*, d'où le nom de la méthode. Cette liste est gérée de façon cyclique en remplaçant à chaque itération le mouvement interdit le plus ancien par le mouvement empêchant de reconsidérer à la prochaine étape la dernière solution visitée. Si l'on suppose que chaque mouvement m effectué est inversible ($(s \oplus m) \oplus m^{-1} = s$), on introduira m^{-1} (l'inverse du mouvement m réalisé) dans LT à chaque itération.

Le paramètre $\Theta = |LT|$ représente la longueur de la liste de tabous et joue un rôle très important dans la méta-heuristique Tabou. Il devra être choisi avec le plus grand soin. Sa valeur, généralement déterminée de manière empirique, dépend fortement du type de problèmes étudié et de la taille du voisinage. Taillard [Tai93b] a montré qu'il est généralement préférable de considérer une longueur de la liste de tabous qui varie dynamiquement au cours de l'algorithme plutôt qu'une longueur fixe. Avec cette mémoire à court terme on espère que, lorsque l'on pourra à nouveau effectuer l'inverse d'un mouvement réalisé précédemment, la probabilité de retourner vers une solution déjà visitée soit faible bien que non nulle [Tai93b]. C'est la raison pour laquelle, en pratique, il est souvent plus judicieux d'interdire certaines caractéristiques des derniers mouvements réalisés plutôt que l'inverse de ceux-ci, même si pour cela il faut gérer plusieurs listes de tabous.

Le mécanisme des mouvement tabous, qui a pour but de tenter d'empêcher l'algorithme de cycler, peut dans certaines situations s'avérer une entrave à l'exploration efficace du domaine des solutions. En effet, ce mécanisme risque d'interdire de retourner non seulement à la solution précédente mais aussi à tout un ensemble de solutions parmi lesquelles certaines pourraient ne pas avoir encore été visitées et être de meilleure qualité que celles déjà rencontrées. Il faut donc, pour ce type de situations, un procédé qui puisse annuler le statut tabou d'un mouvement si l'application de ce dernier à la solution courante permet d'atteindre une solution jugée bonne sans créer un risque de cyclage de l'algorithme. C'est la notion de *critère d'aspiration* que l'on peut mettre en oeuvre à l'aide d'une fonction auxiliaire A(z) appelée *fonction d'aspiration* et dont le domaine de définition s'étend à l'ensemble des valeurs z de la fonction objectif f. Étant donné une solution courante s telle que f(s) = z, A(z) représente le niveau d'aspiration associé à la

valeur z, c'est-à-dire le seuil au dessous duquel la RT est assurée de ne pas cycler. Le statut tabou d'un mouvement m sera annulé si son application à la solution courante s permet de satisfaire la condition d'aspiration suivante : $f(s \oplus m) < A(f(s))$. De nombreuses fonctions d'aspiration, dont certaines extrêmement sophistiquées, ont été proposées dans la littérature. Pour une présentation détaillée de celles-ci nous renvoyons le lecteur aux travaux de Glover et al. [Glo93a] et Glover et Laguna [Glo93b]. Nous ne considérerons ici que le critère d'aspiration le plus simple qui consiste à révoquer le statut tabou d'un mouvement si son application à la solution courante s conduit à une solution de meilleure qualité que celle de la meilleure solutions s* rencontrée jusque-là. Dans ce cas, on pose $A(f(s)) = f(s^*)$ pour toute solution $s \in S$ et le mouvement tabou m sera tout de même effectué si $f(s \oplus m) < f(s^*)$. Ce critère très sévère n'ajoute pas beaucoup de flexibilité à l'algorithme mais lui permet de ne pas "oublier" de considérer de tels mouvements si la situation décrite ci-dessus se présente.

Dans sa version de base, la méta-heuristique Tabou peut se décrire de façon générique comme suit :

Choisir ou générer une solution initiale $s \in S$; $s^* := s$;
nbiter := 0 ; { *Compteur d'itérations* }
best_iter := 0 ; { *Itération à laquelle on a trouvé la meilleure solution* s* }
LT := $\varnothing$; { *La liste de tabous est vide initialement* }
critère_arrêt := faux ; { *Variable booléenne qui indique s'il faut poursuivre le processus de recherche* }
Tant que (critère_arrêt = faux) **faire**
nbiter := nbiter + 1;
Engendrer $V(s) \subseteq N(s) = \{s' \in S \mid s' = s \oplus m, m \in M(s)\}$;
Générer $V^* = \{s' \in V(s) \mid f(s') < A(f(s))$ ou $s' = s \oplus m, m \in M(s) \setminus LT\}$;
Choisir la meilleure solution $s' \in V^*$; $s := s'$;
Si $|LT| \geq \Theta$
alors $LT = (LT \setminus \{\text{plus\_ancien\_mvt}\}) \cup \{m^{-1}\}$
sinon $LT = LT \cup \{m^{-1}\}$;
Si $f(s) < f(s^*)$
alors $s^* := s$;
best_iter := nbiter ;
Mise à jour de critère_arrêt ;

Tableau 2-4 : La méta-heuristique Tabou.

Différents critères d'arrêt de la méta-heuristique Tabou peuvent être utilisés. Les plus courants stoppent l'algorithme

- après un certain nombre d'itérations ou un certain temps de calcul,
- après qu'un certain nombre d'itérations ait été effectué sans que la meilleure solution s* n'ait été améliorée.

Le lecteur intéressé par des versions plus élaborées de l'algorithme Tabou est invité à consulter les références [Glo89, Glo90, Her92, Glo93a et Tai93b]. Certains des raffinements de la méthode comme l'intensification et la diversification de la recherche, dont le but est d'améliorer la robustesse et l'efficacité de la recherche, seront exposés dans les chapitres 4 et 5 de cette thèse.

Malgré la puissance de cette méthode et les excellents résultats obtenus pour des problèmes d'optimisation combinatoire très divers, il n'existe pas, contrairement à la méthode du recuit simulé, de résultat théorique qui garantisse la convergence de l'algorithme vers une solution optimale du problème traité. Les seuls théorèmes de convergence connus à ce jour (Faigle et Kern [Fai92]) indiquent qu'une version probabiliste de la RT, s'apparentant au recuit simulé, a une probabilité tendant vers 1 de se trouver en une solution optimale après un temps infini. Un tel résultat n'est donc pas très utile en pratique puisqu'il suffit que l'algorithme visite au moins une fois une solution optimale lors de son exploration des solutions admissibles pour qu'il soit efficace.

Une des premières adaptations de la méta-heuristique tabou pour le PTV est décrite par Willard [Wil89]. Les principaux résultats basés sur cette méthode sont dus à Pureza et Franca [Pur91], Osman [Osm91, Osm93a], Gendreau et al. [Gen94], Taillard [Tai93a] et Rochat et Taillard [Roc95]. Pour les PTV réels, il faut mentionner les études de Semet et Taillard [Sem93] et Rochat et Semet [Roc94].

2.3.3 Les heuristiques évolutives

Ces deux dernières décennies ont été marquées, entre autres, par un très net rapprochement de deux grandes disciplines qui s'étaient jusqu'alors développées indépendamment : les sciences du vivant et les sciences de l'ingénieur. Dans un premier temps, ce sont les spécialistes des sciences du vivant qui ont commencé à utiliser de façon quasi systématique les outils produits par les ingénieurs. A ce sujet, citons l'exemple significatif de l'utilisation conjointe de l'électronique et de l'informatique dans

le domaine du génie médical. Ce n'est que plus récemment que les ingénieurs, après s'être intéressés à l'intelligence artificielle et aux réseaux de neurones artificiels, se sont inspirés des processus naturels qui gouvernent la vie pour concevoir et développer des nouveaux produits et méthodes artificiels.

Les approches évolutives que nous présenterons succinctement dans cette section font partie d'une nouvelle discipline en pleine expansion appelée la *vie artificielle*. Ce nouveau domaine, situé entre la biologie, l'informatique et les mathématiques, tente d'étudier les principes dynamiques fondamentaux du monde vivant. Son but est de mieux comprendre ces principes et de s'en inspirer pour créer des systèmes artificiels manifestant des qualités comparables à celles des systèmes vivants. Cette discipline constitue un formidable outil d'investigation scientifique nous permettant d'accéder et d'explorer des sciences dont la complexité intrinsèque dépasse de beaucoup celle des systèmes développés jusque-là par les ingénieurs. Et c'est grâce aux ordinateurs, dont la puissance de calcul se multiplie par un facteur voisin de 1000 tous les dix ans, que nous avons les moyens nécessaires pour pénétrer dans ces degrés de complexité.

Dans le domaine des mathématiques en général et celui de l'optimisation combinatoire en particulier, les structures et les mécanismes d'évolution des organismes vivants ont servi de base au développement de nouvelles techniques de recherche, appelées *heuristiques évolutives*, qui sont toujours plus sophistiquées et de plus en plus performantes. Ces algorithmes évolutifs sont des méthodes itératives qui gèrent et manipulent un groupe de solutions distinctes à chaque itération du processus de recherche. De ce point de vue là, elles se distinguent des méthodes constructives et séquentielles qui se concentrent sur une unique solution. C'est l'émergence de l'intelligence collective, du règne de chacun pour tous. L'idée principale des heuristiques évolutives est donc d'exploiter au mieux les caractéristiques et propriétés d'un ensemble $\mathcal{P}$ de solutions, appelé *population*, pour guider efficacement la recherche vers des bonnes solutions de l'espace S. La taille de cette population $\mathcal{P}$ est un paramètre important de ce type de méthodes qu'il faudra déterminer avec le plus grand soin. Le processus itératif à la base d'une méthode évolutive a généralement pour objectif d'améliorer la qualité moyenne des solutions présentes dans $\mathcal{P}$.

Les heuristiques évolutives les plus connues sont les *algorithmes génétiques*. Ce sont des méthodes évolutives qui reposent sur les principes d'évolution naturelle, présentées pour la première fois par Darwin [Dar02]. C'est en quelque sorte le pendant informatique de la sélection naturelle des êtres vivants. Les principales adaptations de ces

algorithmes au PVC et au PTV sont dues à Thangiah [Tha93], Thangiah et al. [Tha91, Tha94], Potvin [Pot95a], Potvin et al. [Pot95b] et Russell [Rus95].

Il existe d'autres heuristiques évolutives telles que l'*algorithme de la fourmi*, développé initialement par Colorni et al. [Col91] puis adapté au PVC par les mêmes auteurs [Col92], la méthode de *recherche distribuée* (scatter search en anglais) [Glo93c, Glo94] ainsi qu'une méthode probabiliste de diversification et d'intensification pour les heuristiques de recherche locale du PTV proposée par Rochat et Taillard [Roc95]. Cette dernière méthode, décrite en détail dans le chapitre 5 de cette thèse, est un nouvel algorithme robuste qui se trouve à la croisée entre la méta-heuristique Tabou et les algorithmes génétiques.

En utilisant ces approches évolutives dans le domaine de l'optimisation combinatoire, nous espérons que l'interaction entre plusieurs solutions distinctes, chacune d'entre elles étant incapable d'avoir une vision claire de l'espace dans lequel elle se trouve, permettra de découvrir une région de l'espace S contenant de très bonnes solutions du problème étudié.

3. Heuristiques utilisées

Ce chapitre présente deux heuristiques pour le PTVFT qui seront utilisées dans l'ensemble des méthodes présentées lors des chapitres ultérieurs de cette première partie. Il s'agit d'une part, d'une heuristique constructive qui est utilisée comme méthode d'initialisation de nos algorithmes et d'autre part, d'une heuristique de type r–opt. Cette dernière sera appliquée soit dans le processus itératif de la méta-heuristique Tabou, soit dans la phase de post-optimisation de nos méthodes.

3.1 Heuristique constructive de Solomon

Afin de disposer d'une méthode rapide d'initialisation pour les algorithmes que nous désirons développer, nous avons implanté une heuristique constructive simple. En effet, comme nous l'avons déjà relevé dans le chapitre 2, les heuristiques sont très intéressantes d'un point de vue pratique car elles sont généralement basées sur des concepts simples à mettre en oeuvre et de surcroît, elles peuvent tenir compte d'un grand nombre de contraintes additionnelles.

Solomon [Sol87] est le premier en 1987 à avoir généralisé plusieurs heuristiques développées pour le PTV (l'heuristique de Clarke et Wright [Cla64], l'heuristique de Gillett et Miller [Gil74] ainsi que les heuristiques d'insertion et du plus proche voisin, décrites par exemple dans Rosenkrantz et al. [Ros77]) afin de pouvoir résoudre le PTVFT. En testant celles-ci sur 56 problèmes académiques, Solomon a montré que l'heuristique d'insertion a une performance et une stabilité supérieures à toutes les autres. De plus, lorsqu'elle ne devance pas les autres méthodes, son écart par rapport à la meilleure solution est très faible. Comme dit précédemment, les deux principales composantes du PTVFT sont le problème de l'affectation d'un groupe de clients à un véhicule et, celui qui influe le plus sur la construction d'une solution, l'ordonnancement de ces clients pour chaque véhicule considéré. Comme c'est l'heuristique d'insertion qui se trouve être la mieux adaptée à ce dernier problème, notre choix s'est porté sur cette heuristique comme méthode d'initialisation. Elle est suffisamment flexible et permet d'éviter des solutions irréalisables (lorsque le nombre m de véhicules n'est pas borné supérieurement) bien qu'elle ait le défaut de produire des tournées périphériques.

3.1.1 Description

L'heuristique constructive basée sur un algorithme d'insertion est une généralisation de la méthode de Mole et Jameson [Mol76] pour tenir compte des fenêtres de temps. Sa principale caractéristique est d'insérer, à un pas donné du processus d'élaboration du plan des tournées de livraison, un nouveau client entre deux clients adjacents de la tournée partielle admissible T_i déjà construite et définie par :

$$T_i = (x_{i_0}, x_{i_1}, \ldots, x_{i_j}, x_{i_{j+1}}, \ldots, x_{i_{n_i+1}}), \quad x_{i_0} = x_{i_{n_i+1}} = x_0$$

Les différentes tournées T_1, ..., T_i, ... seront construites séquentiellement. Pour chaque nouvelle tournée, on initialise ce type d'heuristiques séquentielles par l'utilisation de l'un des deux critères suivants :

1. Prendre comme premier point de livraison le client non desservi qui se trouve le plus éloigné du dépôt en distance. C'est l'initialisation *Farthest.*
2. Prendre comme premier point de livraison le client non desservi dont la limite inférieure de sa fenêtre de temps est la plus basse. C'est l'initialisation *Earliest.*

Nous allons supposer qu'initialement chaque véhicule quitte le dépôt le plus tôt possible, c'est-à-dire en d_0. Lorsque le plan définitif des tournées de véhicules aura été élaboré, l'heure de départ du dépôt de chaque véhicule sera ajustée afin d'éliminer tout temps d'attente superflu chez le premier client de chaque tournée (étape importante si les temps d'attente sont pris en compte dans la fonction objectif).

3.1.2 Les contraintes de fenêtres de temps

Un point clef du bon fonctionnement et de l'efficacité des heuristiques pour le PTVFT est l'incorporation des contraintes de fenêtres de temps dans le processus d'élaboration des tournées. Nous allons donc tout d'abord présenter les conditions nécessaires et suffisantes pour le respect de ces contraintes lors de l'insertion d'un nouveau client x_u entre les clients x_{i_j} et $x_{i_{j+1}}$ $(0 \le j \le n_i)$ de T_i pour laquelle les heures de début de service h_{i_r} ainsi que les temps d'attente w_{i_r} $(0 \le r \le n_i+1)$ sont connus. Nous noterons par $h^u_{i_r}$ la nouvelle heure de début de service chez le client x_{i_r} une fois le client u inséré sur T_i. En supposant que l'inégalité triangulaire est respectée à la fois pour les distances et les temps de trajet entre chaque paire de clients, l'insertion d'un nouveau

client permet de définir une modification d'horaire $DT_{i_{j+1}}$ (déplacement dans le temps de l'heure de début de service) chez le client $x_{i_{j+1}}$:

$$\begin{aligned} DT_{i_{j+1}} &= h^u_{i_{j+1}} - h_{i_{j+1}} \geq 0 \\ \text{avec } h_u &= \max(h_{i_j} + s_{i_j} + t_{i_j u} + a_u, d_u) \\ h^u_{i_{j+1}} &= \max(h_u + s_u + t_{u i_{j+1}} + a_{i_{j+1}}, d_{i_{j+1}}) \end{aligned}$$

Par suite, nous pouvons déterminer :

$$DT_{i_{r+1}} = \max(0, DT_{i_r} - w_{i_{r+1}}), \quad j < r \leq n_i$$

Si $DT_{i_{j+1}} > 0$, certains clients x_{i_r} ($j < r \leq n_i+1$) pourraient violer les contraintes de fenêtres de temps. Nous devons donc examiner ces clients de manière séquentielle pour tester leur admissibilité horaire jusqu'à ce que l'on trouve un client x_{i_r} ($r \leq n_i+1$) pour lequel on a :

- soit $DT_{i_r} = 0$,
- soit x_{i_r} viole les contraintes de fenêtres de temps.

Dans le plus mauvais cas de figure, il faudra examiner tous les clients x_{i_r} ($j < r \leq n_i+1$) pour s'assurer de l'admissibilité horaire de T_i après l'insertion du client x_u. Sur la base de ces remarques, nous pouvons en déduire la proposition suivante :

Proposition

Les conditions nécessaires et suffisantes du respect des contraintes de fenêtres de temps lors de l'insertion d'un client x_u entre les clients x_{i_j} et $x_{i_{j+1}}$ ($0 \leq j \leq n_i$) de la tournée partielle admissible ($x_{i_0}, x_{i_1}, \ldots, x_{i_j}, x_{i_{j+1}}, \ldots, x_{i_{n_i+1}}$) où $x_{i_0} = x_{i_{n_i+1}} = x_0$ sont :

$$h_u \leq f_u$$

$$h_{i_r} + DT_{i_r} \leq f_{i_r}, \quad j < r \leq n_i+1$$

3.1.3 Algorithme

En plus du choix du type d'initialisation (Farthest ou Earliest), la méthode d'insertion de Solomon se caractérise par 4 paramètres réels (α_1, α_2, λ et μ) vérifiant :

$$\boxed{\begin{aligned} 1 \geq \alpha_1 &\geq 0 \\ \alpha_2 = 1 - \alpha_1 &\geq 0 \\ \lambda &\geq 0 \\ \mu &\geq 0 \end{aligned}}$$

Pour chaque insertion admissible d'un client non desservi x_u entre x_{i_j} et $x_{i_{j+1}}$ sur la tournée T_i (voir Figure 3.1), nous calculons :

(i) L'accroissement de distance occasionné sur la tournée T_i :

$$\boxed{\Delta D(i_j, u) = c_{i_j u} + c_{u i_{j+1}} - \mu \cdot c_{i_j i_{j+1}}}$$

(ii) Le délai occasionné dans l'heure de début de service chez le client $x_{i_{j+1}}$:

$$\boxed{\Delta T(i_j, u) = h^u_{i_{j+1}} - h_{i_{j+1}}}$$

Sur la base de ces deux mesures, on peut déterminer les deux critères de sélection d'un nouveau client qui seront utilisés à chaque itération de l'algorithme :

(i) Le coût d'insertion du client x_u entre les clients x_{i_j} et $x_{i_{j+1}}$ sur la tournée en construction T_i. C'est une combinaison convexe des deux mesures précédentes.

$$\boxed{C_1(i_j, u) = \alpha_1 \cdot \Delta D(i_j, u) + \alpha_2 \cdot \Delta T(i_j, u)}$$

(ii) Le gain de l'insertion du client x_u sur la tournée en construction par rapport à sa desserte directe depuis le dépôt :

$$\boxed{C_2(i_j, u) = \lambda \cdot c_{0u} - C_1(i_j, u)}$$

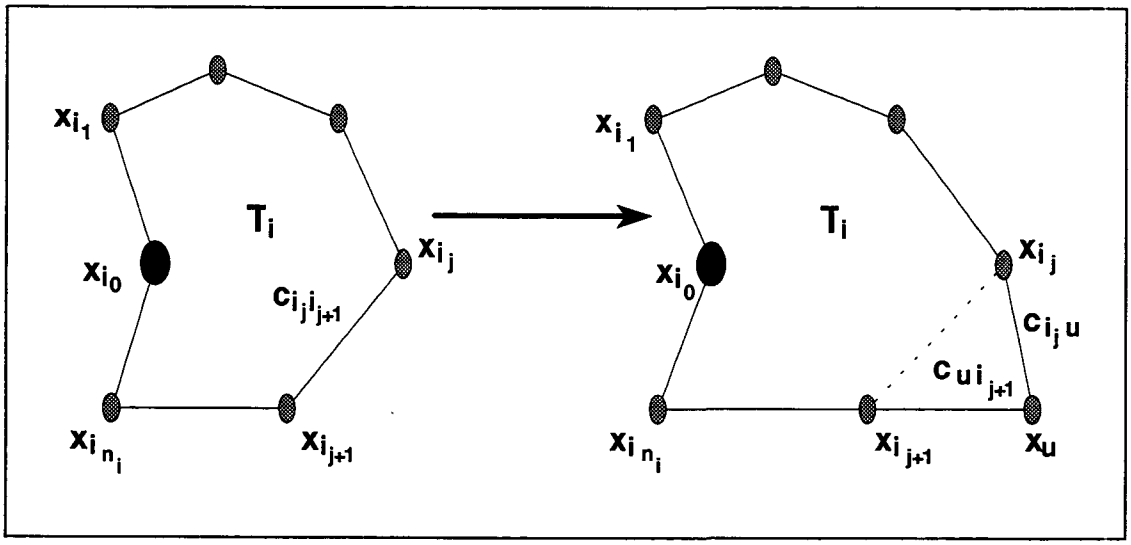


Figure 3.1 : Insertion d'un client non desservi x_u entre les clients x_{i_j} et $x_{i_{j+1}}$ de T_i.

Le but de cette heuristique d'insertion est de maximiser l'avantage que l'on obtient en desservant un client sur la tournée partielle en construction, plutôt que par un service direct depuis le dépôt. La meilleure place d'insertion admissible pour un client non desservi est celle qui minimise la combinaison convexe de la distance et du temps occasionnés par cette insertion, c'est-à-dire celle qui minimise la mesure de la distance et du temps supplémentaires requis pour desservir un nouveau client sur la tournée partielle T_i. Il est important de noter que le défaut de cette méthode réside dans l'apparition d'effets de bord nuisibles lorsque les points de livraison sont trop groupés. L'algorithme d'insertion peut se décrire de la manière suivante :

(0) $i := 0$

(1) $i := i+1$;
Initialiser T_i avec la méthode Farthest ou Earliest ;

(2) Pour tout client admissible x_u non desservi, calculer sa meilleure position d'insertion dans T_i entre $x_{i_{j(u)}}$ et $x_{i_{j(u)+1}}$ comme suit :

$$C_1(i_{j(u)}, u) = \min_{r=0,\ldots,n_i-1} C_1(i_r, u)$$

(3) Déterminer ensuite le meilleur client x_{u*} devant être inséré sur T_i entre $x_{i_{j(u*)}}$ et $x_{i_{j(u*)+1}}$:

$$C_2(i_{j(u*)}, u*) = \max_{x_u \text{ non desservi et admissible}} C_2(i_{j(u)}, u)$$

(4) Insérer x_{u*} dans T_i entre $x_{i_{j(u*)}}$ et $x_{i_{j(u*)+1}}$;

(5) **Si** tous les clients sont desservis
alors STOP
sinon **Si** plus aucun client n'est admissible
alors Aller en (1)
sinon Aller en (2) ;

Tableau 3-1 : Heuristique d'insertion pour le PTVFT.

3.1.4 Quelques considérations sur les paramètres

En posant α_1 à 1 et α_2 à 0 on favorise la minimisation de la distance totale parcourue alors qu'en fixant α_1 à 0 et α_2 à 1 on cherche à minimiser la durée totale des tournées, incluant les temps d'attente. Toute autre combinaison pour ces deux

paramètres minimise la combinaison convexe des coûts d'insertion dus à la distance et à la durée des tournées. Remarquons aussi que différentes valeurs de λ et μ conduiront à différents critères de sélection pour l'insertion d'un nouveau client sur la tournée partielle en construction.

Notons de plus, que lorsque $\alpha_2 = 0$ cette heuristique n'est rien d'autre que l'heuristique proposée par Mole et Jameson [Mol76] en 1976.

Dans le cadre des problèmes que nous avons étudiés, les couples de paramètres (α_1, α_2) et (λ, μ) suivants ont été considérés pour chaque type d'initialisation :

$$\alpha_1 = 0.0, 0.1, \ldots, 1.0$$
$$\alpha_2 = 1 - \alpha_1$$
$$(\lambda, \mu) = \begin{cases} \lambda = 1.25, 1.50, 1.75, 2.00 & \text{et} \quad \mu = \lambda - 1 \\ \lambda = 0.0, 0.5, 1.0, 1.5 & \text{et} \quad \mu = 1 \end{cases}$$

Tableau 3-2 : Choix des paramètres pour l'heuristique d'insertion de Solomon.

Comme 11 valeurs pour (α_1, α_2), 8 valeurs pour λ et μ et deux types d'initialisation ont été définis, il y a 176 combinaisons possibles de ces différents paramètres. Toutefois, il est déconseillé de perdre trop de temps sur le choix de ces paramètres. Dans la conclusion de son étude, Solomon recommande l'utilisation d'environ 20 de ces combinaisons pour obtenir, dans des temps de calcul raisonnables, d'excellentes solutions initiales admissibles pour le PTVFT.

3.2 Méthode d'amélioration de type r–opt

Il existe peu d'articles dans la littérature traitant des méthodes d'amélioration pour le PTVFT. La difficulté majeure pour la mise au point de telles méthodes provient de l'orientation imposée aux tournées élaborées par les contraintes de fenêtres de temps. Ceci ne permet pas d'appliquer aisément les procédures d'échanges d'arcs de la méthode de type r–opt décrite en section 2.3.2.1.2 . En effet, leur utilisation risque de créer une nouvelle tournée qui ne respecterait plus les contraintes d'admissibilité horaire.

Baker et Schaffer [Bak86] ont mené une étude sur des méthodes d'amélioration pour le PTVFT qu'ils ont testées sur des solutions approchées de divers problèmes-tests. Celles-ci sont des extensions des procédures d'échanges d'arcs 2–opt et 3–opt de Lin

[Lin65], adaptées pour tenir compte des contraintes de capacité et de fenêtres de temps du PTVFT. Leurs résultats indiquent que leurs méthodes sont efficaces dans l'amélioration de la qualité des solutions produites mais par contre, les temps de calcul requis pour les obtenir sont très importants.

Pour remédier à ce problème, Solomon et al. [Sol88a] ont suggéré l'idée d'employer des méthodes d'accélération des procédures d'échange d'arcs qui sont des généralisations des algorithmes de Lin et Kernighan [Lin73] et Or [Or76]. L'originalité de leur approche réside dans une implémentation astucieuse et efficace de ces algorithmes qui a pour effet d'accélérer les contrôles des contraintes de fenêtres de temps. Rappelons que ces contrôles sont nécessaires pour assurer l'admissibilité de la tournée traitée. Cette implémentation, obtenue en exploitant la structure inhérente au PTVFT à l'aide d'un pré-traitement des données, a conduit à une amélioration significative des temps de calcul reportés par Baker et Schaffer sans aucune dégradation sur la qualité des solutions obtenues.

A la vue de ces résultats, nous avons opté pour la méthode de Solomon comme heuristique d'amélioration. Par conséquent, lorsque nous ferons référence à un algorithme de type r–opt dans la suite de ce texte, il s'agira de cette méthode.

4. Les problèmes issus de la réalité

4.1 Introduction

Tôt ou tard, l'entreprise dont le fonctionnement est tributaire du transport de ses produits se rend compte que des gains substantiels pourraient être réalisés grâce à une bonne gestion des moyens disponibles.

Trois aspects primordiaux interviennent dans la gestion des transports de marchandises par route :

1. La prise en compte d'une multitude de contraintes.
2. L'exploitation rationnelle du réseau routier à disposition.
3. L'utilisation optimale des véhicules à disposition.

4.1.1 Les contraintes

Comme nous l'avons déjà mentionné dans la présentation générale du problème d'élaboration de tournées de véhicules, les modèles présentés dans la littérature sont le plus souvent des simplifications excessives des problèmes rencontrés dans la pratique. En effet, les objectifs et les contraintes à prendre en compte, tant au niveau de l'élaboration des tournées qu'à celui de la gestion de la flotte de véhicules et de leurs chauffeurs ou du service à la clientèle, sont nombreux et des plus variés dans la résolution d'un problème réel. De plus, certaines de ces contraintes qui répondent à des aspects pratiques de la situation étudiée sont difficiles à modéliser. Nous avons en effet remarqué que pour le responsable des transports d'une entreprise, les contraintes sont des *exigences souples* c'est-à-dire des restrictions qui peuvent être levées en certaines occasions et à certains coûts. Ce constat peut être illustré à l'aide des deux exemples suivants. Premièrement, celui d'une contrainte horaire de livraison chez un client qui peut parfois être violée de quelques minutes pour permettre une amélioration du plan de transport sans toutefois engendrer des pénalités économiques significatives. Le deuxième exemple concerne la contrainte de charge utile d'un véhicule. En dehors de la nécessité physique de la satisfaire, elle s'avère être une exigence souple dans beaucoup de problèmes réels. En

effet, le responsable des transports ne la respectera qu'approximativement si il estime que le gain procuré, au niveau de la fonction objectif par le non respect de cette contrainte, est suffisamment important pour courir le risque d'être amendé. Ce type de considérations est généralement traité au sein du système informatique d'aide à la décision qui englobe les algorithmes de résolution du problème d'élaboration de tournées de véhicules. Les méthodes que nous avons développées tiennent compte des caractéristiques suivantes que nous avons regroupées par catégories. Le lecteur intéressé par une présentation détaillée et complète des contraintes du PTV peut se référer à Ronen [Ron88].

Caractéristiques pour la flotte de véhicules

- nombre de véhicules,
- différents types de véhicules (flotte hétérogène) :
  - 2 essieux,
  - 4 essieux,
  - remorque,
  - semi-remorque,
  - camionnette,
- données techniques pour chaque véhicule :
  - capacité volumique,
  - charge utile,
  - frais fixes d'utilisation,
  - coût au kilomètre,
- flotte basée en un (plusieurs) dépôt(s),
- utilisation de transporteurs professionnels ou non.

Caractéristiques pour la gestion des chauffeurs

- nombre de chauffeurs,
- types de véhicules qu'il peut conduire,
- durée de travail et de conduite par jour,
- nombre de pauses dans une journée et leur durée.

Caractéristiques du service à la clientèle

- livraison et/ou ramassage chez chaque client,
- quantité à livrer connue à l'avance,
- degré de priorité,
- nombre et durée des fenêtres de temps pour chaque jour de service en fonction de la période de l'année,
- temps d'accès,
- temps de chargement et de déchargement,
- contraintes d'accessibilité (tous les types de camion ne peuvent pas forcément accéder chez un client pour des raisons géographiques ou de configuration des lieux,
- exigences particulières.

Caractéristiques des tournées

- longueur et / ou durée maximale,
- heures légales pendant lesquelles un véhicule peut rouler,
- relation de précédence entre les lieux de livraisons,
- équilibrage des tournées,
- tournées multiples pour un même véhicule,
- points de passage fixes.

4.1.2 Réseau routier

Le second aspect primordial cité dans l'introduction implique toutefois qu'un réseau routier informatisé (RRI ou GIS en anglais pour Geographical Information Systems) soit disponible. Dans le cadre de nos études, nous avons utilisé les bases de données relatives aux réseaux routiers suisse et lausannois qui ont été développées au sein de la chaire de recherche opérationnelle de l'École Polytechnique Fédérale de Lausanne. La base de données du réseau routier suisse à notre disposition a été créée à partir des cartes routières, à l'échelle de 1 : 50'000, de l'Office Fédéral de topographie. Elle contient des informations sur plus de 14'000 localités ou carrefours et décrit les frontières avec les pays limitrophes à la Suisse, les lacs et principaux cours d'eau ainsi que plus de 20'000 tronçons routiers. Les principales données utilisées dans nos

méthodes de résolution du PTVFT sont le nom, le canton et les coordonnées nationales pour les localités et les extrémités, la longueur hectométrique ainsi qu'un type pour les tronçons routiers. Les types définis pour ces derniers sont au nombre de 10 et dépendent de leur emplacement géographique et de leur importance. Nous parlerons donc d'*autoroutes*, de *routes de plaine*, de *routes de ville* ou de *routes de montagne* en sachant que les trois derniers types cités se subdivisent en route de 1ère, 2e ou 3e catégorie.

Des vitesses moyennes (km/heure) sont ensuite affectées à chacun de ces dix types de routes du réseau, comme décrit dans le Tableau 4-1. Celles-ci ont été choisies de façon à se calquer le mieux possible à la réalité. Nous nous sommes basés pour cela sur les vitesses moyennes enregistrées lors de livraisons réellement effectuées, par les camions des sociétés concernées, certains jours représentatifs de l'année. Notons de plus que nos méthodes sont suffisamment souples pour autoriser la modifications de ces vitesses en fonction du type de véhicules utilisés, de la saison ou de la période d'une journée.

TYPE DE ROUTES	CATÉGORIE		
	1ère	2e	3e
Autoroute	70	—	—
Plaine	55	50	30
Ville	40	25	10
Montagne	40	30	10

Tableau 4-1: Vitesses moyennes ([km/h]) utilisées par type de routes.

Pour la ville de Lausanne, des informations similaires à celles contenues dans le réseau routier suisse sont disponibles pour plus de 2'300 carrefours et de 4'500 tronçons routiers. Les seules différences significatives sont l'absence des coordonnées nationales, l'utilisation du décamètre en lieu et place de l'hectomètre comme unité de distance et surtout le fait que le réseau est orienté, ceci afin de tenir compte des sens uniques.

Grâce à ces bases de données, nous sommes à même de calculer les durées et les distances des trajets optimaux (par exemple le plus court ou le plus rapide) entre deux localités ou carrefours du réseau. Pour ce faire, nous utilisons une version modifiée de l'algorithme de Dijkstra, proposée par Bovet [Bov86], qui fournit le trajet optimal en indiquant les localités ou carrefours de passage ainsi que les distances et les temps intermédiaires depuis la localité de départ.

Ces bases de données sont complétées et mises à jour régulièrement afin de fournir des informations aussi exactes et précises que possible. C'est un élément clef pour déterminer de bonnes solutions au problème d'élaboration de tournées de véhicules.

4.1.3 Objectifs à atteindre

Si la formulation du problème se prête à une modélisation précise, on cherchera éventuellement une méthode fournissant la meilleure solution selon un critère de minimisation des coûts ou de quelqu'autre mesure de qualité. Dans ce cas, le but de la fonction objectif est de minimiser une combinaison pondérée des facteurs suivants :

- le nombre de véhicules,
- la distance totale parcourue,
- la durée totale (avec ou sans temps d'attente),
- le coût total du plan de tournées.

Cette fonction peut aussi inclure un facteur représentant les pénalités pour le non respect de certaines contraintes du problème (fenêtres de temps, accessibilité de la flotte de véhicules).

Mais en général, la situation est relativement complexe si bien que toutes les contraintes ne sont pas susceptibles d'être exprimées en termes mathématiques. Dès lors, on ne se contente pas de chercher la meilleure solution puisque l'on ne sait peut-être pas exprimer le critère de qualité d'un résultat, mais un éventail de bonnes solutions. Parmi celles-ci, le responsable des transports fera son choix sur la base de critères qui lui sont propres.

4.1.4 Conception d'un système d'aide à la décision

La conception des systèmes informatiques d'aide à la décision dans le domaine de la distributique a très nettement évolué depuis le début des années septante. À cette époque, un des premiers systèmes entièrement automatique à apparaître sur le marché a été le système «VSPX» d'IBM qui incorporait des heuristiques simples de première génération (par exemple la méthode de Clarke et Wright [Cla64]). Bien que de nombreuses entreprises aient testé ce type de systèmes, très peu d'entre elles en furent satisfaites. La raison de cette déception tenait principalement dans le fait que les méthodes utilisées n'étaient pas assez sophistiquées et souples d'emploi pour permettre

la gestion de problèmes réels complexes. Malgré la correction des erreurs de jeunesse de ces systèmes informatiques d'aide à la décision entièrement automatisés et l'apparition sur le marché de méthodes de résolution du PTV plus efficaces et intelligentes (voir section 2.3.2.2), l'utilisation dans la pratique de tels systèmes reste marginale.

Dans un article récent, Fisher [Fis95] se pose la question de savoir pourquoi ce type de systèmes n'est pas plus implanté dans l'industrie si les méthodes de résolution utilisées sont si efficaces. Question d'autant plus pertinente si l'on se réfère à un article de Hooban [Hoo88] qui indique qu'aux USA, 100 à 200 mille entreprises entretiennent une flotte privée de véhicules. Le marché potentiel pour de tels logiciels est donc considérable.

Le décalage entre ce marché potentiel et le nombre de systèmes installés est dû à plusieurs causes. Par exemple au manque de formation des responsables de la logistique dans les entreprises ou à une certaine méfiance par rapport aux systèmes informatisés. Un élément de réponse qui nous semble plus important est le manque de robustesse des algorithmes mathématiques employés, ce qui rend difficile leur transfert d'une compagnie à une autre ou leur utilisation dans une région géographique différente. A mesure que le réel besoin d'outils robustes pour la résolution du PTV se faisait sentir au cours des dix dernières années, les ressources aussi bien matérielles qu'intellectuelles pour atteindre un tel but ne cessaient de croître. La précision des données nécessaires au calcul de la distance, du temps et du coût du trajet entre deux localités a énormément évolué avec l'apparition sur le marché de réseaux routiers informatisés fiables. De plus, l'évolution rapide de la puissance de calcul des ordinateurs a permis d'obtenir des solutions de qualité toujours supérieure dans des temps de calcul de plus en plus faibles.

Nous pouvons imaginer plusieurs approches pour développer des systèmes robustes. La plus simple pourrait être la création de logiciels où l'interactivité joue un rôle important, permettant par exemple à l'utilisateur de :

- modifier manuellement des solutions proposées par l'ordinateur,
- corriger d'éventuels problèmes,
- choisir l'algorithme de résolution en fonction d'une situation précise ou de critères propres,
- trouver un équilibre acceptable lorsqu'un problème a des objectifs conflictuels.

De tels systèmes ont pour but d'aider les responsables de la logistique mais en aucun cas de les remplacer. Bien que les modules d'optimisation doivent être et rester une composante majeure des systèmes d'aide à la décision, c'est le mariage du savoir faire de l'utilisateur avec la flexibilité et la puissance de calcul des ordinateurs qui permet d'obtenir un système informatisé efficace et robuste.

Une approche plus complexe pour créer des outils robustes pour la résolution du PTV est de développer une nouvelle génération d'algorithmes : les méthodes évolutives (voir section 2.3.3). Nous présenterons en détail, dans le chapitre 5, une nouvelle technique appartenant à cette catégorie. En plus d'être très efficace et robuste, cette approche peut être aisément parallélisée sur un nombre arbitraire de processeurs. Cette particularité, entre autres, lui procure un indéniable avantage sur les algorithmes développés à ce jour.

4.2 Un problème réel de distribution

Cette section traite d'un problème réel d'élaboration de tournées de véhicules rencontré par une importante entreprise suisse de production et de distribution de nourriture pour bétail. Les coûts de production de cette société étant relativement bas, les coûts inhérents au transport et à la distribution de leurs produits représentent un grand pourcentage de leurs coûts totaux. Les membres de la direction étaient donc conscients des gains substantiels qui pourraient être réalisés avec une gestion rationnelle et efficace de leur système de distribution mais ils éprouvaient de grandes difficultés à élaborer des tournées de véhicules satisfaisant la multitude de contraintes de leur problème. Les fâcheuses conséquences de cet état de fait étaient d'une part, une diminution de la qualité du service à leur clientèle et d'autre part, un risque d'être sévèrement amendé pour n'avoir pas respecté les ordonnances fédérales sur les transports en vigueur en Suisse.

4.2.1 Description du problème

Le problème fondamental de la société consiste en l'élaboration effective des tournées de véhicules de n'importe quel jour de la semaine sur la base des commandes de leur clientèle connues le jour précédent. Dans cette section, nous présentons deux méthodes heuristiques dont l'objectif est de minimiser la distance totale parcourue par les véhicules de l'entreprise tout en satisfaisant la grande variété de contraintes du problème.

La société, implantée en Suisse Romande, produit essentiellement de la nourriture animale ainsi que diverses sortes de farines. Ces produits sont vendus à la clientèle dans des sacs pesant entre 25 et 50 kilos pour un poids moyen de 42 kilos. La production se déroule en majeure partie le jour, mais parfois également la nuit.

4.2.1.1 Les clients

Les clients actifs de l'entreprise, au nombre de 5000 environ et principalement situés en Suisse Romande, se classifient en deux groupes distincts en fonction de leurs fréquences de livraison :

- les *petits clients*, pour la plupart des paysans, qui sont ravitaillés une fois par mois et représentent 70 à 80 % de la clientèle et 45 à 50 % des commandes,
- les *gros clients*, généralement des grossistes, des boulangers ou des sociétés agricoles, qui ont une fréquence de livraison comprise entre une fois par semaine et une fois toutes les 3 semaines.

Certains clients de la société ne sont pas accessibles par tous les types de véhicules. En effet, un client peut, de par sa position géographique, devenir inaccessible pour des camions avec remorques, ou, en raison des restrictions de circulation sur son chemin d'accès, être interdit aux camions avec 4 essieux. Ainsi, pour tenir compte de cette contrainte, chaque client s'est vu attribuer une liste de camions pouvant accéder chez lui. Cette liste est bien évidemment une sous-liste de l'ensemble de la flotte de véhicules de la société. Si aucune contrainte d'accessibilité n'est précisée, une valeur par défaut est attribuée à cette liste, à savoir l'ensemble de la flotte de véhicules.

Les livraisons à la clientèle ne peuvent se dérouler que dans des plages horaires bien précises, généralement fixées par les clients eux-mêmes et correspondant à leurs heures d'ouvertures ou de disponibilités. Pour tenir compte de cette contrainte, des fenêtres de temps ont été attribuées, pour chaque jour de la semaine, à l'ensemble des clients de la société. Celles-ci se caractérisent par leur nombre (0 si le client est fermé toute la journée, 1 ou 2) ainsi que par les extrémités de chacune d'entre elles (par exemple 8h00-11h30 et 13h30-16h00 s'il y a deux fenêtres de temps). Généralement, les clients ne seront pas desservis pendant la pause de midi. Si aucune contrainte de temps n'est spécifiée pour un client, une fenêtre de temps par défaut (5h00-20h00) lui est attribuée.

4.2.1.2 La flotte de véhicules

La clientèle est desservie à partir du dépôt, situé aux alentours de Lausanne, avec une flotte hétérogène de 14 véhicules pouvant tirer des remorques. Chaque camion est caractérisé par sa charge utile et sa capacité volumique. Cependant, en raison de la nature des produits transportés, la capacité volumique ne sera pas prise en considération et seule la contrainte portant sur le poids des produits doit être considérée lors du chargement des camions. Un véhicule, dont la charge utile varie entre 13 et 17 tonnes, effectue en moyenne une vingtaine de livraisons par jour. Le nombre de commandes journalières se situe entre 100 et 170, ce qui représente un volume de livraison quotidien d'environ 150 tonnes.

Nous distinguons trois phases différentes lors du déroulement d'une tournée, à savoir :

1. Le trajet d'un client à un autre selon le plan de route élaboré. La durée d'un tel trajet est appelée *temps de route.*
2. Les manoeuvres d'approche, qui consistent à trouver le chemin allant du centre du village chez le client, à se parquer et finalement à débâcher le véhicule. La durée totale de ces opérations pour un client, constituée d'un temps fixe de 10 minutes plus d'un temps variable en minutes spécifique à chaque client, est appelée *temps d'accès.*
3. Le déchargement de la commande. La durée de cette manoeuvre pour un client x_i $(0 \leq i \leq n)$, appelée *temps de service*, est égale à $(10+\sigma_i)$ minutes par tonne livrée où σ_i est un temps variable spécifique à chaque client x_i.

La durée totale d'une tournée inclut les temps de route, les temps d'accès, les temps de service et les éventuels temps d'attente qui surviennent lorsqu'un chauffeur arrive chez un client en dehors de ses fenêtres de temps. Par décision des dirigeants de la société, cette durée ne doit pas dépasser une durée maximale D_{max} = 10h15.

Dans le souci de cerner le mieux possible la réalité, nous avons décidé de tenir compte des pauses des chauffeurs lors de l'élaboration du plan de route d'une tournée de livraison. Nous nous sommes basés pour cela sur l'ordonnance fédérale sur la durée du

travail et du repos des conducteurs professionnels de véhicules automobiles (ci après l'O.T.R.) dont les points importants sont les suivants :

1. Le service au volant, c'est-à-dire la durée de la conduite, ne doit pas excéder 9 heures par jour.
2. Après 5 heures 1/2 de travail ininterrompu et au plus tard après 4 heures de conduite ininterrompue, le chauffeur fera une pause d'au moins une heure.
3. Est réputée ininterrompue, la durée de travail ou de la conduite qui n'est pas coupée par une pause de 30 minutes au moins.

Nous avons décidé de ne considérer que deux pauses par jour d'une durée totale de 1h30, à savoir une le matin d'une durée de 30 minutes et une à midi d'une durée de 60 minutes. Cette limitation du nombre de pauses fait suite au raisonnement suivant :

Traditionnellement la pause de midi a lieu entre 11h45 et 13h00. Cela permet d'une part de la faire coïncider avec les heures de fermeture habituelles des commerces (12h00-13h30 ou 12h00-14h00) et d'autre part d'autoriser les chauffeurs à travailler jusqu'à 18h15 (11h45 + 1h00 + 5h30) sans prendre une nouvelle pause tout en respectant l'O.T.R. Comme à l'heure actuelle les chauffeurs quittent le dépôt à 5h00 en raison des fenêtres de temps de certains clients, il faut qu'ils aient achevé leur travail à 16h45 (5h00 + 1h30 + D_{max}) pour respecter la durée maximale d'une tournée. Par conséquent, il n'est pas nécessaire de considérer une pause durant l'après-midi. Notons aussi que de par la position géographique des clients de la société, aucun trajet ne dure plus de 4 heures. Finalement, afin que la pause de midi puisse se dérouler entre 11h45 et 13h00 et que l'O.T.R. soit respecté, nous avons décidé de placer la pause du matin des chauffeurs entre 7h30 et 10h00 (10h00 < 05h00 + 05h30 et 13h00 < 7h30 + 0h30 + 5h30).

En résumé, notre problème d'optimisation des plans de transport de la société consiste en l'élaboration d'un ensemble de tournées qui minimise la distance totale parcourue tout en satisfaisant les contraintes suivantes :

contraintes classiques :

- chaque tournée se termine à son point de départ, le dépôt,
- chaque commande est livrée par un unique véhicule,

- le poids total des produits transportés par chaque véhicule ne peut excéder sa charge utile,
- la durée totale de chaque tournée ne doit pas dépasser D_{max}.

contraintes additionnelles :

- chaque client n'est accessible que par un sous-ensemble déterminé de véhicules,
- chaque commande doit être livrée au cours d'une des fenêtres de temps prédéfinies pour chaque client,
- chaque chauffeur doit faire deux pauses, l'une d'une demi-heure entre 7h30 et 10h et l'autre d'une heure entre 11h45 et 13h00, lors du déroulement de la tournée qu'il effectue.

4.2.2 Les notations

Avant de décrire les méthodes heuristiques proposées, nous introduisons ici les notations, complémentaires à celles définies dans la section 1.4, qui seront utilisées par la suite.

Chaque client x_i doit être desservi au cours d'une de ses deux fenêtres de temps, définies par leurs bornes minimales et maximales exprimées en minutes, comme décrit dans la Figure 4.1. Les bornes d_i^j et f_i^j sont, respectivement, les limites de temps inférieures et supérieures de la fenêtre de temps j, j = 1, 2, du client x_i. Lorsqu'un client ne peut être desservi que dans une seule fenêtre de temps, nous avons alors $d_i^2 = d_i^1$ et $f_i^2 = f_i^1$.

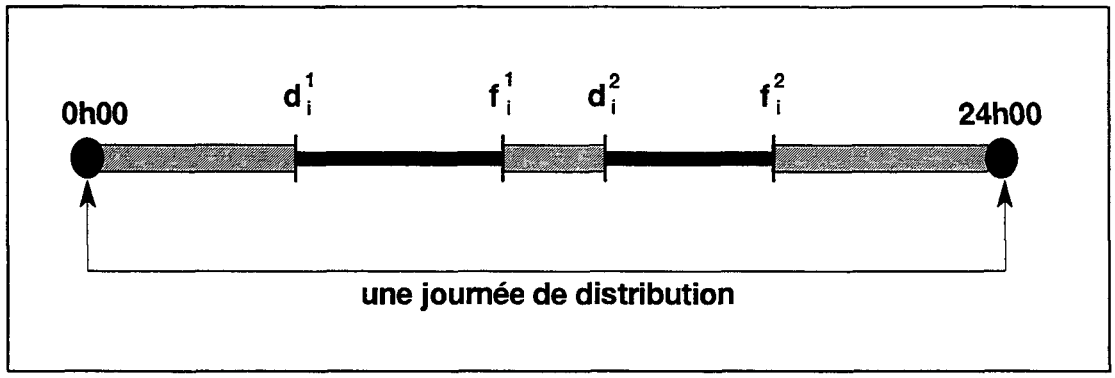


Figure 4.1 : Fenêtres de temps pour chaque client x_i.

Afin de modéliser efficacement les pauses des chauffeurs, nous introduisons deux clients *fictifs*, x_{n+1} et x_{n+2}, pour représenter respectivement la pause du matin et la pause de midi. Ces clients fictifs satisfont les conditions suivantes :

- une pause se déroule toujours après le déchargement de la commande chez un client x_i, $0 \leq i \leq n$,
- ces clients fictifs ont des fenêtres de temps spécifiques qui correspondent aux plages horaires 7h30-10h00 et 11h45-13h00. Exprimées en minutes, les bornes de ces fenêtres de temps sont : $d^1_{n+1} = d^2_{n+1} = 450$, $f^1_{n+1} = f^2_{n+1} = 600$ pour la pause matinale et $d^1_{n+2} = d^2_{n+2} = 705$, $f^1_{n+2} = f^2_{n+2} = 780$ pour la pause de midi,
- les temps d'accès à ces clients fictifs sont évidemment nuls : $a_{n+1} = a_{n+2} = 0$,
- le temps de service de chacun de ces deux clients fictifs est égal à la durée de leur pause : $s_{n+1} = 30$ et $s_{n+2} = 60$,
- la longueur c_{ij} et la durée t_{ij} du trajet entre les clients x_i et x_j doivent se calculer différemment lorsque x_i et/ou x_j sont des clients fictifs. Ainsi, nous avons trois cas particuliers qui peuvent se présenter :

  1. x_i et x_j sont des clients fictifs. Alors $c_{ij} = t_{ij} = 0$.
  2. Seul x_j est un client fictif. Alors $c_{ij} = t_{ij} = 0$.
  3. Seul x_i est un client fictif. Alors $c_{ij} = c_{rj}$ et $t_{ij} = t_{rj}$ où x_r est le premier client non fictif qui précède x_i sur la tournée considérée. Bien que formellement un double indiçage devrait être introduit dans ce cas pour indiquer la tournée considérée, nous nous en abstiendrons dans la suite de cette étude pour simplifier les notations.

- le poids des commandes associées aux clients fictifs sont nulles : $q_{n+1} = q_{n+2} = 0$.

De manière à considérer ces clients fictifs dans nos notations, nous redéfinissons l'ensemble X des clients comme suit : $X = \{x_0, \ldots, x_i, \ldots, x_n, x_{n+1}, x_{n+2}\}$, chaque client x_i, $i \geq n+1$, devant être servi par tous les camions de la société.

Avant de présenter notre première méthode heuristique, précisons que les valeurs numériques des paramètres contrôlant nos algorithmes, pour les données traitées, seront indiquées dans la section 4.2.5.

4.2.3 L'heuristique d'initialisation

Notre approche pour résoudre ce problème réel d'élaboration de tournées consiste à combiner deux méthodes heuristiques :

1. un algorithme simple et rapide,
2. une méthode basée sur les techniques de recherche avec tabous.

L'intérêt de la première méthode est double : d'une part pouvoir générer une solution initiale pour la seconde méthode et d'autre part, améliorer la flexibilité du futur logiciel en fournissant après quelques secondes de calcul une solution admissible au responsable des transports. Ce dernier aura donc la possibilité d'obtenir rapidement plusieurs solutions différentes du problème, simplement en modifiant les paramètres de l'algorithme. La méthode heuristique simple que nous avons implantée est une généralisation d'un algorithme constructif, présenté dans le chapitre 3, proposé par Solomon [Sol87]. L'intérêt de notre généralisation réside dans la possibilité de prendre en compte les contraintes d'accessibilité et les pauses des chauffeurs lors de l'élaboration des tournées. Les extensions apportées à la méthode originale sont décrites dans les paragraphes suivants.

L'association des pauses des chauffeurs avec des clients fictifs facilite leur prise en compte. En effet, au lieu de considérer que les tournées sont vides avant leur initialisation, i.e. $T_i = (x_{i_0}, x_{i_1})$ avec $x_{i_0} = x_{i_1} = x_0$ pour tout véhicule v_i, nous considérons qu'elles sont de la forme suivante :

$$T_i = (x_{i_0}, x_{i_1}, x_{i_2}, x_{i_3}) \quad \text{avec} \quad x_{i_0} = x_{i_3} = x_0, \quad x_{i_1} = x_{n+1} \quad \text{et} \quad x_{i_2} = x_{n+2}$$

Dès lors, la seule modification à apporter à la méthode originale de Solomon est de calculer différemment l'accroissement de la longueur de la tournée T_i, $\Delta D(i_j, u)$, lorsque x_{i_j} et/ou $x_{i_{j+1}}$ sont des clients fictifs. Ainsi, nous avons :

si $((x_{i_{j+1}} = x_{n+1}) \text{ et } (x_{i_{j+2}} \neq x_{n+2}))$ ou $((x_{i_{j+1}} = x_{n+2}) \text{ et } (x_{i_j} \neq x_{n+1}))$
alors $\Delta D(i_j, u) = c_{i_j u} + c_{u i_{j+2}} - \mu \cdot c_{i_j i_{j+2}}$

si $(x_{i_{j+1}} = x_{n+1}) \text{ et } (x_{i_{j+2}} = x_{n+2})$
alors $\Delta D(i_j, u) = c_{i_j u} + c_{u i_{j+3}} - \mu \cdot c_{i_j i_{j+3}}$

si $(x_{i_j} = x_{n+1}) \text{ et } (x_{i_{j+1}} = x_{n+2})$
alors $\Delta D(i_j, u) = c_{i_{j-1} u} + c_{u i_{j+2}} - \mu \cdot c_{i_{j-1} i_{j+2}}$

Une fois ces modifications effectuées, la méthode originale peut être utilisée directement.

Pour gérer les contraintes d'accessibilité, nous avons dû appliquer une méthode à deux phases. Dans la première phase, nous résolvons un PTVFT restreint aux clients qui ont des contraintes d'accessibilité et aux véhicules susceptibles de les approvisionner. Comme le nombre de ces clients est faible en comparaison avec le nombre total de clients, la logique des tournées élaborées doit être préservée. C'est la raison pour laquelle l'objectif de cette première phase est de construire des tournées *compactes*, c'est-à-dire des tournées où les clients sont proches (en distance) les uns des autres, même si leur nombre n'est pas minimum. Les tournées partielles ainsi créées pourront être complétées, au cours de la seconde phase, en y insérant des clients non soumis aux contraintes d'accessibilité.

La première phase de notre méthode est une procédure parallèle construisant simultanément plusieurs tournées T_1, T_2, ..., T_h, $h < m$. A chaque itération, l'alternative envisagée pour un client x_u ($0 < u \leq n$) non desservi est soit son insertion sur une des h tournées partielles existantes, soit la création d'une nouvelle tournée T_{h+1}, si $h < m$, avec x_u comme seul client non fictif. Le choix entre ces deux possibilités est déterminé par le coût minimum entre celui de la création d'une nouvelle tournée valant $2 \cdot c_{0u}$ et le meilleur coût d'insertion sur l'une des h tournées existantes qui se calcule comme suit :

$$\min_{i=1,\ldots,h} (\alpha_1 \cdot \Delta D(i_{j(u)}, u) + \alpha_2 \cdot \Delta T(i_{j(u)}, u))$$

Pour garantir la compacité des tournées élaborées, nous modifions le calcul de l'accroissement de la longueur de la tournée T_i, $\Delta D(i_j, u)$, en introduisant la notion de voisinage d'un client x_i ($0 \leq i \leq n$). Le voisinage de x_i est défini comme l'ensemble $N(x_i)$ des N_n plus proches voisins de x_i. Il est important de souligner les deux points suivants : N_n dépend de la taille du problème étudié et $N(x_i)$ peut contenir à la fois des clients soumis ou non aux contraintes d'accessibilité. Dès lors, $\Delta D(i_j, u)$ se calcule de la façon suivante :

$$\Delta D(i_j, u) = \begin{cases} c_{i_j u} + c_{u i_{j+1}} - \mu \cdot c_{i_j i_{j+1}} & \text{si } x_u \in N(x_{i_j}) \cup N(x_{i_{j+1}}) \\ \infty & \text{sinon} \end{cases}$$

Lorsque tous les clients soumis à des contraintes d'accessibilité ont été desservis, nous appliquons la seconde phase de notre méthode. Le point clef de cette seconde phase consiste en premier lieu à compléter les tournées créées précédemment en y insérant des clients non sujets aux contraintes d'accessibilité. Cette étape de complétion se déroule en parallèle, c'est-à-dire qu'à chaque itération nous déterminons quel client insérer sur quelle tournée. Dès qu'il n'est plus possible d'inclure un client sur une tournée existante, nous créons séquentiellement des nouvelles tournées, en utilisant la méthode originale de Solomon, dans lesquelles les clients restants sont desservis.

En combinant les deux extensions exposées ci-dessus, nous avons développé une nouvelle méthode heuristique d'insertion qui est capable de fournir en seulement quelques secondes des solutions admissibles du PTVFT réel étudié. Comme déjà mentionné dans la section 3.1.3, le principal défaut de ce type d'algorithme réside dans l'apparition d'effets de bord nuisibles lorsque les points de livraison sont fortement groupés. Cette situation est présente dans notre problème car il est fréquent d'avoir plusieurs commandes provenant d'un même village. Dans la prochaine section, nous proposons donc une méthode basée sur les techniques de recherche avec tabous afin tout d'abord d'éviter la manifestation de ces effets de bord et ensuite d'améliorer les solutions obtenues à l'aide de notre heuristique d'insertion.

4.2.4 Une heuristique basée sur la méta-heuristique Tabou

Nous décrivons ci-dessous une adaptation de la méthode de recherche avec tabous (RT) à notre problème réel d'élaboration de tournées. L'originalité de notre approche réside dans le rôle essentiel joué par la relaxation des contraintes ainsi que par la stratégie spécifique d'intensification dans la recherche d'une bonne solution du problème réel traité. Avant de décrire en détail ces deux points importants, définissons pour le PTVFT les quatre composantes principales sur lesquelles est basée la RT : l'espace des solutions, la caractérisation d'un mouvement dans cet espace, le voisinage d'une solution et la fonction objectif à optimiser.

4.2.4.1 Espace des solutions

Dans l'espace des solutions S de notre problème, une solution est définie comme un ensemble de tournées desservant la totalité des clients et vérifiant les contraintes d'accessibilité. Par conséquent, les contraintes sur la charge utile des camions, la durée totale des tournées et les fenêtres de temps des clients ont été relâchées. Une telle relaxation est particulièrement intéressante lorsque l'on traite de problèmes, par exemple le nôtre, qui sont fortement contraints.

En effet, la visite de solutions inadmissibles est une astuce permettant d'explorer plus efficacement l'espace des solutions admissibles et d'éviter que la recherche ne se bloque dans un optimum local. Toutefois, seules les solutions admissibles rencontrées au cours de la RT seront considérées lors de la détermination de la meilleure solution obtenue par notre algorithme.

4.2.4.2 Caractérisation et coût d'un mouvement

Un mouvement $m(x_{i_k}, T_j)$ consiste à ôter un client x_{i_k} $(0 < i_k < n+1)$ d'une tournée T_i, la tournée source, et à l'insérer dans une autre tournée T_j, la tournée cible (voir Figure 4.2). Nous considérerons qu'un tel mouvement est admissible, et par conséquent applicable, si et seulement si le client x_{i_k} peut être approvisionné par le véhicule v_j effectuant la tournée cible T_j.

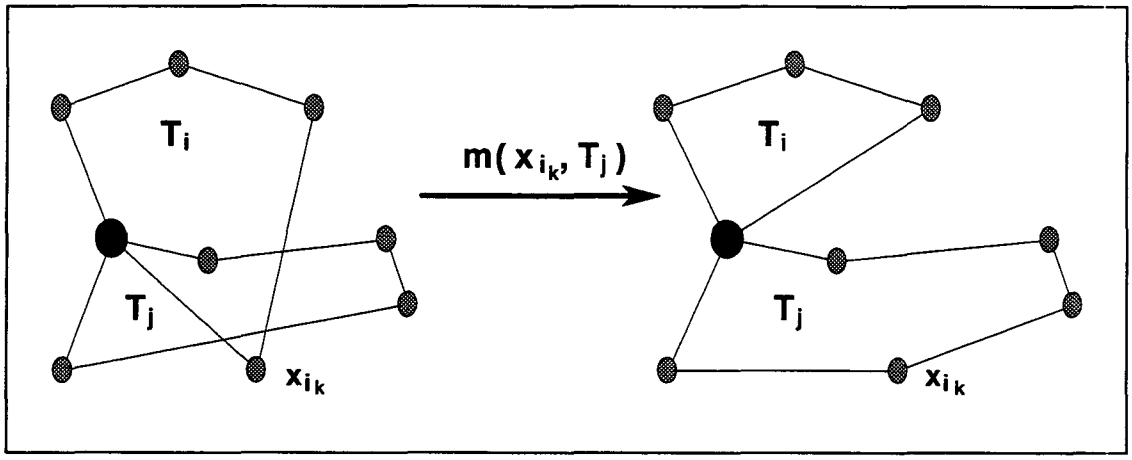


Figure 4.2 : Type de mouvements réalisé dans la RT.

Différentes approches peuvent être utilisées pour déterminer le coût d'un tel mouvement. Idéalement, ce coût devrait être calculé en résolvant de façon optimale un PVC sur les tournées T_i et T_j après le déplacement du client x_{i_k}. Cependant, comme cette approche peut être très coûteuse en terme de temps de calcul, une méthode moins précise mais plus rapide est généralement utilisée. Ainsi, l'évaluation du coût d'un mouvement prend en compte d'une part, la variation des longueurs des tournées T_i et T_j sans que la séquence de visite des clients différents de x_{i_k} soit modifiée et, d'autre part, la modification des coûts de violation des contraintes relâchées.

Pour déterminer tout d'abord la variation des longueurs des tournées T_i et T_j, désignons par $\Delta D^+(j_p, u)$ l'accroissement de longueur occasionné sur T_j lorsque le client x_u $(1 \leq u \leq n)$ est inséré entre les clients x_{j_p} et $x_{j_{p+1}}$ $(0 \leq p \leq n_j)$. De façon similaire, $\Delta D^-(i_k)$ représente la diminution de la longueur de T_i lorsque l'on ôte le client x_{i_k} $(0 \leq k \leq n_i)$. Si l'on ne tient pas compte des clients fictifs, ces deux valeurs se calculent simplement comme suit :

- $\Delta D^+(j_p, u) = c_{j_p u} + c_{u j_{p+1}} - c_{j_p j_{p+1}} \geq 0,$
- $\Delta D^-(i_k) = c_{i_{k-1} i_{k+1}} - c_{i_{k-1} i_k} - c_{i_k i_{k+1}} \leq 0.$

Dans notre cas nous devons calculer différemment $\Delta D^+(j_p, u)$ et $\Delta D^-(i_k)$ lorsque x_{j_p} et/ou $x_{j_{p+1}}$ ainsi que $x_{i_{k-1}}$ et/ou $x_{i_{k+1}}$ sont des clients fictifs. Les trois situations spéciales impliquant des modifications sont illustrées dans le Tableau 4-2 où les ronds blancs et noirs représentent respectivement des clients normaux et fictifs. Le client devant être inséré ou ôté d'une tournée est indiqué par un cercle gris.

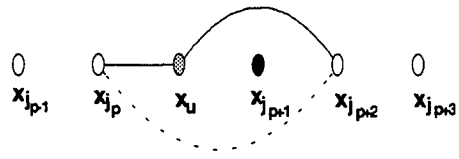
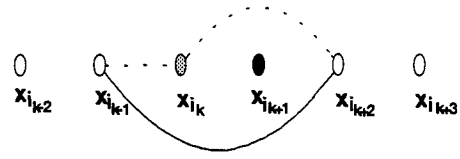
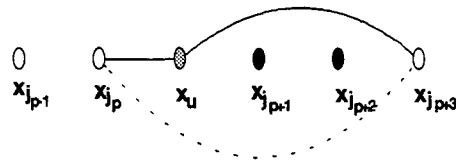
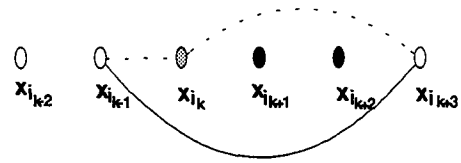
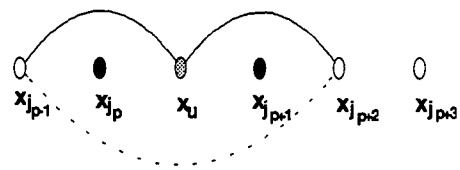
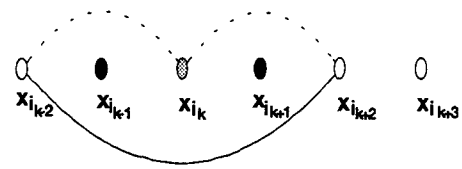
	Insérer x_u sur T_j entre x_{j_p} et $x_{j_{p+1}}$	Ôter x_{i_k} de T_i
Cas 1		
	$\Delta D^+(j_p, u) = c_{j_p u} + c_{u j_{p+2}} - c_{j_p j_{p+2}}$	$\Delta D^-(i_k) = c_{i_{k-1} i_{k+2}} - c_{i_{k-1} i_k} - c_{i_k i_{k+2}}$
Cas 2		
	$\Delta D^+(j_p, u) = c_{j_p u} + c_{u j_{p+3}} - c_{j_p j_{p+3}}$	$\Delta D^-(i_k) = c_{i_{k-1} i_{k+3}} - c_{i_{k-1} i_k} - c_{i_k i_{k+3}}$
Cas 3		
	$\Delta D^+(j_p, u) = c_{j_{p-1} u} + c_{u j_{p+2}} - c_{j_{p-1} j_{p+2}}$	$\Delta D^-(i_k) = c_{i_{k-2} i_{k+2}} - c_{i_{k-2} i_k} - c_{i_k i_{k+2}}$

Tableau 4-2 : Illustration des modifications à apporter à $\Delta D^+(j_p, u)$ et $\Delta D^-(i_k)$.

Notons que les autres cas spéciaux (i.e. lorsque seulement x_{j_p} ou seulement $x_{i_{k-1}}$ sont des clients fictifs) sont traités comme un cas normal car la distance entre un client fictif et un client normal est connue (voir section 4.2.2).

La variation $\delta(x_{i_k}, T_j)$ des longueurs des tournées T_i et T_j occasionnée par le mouvement $m(x_{i_k}, T_j)$ est donnée par la somme du plus petit accroissement de longueur de T_j et de la diminution de longueur de T_i :

$$\delta(x_{i_k}, T_j) = \min_{p=0,1,\ldots,n_j} \Delta D^+(j_p, i_k) + \Delta D^-(i_k)$$

Pour chaque tournée T_i, les coûts de violation des contraintes relâchées sont calculés à l'aide des formules suivantes :

- dépassement C_i de la charge utile autorisée :

$$C_i = \max(0, \sum_{k=1}^{n_i} q_{i_k} - Q_i)$$

- dépassement D_i de la durée autorisée d'une tournée :

$$D_i = \max(0, \sum_{k=0}^{n_i} t_{i_k i_{k+1}} + \sum_{k=1}^{n_i} (a_{i_k} + s_{i_k} + w_{i_k}) - D_{max})$$

- les violations FT_i, exprimées en minutes, des fenêtres de temps :

$$FT_i = \sum_{k=1}^{n_i} \left(\max(0, h_{i_k} - f^2_{i_k})\right)^2$$

où h_{i_k} et $f^2_{i_k}$ sont, respectivement, l'heure de début du service chez le client x_{i_k} et la limite supérieure de la seconde fenêtre de temps du client x_{i_k}.

Dès lors, le coût d'un mouvement se détermine de la façon suivante :

$$\delta(x_{i_k}, T_j) + p_1 \cdot (\delta C_i + \delta C_j) + p_2 \cdot (\delta D_i + \delta D_j) + p_3 \cdot (\delta FT_i + \delta FT_j)$$

où δC_i, δD_i, δFT_i and δC_j, δD_j, δFT_j sont, respectivement, les variations des coûts de violations dues au déplacement du client x_{i_k} de la tournée source T_i vers la tournée cible T_j et $p_r \in \mathbb{R}^+$, r = 1, 2 ou 3, sont trois paramètres de pénalité.

Une fois qu'un mouvement a été réalisé, une procédure de 2–optimalité (voir section 3.2) est exécutée sur les tournées T_i et T_j pour essayer de diminuer leur longueur tout en satisfaisant les contraintes de fenêtres de temps.

4.2.4.3 Voisinage d'une solution

Le voisinage N(s) d'une solution s est défini comme l'ensemble des solutions de S qui peuvent être atteintes depuis s par l'application d'un mouvement. Étant donné que le nombre total de tournées est m et le nombre total de clients est n, le cardinal du

voisinage à considérer à chaque itération est au plus de n·(m–1) puisque chaque client ne peut être déplacé que dans m–1 tournées au plus. De par la taille modérée du voisinage d'une solution dans notre problème, 1000 en moyenne, nous l'explorerons complètement à chaque itération. La détermination de la meilleure solution dans ce voisinage se fait de manière standard comme décrit dans la section 2.3.2.2.2.

4.2.4.4 Fonction objectif

Avec chaque solution nous associons la fonction F_f suivante qui représente la somme des longueurs des tournées :

$$F_f(s) = \sum_{i=1}^{m} \sum_{k=0}^{n_i} c_{i_k i_{k+1}}$$

Étant donné les définitions des coûts de violation des contraintes relâchées, la fonction objectif de notre problème est définie comme suit :

$$F(s) = F_f(s) + p_1 \cdot \sum_{i=1}^{m} C_i + p_2 \cdot \sum_{i=1}^{m} D_i + p_3 \cdot \sum_{i=1}^{m} FT_i$$

Si la solution s est admissible, les valeurs de $F_f(s)$ et F(s) sont identiques. Dans le cas contraire, F(s) contient trois termes de pénalité dus aux violations des contraintes relâchées.

Le choix de valeurs appropriées pour les paramètres de pénalité p_1, p_2 et p_3 est généralement un travail difficile et fastidieux. Pour réaliser cette tâche, nous avons opté pour une stratégie, proposée par Gendreau et al. [Gen94], permettant de faire varier les valeurs de ces paramètres durant le processus de recherche. Dans le cadre de notre problème, nous avons adapté cette stratégie de la façon suivante.

Les paramètres p_1, p_2 et p_3, associés respectivement aux violations des contraintes sur la charge utile, la durée maximale et les fenêtres de temps, sont initialisés avec des valeurs choisies aléatoirement dans la phase d'initialisation de notre approche Tabou. Ensuite, ces valeurs sont modifiées dynamiquement toutes les L_p itérations à l'aide d'un nombre aléatoire réel γ compris entre 1.50 et 2.00. Cette stratégie d'oscillation dynamique des valeurs des paramètres de pénalité conduit la recherche à visiter des

solutions aussi bien admissibles qu'inadmissibles. Toutes les L_p itérations la mise à jour du paramètre p_i, i = 1, 2 ou 3, se déroule comme suit :

- Si les L_p dernières solutions étaient admissibles par rapport à la contrainte associée à p_i alors p_i est fixé à p_i/γ. Si elles sont toutes inadmissibles, p_i est fixé à $\gamma \cdot p_i$. Lorsque des solutions admissibles et inadmissibles ont été visitées durant les L_p dernières itérations, la valeur de p_i reste inchangée.

4.2.4.5 Liste de tabous

Nous avons testé deux listes de tabous. La première mémorise les Θ derniers clients qui ont été déplacés alors que la seconde mémorise les Θ derniers mouvements réalisés. Par conséquent, si le client x_{i_k} est ôté de la tournée T_i à l'itération ℓ, le déplacement de x_{i_k} dans le premier cas ou l'insertion de x_{i_k} sur T_i dans le second, sont interdits jusqu'à l'itération $\ell+\Theta+1$. Pour modéliser la liste de tabous sur les mouvements, nous avons créé un tableau à deux entrées qui nous permet de savoir à tout moment et très rapidement si le mouvement consistant à insérer un client x_i sur une tournée T_k est tabou ou non :

	x_1	x_2	...	x_i	...	x_n
T_1						
...						
T_k				**élément (T_k, x_i)**		
...						
T_m						

Tableau 4-3: Liste de tabous sur les mouvements.

Au début du processus, ce tableau est initialisé avec des zéros. Lorsque, par exemple, un mouvement consistant à ôter le client x_i de la tournée T_k a été réalisé à l'itération ℓ, l'élément (T_k, x_i) du tableau est fixé à $\ell+\Theta+1$. Ainsi donc, l'élément (T_k, x_i) contient le numéro de l'itération à partir de laquelle le mouvement consistant à insérer le client x_i sur la tournée T_k ne sera plus tabou. Pour modéliser la liste tabou sur les clients, le même type de procédés peut être utilisé. Lorsque nous parlerons de liste de tabous dans la suite de ce texte, elle représentera aussi bien celle sur les clients que celle sur les mouvements.

La taille Θ de la liste de tabous se modifie dynamiquement au cours de notre algorithme. Toutes les L_Θ itérations, la valeur de Θ est générée aléatoirement, de façon uniforme, entre une borne inférieure Θ_{min} et une borne supérieure Θ_{max}. Cette stratégie peut se justifier de la manière suivante : afin d'être capable à la fois d'explorer intensivement les solutions voisines d'un optimum local et de s'en extraire lorsque son voisinage a été visité, il est judicieux de modifier la valeur de Θ au cours de la recherche. Nous bénéficions ainsi simultanément des avantages d'une petite et d'une grande valeur de Θ.

En plus de la procédure de modification décrite ci-dessus, la taille Θ de la liste de tabous est adaptée à chaque itération. Nous considérons pour cela $F(s_\ell)$ la valeur de la fonction objectif associée à la solution s_ℓ obtenue à l'itération ℓ et F_f^* la valeur de la fonction objectif associée avec la meilleure solution admissible rencontrée jusqu'à l'itération ℓ. La procédure d'adaptation de Θ est alors la suivante :

$$\begin{array}{|l|}\hline \text{Si } F(s_\ell) < F_f^* \\ \quad \text{alors Si } (F(s_\ell) < F(s_{\ell-1})) \text{ et } (\Theta > \Theta_{min}) \text{ alors } \Theta = \Theta - 1 \\ \quad \text{sinon Si } (F(s_\ell) > F(s_{\ell-1})) \text{ et } (\Theta < \Theta_{max}) \text{ alors } \Theta = \Theta + 1 \\ \hline \end{array}$$

L'utilisation d'un tel processus dynamique d'adaptation de Θ permet d'intensifier la recherche locale lorsqu'une amélioration de la solution courante se produit et aussi, lorsque la solution courante n'a pas été améliorée, de s'échapper plus rapidement d'un optimum local. De plus, lorsque la meilleure solution admissible connue est améliorée, la région proche de l'optimum local obtenu est explorée intensivement même si cela conduit à réexaminer des solutions précédemment visitées.

Lors de la détermination de la prochaine solution à visiter, nous utilisons la fonction d'aspiration communément employée (voir section 2.3.2.2.2) qui enlève le statut tabou d'un mouvement si celui-ci permet de se diriger vers une région prometteuse de l'espace des solutions. Par conséquent, si nous obtenons une solution telle que $F(s_\ell) < F_f^*$, nous acceptons de réaliser un mouvement dont le statut serait tabou.

4.2.4.6 Processus d'intensification

L'intensification est une composante additionnelle importante de la RT dont le but est d'accentuer l'exploration de certaines régions de l'espace des solutions identifiées comme prometteuses au cours de la recherche. Pour ce faire, et à la différence de ce que

nous faisons avec une liste de tabous, nous allons analyser le processus de recherche à plus long terme. Nous désirons par ce biais pouvoir déterminer si les bonnes solutions visitées jusque-là contiennent des caractéristiques communes, c'est-à-dire des structures qui apparaissent régulièrement. Si c'est le cas, nous modifierons alors le processus de recherche afin de favoriser la présence de telles structures dans les nouvelles solutions que l'algorithme visitera. Une telle modification est généralement appliquée périodiquement et pour une courte durée après laquelle le processus de recherche reprend normalement son cours. La plus simple des stratégies d'intensification consiste à retourner vers une des meilleures solutions visitées précédemment, puis à reprendre le processus de recherche à partir de celle-ci en diminuant la longueur de la liste de tabous pour un nombre restreint d'itérations. De cette façon, nous espérons explorer minutieusement le voisinage de cet optimum local.

Notre stratégie d'intensification a pour but de forcer le processus de recherche à examiner intensivement une région prometteuse de l'espace des solutions dans laquelle les solutions sont proches les unes des autres dans un certain sens, par exemple en possédant des séquences similaires de visite de clients. Dans le cadre de notre problème, nous définissons des solutions proches comme étant des solutions qui sont constituées de tournées identiques. Dès lors, notre processus d'intensification va consister à mettre un statut tabou sur certaines tournées de la meilleure solution admissible s* rencontrée jusque-là et, pendant L_{int} itérations, considérer uniquement des mouvements impliquant des clients situés sur des tournées dont le statut n'est pas tabou. Par conséquent, notre procédure d'intensification peut aussi être considérée comme une réduction astucieuse du voisinage de s*. Une approche similaire a été proposée par Taillard [Tai93a] pour la résolution du PTV classique à l'aide de méthodes de recherche itératives parallèles.

Pour déterminer quelles tournées de s* doivent recevoir un statut tabou, nous calculons le barycentre B_i de chaque tournée T_i de s*. B_i est défini comme le barycentre des clients x_{i_k} desservis sur T_i. Les coordonnées polaires d'un barycentre B_i sont données dans un repère dont l'origine est le dépôt. Ensuite, nous classons les tournées selon les valeurs croissantes des coordonnées angulaires des barycentres. Une fois ce classement établi, la RT est intensifiée sur une séquence de m_{int} tournées comprenant une tournée choisie aléatoirement et ses m_{int} - 1 successeurs dans le classement établi au préalable. Les autres tournées reçoivent un statut tabou pour les L_{int} prochaines itérations, i.e. aucun client ne peut être ôté ou inséré sur l'une de celles-ci pendant L_{int} itérations. Ainsi, la recherche n'examine intensivement qu'un sous-ensemble de clients

qui sont spatialement proches. De plus, nous permettons la création de nouvelles tournées au cours de ce processus d'intensification si des véhicules sont disponibles.

Si la solution s* est améliorée au cours des L_{int} itérations, le processus d'intensification est stoppé et le procédé global de recherche reprend. Dans le cas contraire, une nouvelle séquence de m_{int} tournées est examinée. Lorsque tous les sous-ensembles de m_{int} tournées ont été générés, m_{int} est augmenté de 2 unités et l'intensification se poursuit jusqu'à ce que soit la solution s* est améliorée, soit m_{int} est plus grand ou égal au nombre de tournées m* de s*. Il est bon de préciser que comme la topologie suisse renferme bon nombre d'obstacles naturels tels que les montagnes et les lacs, les séquences de tournées pour lesquelles leurs barycentres se trouve de part et d'autre de tels obstacles ne sont pas considérées (voir Figure 4.3).

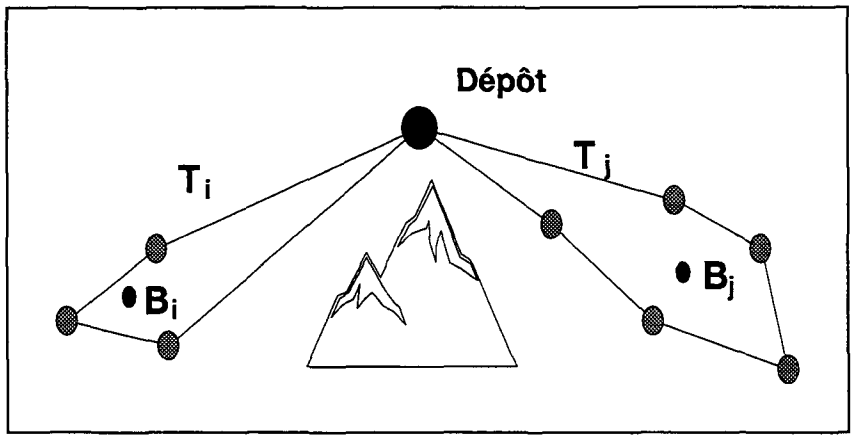


Figure 4.3 : Illustration du cas où les barycentres se trouvent de part et d'autre d'un obstacle naturel.

Pour chaque nouvelle séquence de tournées considérée, de nouvelles valeurs sont générées pour la taille Θ de la liste de tabous et les paramètres de pénalité p_1, p_2 et p_3. De surcroît, comme la taille de la liste de tabous doit être diminuée suite à la réduction de la taille du voisinage imposée par l'intensification, Θ sera générée aléatoirement de façon uniforme dans un nouvel intervalle défini par ses bornes Θ^{int}_{min} et Θ^{int}_{max}.

Finalement notons que, dans notre adaptation de la méthode Tabou, la stratégie d'intensification n'est utilisée que lorsque s* n'a pas été amélioré depuis L_{gen} itérations. La valeur L_{gen} représente donc le nombre d'itérations que la recherche consacre au problème dans son intégralité avant d'invoquer le processus d'intensification. Si nous désignons par ℓ^* l'itération à laquelle la meilleure solution admissible connue s* de valeur F_f^* à été trouvée, et par L_{max} le nombre total d'itérations alloué au processus de recherche, notre adaptation de la méta-heuristique Tabou se décrit comme suit :

(1) : Employer l'heuristique d'insertion rapide pour obtenir une solution initiale s_0 de valeur $F_f(s_0)$.

(2) : Initialiser $F_f^* = F_f(s_0)$, $s^* = s_0$, $\ell^* = 0$, $\ell = 0$, m_{int}, L_Θ, L_{gen}, L_{int}, L_p.
Générer les paramètres de pénalité p_1, p_2 et p_3 ainsi qu'une taille Θ de la liste de tabous entre Θ_{min} et Θ_{max}.

(3) : $\ell = \ell + 1$.

(Détermination du prochain mouvement à effectuer *)*

Déterminer le meilleur mouvement $m^*(x_{i_k}, T_j)$ qui n'est pas tabou ou celui qui satisfait les conditions d'aspiration. Obtenir la solution s_ℓ en déplaçant le client x_{i_k} de la tournée source T_i vers la tournée cible T_j.
Améliorer les tournées T_i et T_j avec une procédure de 2–optimalité.
Selon la liste de tabous considérée, interdire soit le déplacement de x_{i_k} soit l'insertion de x_{i_k} sur T_i pendant les Θ prochaines itérations.

(Mise à jour de la meilleure solution rencontrée *)*

Si $(F(s_\ell) = F_f(s_\ell))$ et $(F(s_\ell) < F_f^*)$ **alors**
Poser $F_f^* = F(s_\ell)$, $\ell^* = \ell$, $s^* = s_\ell$,
Fixer m_{int} à sa valeur initiale.

(Mise à jour des paramètres de pénalité et de Θ *)*

Si $\ell \bmod L_\Theta = 0$
alors Générer une nouvelle valeur de Θ entre Θ_{min} et Θ_{max}
sinon Adapter Θ selon le procédé décrit en section 4.2.4.5.

Si $\ell \bmod L_p = 0$
alors Mettre à jour les paramètres de pénalité p_1, p_2 et p_3.

(Processus d'intensification *)*

Si $(\ell - \ell^* \geq L_{gen})$ et $((\ell - (\ell^* + L_{gen})) \bmod L_{int}=0)$ et $(m_{int} < m^*)$ **alors**
Si tous les sous-ensembles de m_{int} tournées consécutives ont été examinés **alors** $m_{int} = m_{int} + 2$.

Si $m_{int} < m^*$ **alors**
Générer une nouvelle séquence de m_{int} tournées,
Générer les paramètres de pénalité p_1, p_2 et p_3 ainsi qu'une nouvelle valeur Θ entre Θ_{min}^{int} et Θ_{max}^{int}.

(4) : **Si** $\ell = L_{max}$
alors STOP
sinon Retourner en (3).

Tableau 4-4 : Adaptation de la méta-heuristique Tabou.

4.2.5 Les résultats numériques

Dans le cadre de notre étude, nous nous sommes intéressés à l'élaboration de plans de routes à partir des commandes réelles enregistrées pendant une semaine représentative de l'année. Par conséquent, notre analyse est basée sur un ensemble de clients et des commandes qu'ils ont passées chaque jour de cette semaine. Pour chacun de ces jours, la longueur totale des tournées effectivement réalisées par les chauffeurs de l'entreprise a été recalculée à l'aide de la même matrice des distances utilisée dans nos méthodes. Dès lors, nous pouvons aisément comparer les plans de routes suivis par la société et les solutions fournies par nos heuristiques.

Le tableau suivant décrit les plans de routes suivis par les chauffeurs de l'entreprise lors de la semaine considérée :

Jour de la semaine	Plans de transport de l'entreprise	
	Nombre de camions	Distance [km]
Lundi	8	1'554
Mardi	12	2'657
Mercredi	10	1'763
Jeudi	9	1'367
Vendredi	14	2'543

Tableau 4-5 : Plans de routes actuels de la société.

L'observation attentive des données associées à chacune des journées de la semaine représentative étudiée a mis en évidence que plusieurs commandes provenaient de clients localisés dans le même village. Nous avons donc, dans le double but de diminuer le temps de calcul et de réduire la taille du voisinage d'une solution, étudié également le comportement de nos heuristiques sur des problèmes où les commandes provenant de clients d'une même localité sont agrégées en un unique ordre de livraison si les fenêtres de temps de ces clients sont compatibles.

Nos observations nous ont conduit à utiliser les données originales pour le lundi, le mercredi et le jeudi et des commandes agrégées pour le mardi et le vendredi qui sont les deux jours de la semaine où les clients à desservir sont les plus nombreux. Dès lors, le nombre journalier de commandes varie entre 89 et 129 pour un poids moyen d'environ 150 tonnes. Les caractéristiques des différents problèmes testés sont résumées dans le Tableau 4-6.

Jour de la semaine	Poids total [kg]	Nombre total de commandes	
		Normales	Agrégées
Lundi	121'235	**105**	65
Mardi	161'423	129	**89**
Mercredi	157'853	**109**	76
Jeudi	128'240	**118**	72
Vendredi	202'461	170	**129**

Tableau 4-6 : Caractéristiques des problèmes étudiés.

Les expériences numériques réalisées ont été obtenues sur une station Silicon Graphics Indigo (200 Mhz) et les temps de calcul sont exprimés en secondes. Dans les sections suivantes, nous allons donner les valeurs des différents paramètres de nos heuristiques qui permettent d'obtenir les résultats présentés.

4.2.5.1 Méthode heuristique d'insertion

Pour utiliser cette méthode nous devons d'une part, fixer les paramètres conditionnant l'insertion du meilleur client non desservi sur les tournées en construction et d'autre part, déterminer la taille du voisinage N_n pour chacun des clients soumis aux contraintes d'accessibilité. Cette taille, fixée à 15, n'est utilisée que dans la première phase de notre heuristique où nous élaborons une solution partielle du problème qui considère uniquement les clients ayant des contraintes d'accessibilité. Les paramètres considérés pour la seconde phase de notre algorithme d'insertion sont ceux décrits dans la section 3.1.4. Les 176 combinaisons possibles de ceux-ci ont été testées pour chaque jour de la semaine et dans un second temps, les solutions obtenues ont été classées dans un ordre croissant en fonction de la longueur totale des tournées qui les composent. Nous avons donc pu déterminer :

- le meilleur ensemble de paramètres en moyenne, c'est-à-dire celui fournissant la meilleure moyenne des longueurs totales des cinq plans de transport élaborés. C'est le type *BestMoy*.
- trois jeux de paramètres pour chaque journée qui correspondent à la moins bonne solution, à la solution moyenne et à la meilleure solution obtenue pour le jour considéré. Ce sont respectivement les types de paramètres *Moins Bon*, *Moyen* et *Meilleur*.

Cela donne donc quatre jeux de paramètres distincts pour chaque jour de la semaine. Ces derniers, présentés dans le Tableau 4-7, ont été utilisés pour générer les solutions initiales admissibles de notre algorithme Tabou.

Jour de la semaine	Type de paramètres	Jeu de paramètres					Solution obtenue		
		Critère d'initialisation	α_1	α_2	λ	μ	Camions	Distance [km]	Temps CPU
Lundi	Moins bon	Farthest	0.4	0.6	0.0	1.0	10	2'245	0.49
	Moyen	Farthest	0.0	1.0	1.5	1.0	8	1'802	0.43
	Meilleur	Earliest	0.4	0.6	0.5	1.0	8	1'550	0.77
	BestMoy	Farthest	0.7	0.3	1.5	1.0	8	1'668	0.36
Mardi	Moins bon	Earliest	0.2	0.8	0.0	1.0	14	4'136	0.46
	Moyen	Earliest	1.0	0.0	1.5	0.5	12	3'103	0.25
	Meilleur	Farthest	0.9	0.1	2.0	1.0	12	2'870	0.24
	BestMoy	Farthest	0.7	0.3	1.5	1.0	12	2'886	0.23
Mercredi	Moins bon	Earliest	0.0	1.0	1.0	1.0	12	3'131	0.24
	Moyen	Earliest	0.7	0.3	0.5	1.0	10	2'203	0.34
	Meilleur	Farthest	1.0	0.0	1.0	1.0	10	1'936	0.27
	BestMoy	Farthest	0.7	0.3	1.5	1.0	10	2'006	0.25
Jeudi	Moins bon	Farthest	0.1	0.9	0.0	1.0	9	1'712	0.77
	Moyen	Earliest	0.4	0.6	1.75	0.75	9	1'398	0.56
	Meilleur	Farthest	0.7	0.3	2.0	1.0	8	1'226	0.53
	BestMoy	Farthest	0.7	0.3	1.5	1.0	8	1'290	0.54
Vendredi	Moins bon	Earliest	0.9	0.1	1.0	1.0	14	3'291	0.63
	Moyen	Earliest	0.3	0.7	2.0	1.0	14	3'036	0.42
	Meilleur	Farthest	1.0	0.0	1.5	0.5	14	2'639	0.54
	BestMoy	Farthest	0.7	0.3	1.5	1.0	14	2'797	0.49

Tableau 4-7 : Solutions obtenues avec l'heuristique rapide d'insertion.

Les variations entre les différents jeux de paramètres retenus peuvent s'expliquer par le fait que l'ensemble des clients à desservir chaque jour est différent. De surcroît, la contrainte sur les fenêtres de temps est plus ou moins restrictive au sein de chacun des cinq problèmes étudiés.

4.2.5.2 Méta-heuristique Tabou

Nous devons à présent fixer les différents paramètres de notre algorithme de recherche avec tabous. Dans un premier temps, nous avons observé le comportement de la recherche pour différentes valeurs de la taille Θ de la liste de tabous lorsque cette taille

ne varie pas au cours de l'exécution de l'algorithme. La Figure 4.4 montre la variation de la distance moyenne parcourue sur la semaine en fonction de la taille de la liste de tabous.

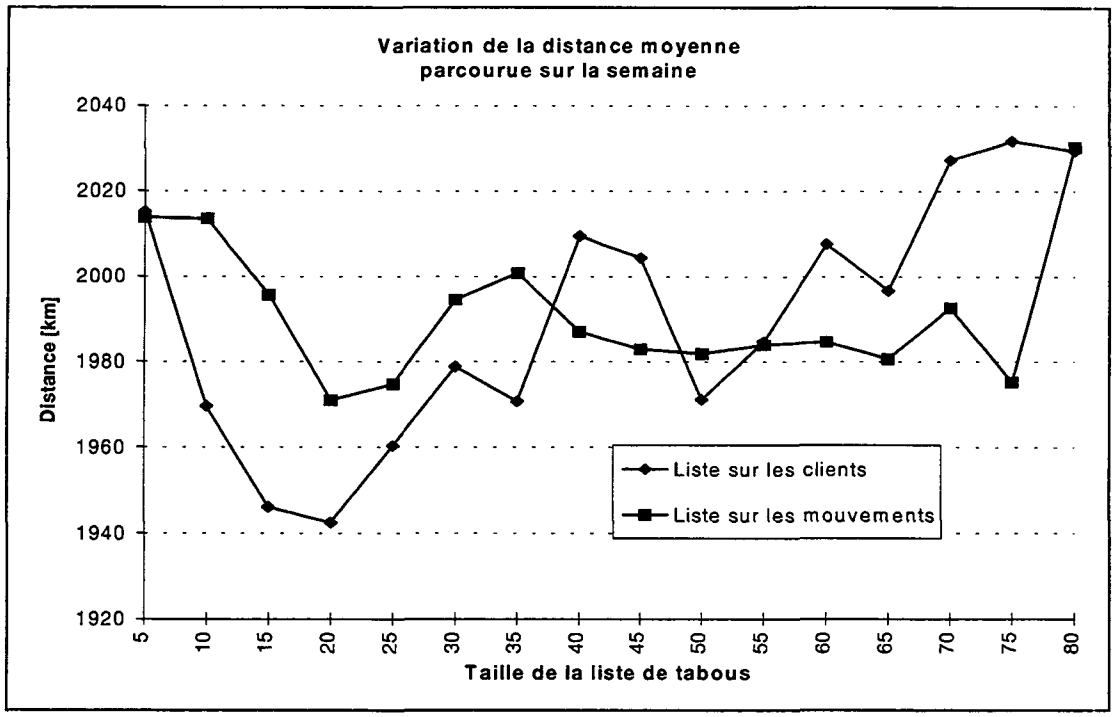


Figure 4.4 : Comportement de la recherche en fonction de la taille de la liste de tabous.

Sur la base de cette figure, nous avons retenu plusieurs intervalles de variation possible de la taille de la liste de tabous, à savoir [15,35], [15,25] et [10,55] pour la liste de tabous sur les clients ainsi que [15,70], [15,35] et [35,70] pour la liste de tabous sur les mouvements. Dans un second temps, nous nous sommes intéressés à faire varier les valeurs de L_{gen} et de L_{int} lorsque la taille de la liste de tabous variait dynamiquement à l'intérieur des intervalles précités. Sur la base de cette étude nous avons décidé de retenir une liste de tabous sur les clients dont la taille Θ varie dynamiquement entre $\Theta_{min} = 10$ et $\Theta_{max} = 55$. Lorsqu'aucune amélioration de la meilleure solution admissible connue n'a été réalisée après $L_{gen} = 1'500$ itérations, le processus d'intensification est appliqué. Initialement, des séquences de $m_{int} = 4$ tournées consécutives sont examinées pour $L_{int} = 750$ itérations. En raison de la diminution de la cardinalité du voisinage, la taille de la liste de tabous varie entre $\Theta^{int}_{min} = 3$ et $\Theta^{int}_{max} = 18$ au cours de l'intensification de la recherche. La longueur L_p du cycle de mise à jour dynamique des paramètres de pénalité a été fixée à 10 itérations lorsque les commandes ne sont pas agrégées et à 20 itérations dans le cas contraire et finalement, le nombre maximum d'itérations L_{max} autorisées pour la recherche est de 120'000.

Pour ces valeurs de paramètres, les résultats obtenus par notre RT pour chacune des journées de la semaine sont présentés dans le Tableau 4-8. Afin de tester la robustesse de notre approche, chaque problème a été résolu quatre fois en prenant comme solutions initiales admissibles de la RT les solutions obtenues avec les quatre types de paramètres de l'heuristique rapide d'insertion (voir Tableau 4-7). Pour chaque journée, nous indiquons les pourcentages de variation des longueurs totales des tournées calculés par rapport aux longueurs totales des tournées réalisées effectivement par les véhicules de la société.

Dans les premières colonnes du Tableau 4-8 nous indiquons les pourcentages calculés pour les meilleures solutions construites avec l'heuristique d'insertion ainsi que pour celles élaborées avec ce même algorithme mais avec le type de paramètre BestMoy. En ce qui concerne la RT, nous reportons les pourcentages de variation sur les moyennes des valeurs des solutions et les temps de calcul moyens observés après 30'000 et 120'000 itérations. De plus, nous donnons les pourcentages d'écart avec la meilleure solution rencontrée (lors des expériences numériques menées pour fixer les différentes valeurs de nos paramètres) et le temps de calcul nécessaire pour l'obtenir.

Il est important de relever que cette meilleure solution n'est pas forcément obtenue à partir de la meilleure solution fournie par l'heuristique d'insertion.

Jour de la semaine	Méthode d'insertion		Méthode de RT					
	Type de paramètres		*Solution moyenne*				*Meilleure solution*	
	BestMoy	Meilleur	30'000 itérations		120'000 itérations		120'000 itérations	
Lundi	7.3	-0.3	-7.9	656	-8.3	2'769	-10.5	2'793
Mardi	8.6	8.0	1.8	406	0.4	1'634	-0.4	1'814
Mercredi	13.8	9.8	0.1	600	-0.1	3'175	-0.9	605
Jeudi	-5.6	-10.3	-17.6	814	-18.7	3'335	-22.2	3'168
Vendredi	10.0	3.8	-2.6	821	-4.1	2'660	-5.4	2'462
Moyenne:	6.8	2.2	-5.2	659	-6.2	2'715	-7.7	2'168

Tableau 4-8 : Pourcentages de variation sur les longueurs totales des tournées et temps de calcul.

Dans le Tableau 4-9, nous reportons le nombre de camions utilisés dans les résultats présentés dans le Tableau précédent.

Jour de la semaine	Solution actuelle	Méthode d'insertion		Méthode de RT		
		Type de paramètres		*Solution moyenne*		*Meilleure solution*
		BestMoy	Meilleur	30'000 itér.	120'000 itér.	120'000 itér.
Lundi	8	8	8	8.0	8.0	8
Mardi	12	12	12	12.0	11.75	11
Mercredi	10	10	10	10.0	10.0	10
Jeudi	9	8	8	8.0	8.0	8
Vendredi	14	14	14	13.0	13.0	13

Tableau 4-9 : Nombre de camions utilisés.

Il est important de relever que la plupart des plans de transport de la société ne respectent pas les contraintes prises en compte dans notre formulation. Ces violations dénotent les difficultés rencontrées par les responsables de l'entreprise pour élaborer des tournées de véhicules admissibles. Ces difficultés sont d'ailleurs une des raisons qui ont motivé notre étude. Les violations totales pour chaque jour de la semaine considérée sont résumées dans le Tableau 4-10.

Jour de la semaine	Violation totale sur la durée des tournées	Violation totale sur les fenêtres de temps	Surcharge totale des véhicules [kg]
Lundi	3h33	2h45	795
Mardi	6h34	1h41	—
Mercredi	2h28	2h10	1'810
Jeudi	0h40	0h51	2'840
Vendredi	4h39	3h21	345

Tableau 4-10 : Violations des contraintes pour les solutions actuelles de la société.

Les résultats que nous avons obtenus sont très encourageants. En effet, les pourcentages indiqués dans le Tableau 4-8 sont déterminés sur la base des longueurs totales des plans de transport de la société qui ne sont pas admissibles. De surcroît, même si la distance totale parcourue durant la semaine est augmentée de 6.8% en moyenne pour les solutions obtenues avec la méthode d'initialisation (2.2% pour les meilleures solutions), celle-ci diminue de 6.2% en moyenne en utilisant la méta-heuristique Tabou. La RT domine très nettement la procédure d'initialisation, même lorsque le nombre total d'itérations autorisées est réduit. Ainsi, en limitant la RT à 30'000 itérations nous constatons une diminution moyenne de 5.2% de la distance totale parcourue lors des tournées effectuées au cours de la semaine.

Concernant le nombre de véhicules utilisés, les solutions établies à l'aide des deux méthodes sont très semblables. En effet, tous les plans de tournées calculés nécessitent l'utilisation d'un nombre de camions identique à celui effectivement mis en oeuvre par la société, à l'exception des données pour le jeudi, le vendredi et parfois le mardi. Pour la journée du mardi, la RT permet parfois d'obtenir une solution avec un camion de moins lorsque nous laissons évoluer le processus de recherche assez longtemps. Pour le vendredi, seul la RT permet d'obtenir une solution utilisant un véhicule de moins alors que pour le jeudi, nos deux méthodes proposent des plans de routes comportant un véhicule de moins.

En utilisant le type de paramètres BestMoy, l'heuristique d'insertion n'améliore les plans de transport effectués dans la réalité que pour la journée du jeudi. Il faut souligner que le jeudi est précisément le seul jour de la semaine où le plan de transport de la société satisfait presque les contraintes sur les fenêtres de temps et sur la durée des tournées. Pour tous les autres problèmes, cette procédure d'initialisation donne des solutions admissibles dont les distances sont environ 10% plus longues que celles des solutions actuelles de l'entreprise. Cependant, l'intérêt pour cette méthode reste entier puisqu'elle fournit des plans de transport admissibles en seulement quelques secondes de temps de calcul.

Toutes les solutions obtenues à l'aide de l'algorithme d'initialisation sont significativement améliorées par notre adaptation de la méta-heuristique Tabou. Nos résultats peuvent être classifiés en deux groupes. Pour le premier, qui contient les PTVFT survenant le lundi, le jeudi et le vendredi, notre RT fournit de très bonnes solutions qui requièrent presque à chaque fois un véhicule de moins. Pour le second groupe, constitué des PTVFT du mardi et du mercredi, la performance de notre RT est assez modeste. Ceci peut s'expliquer par le fait que les violations de contraintes enregistrées pour ces deux journées sont particulièrement importantes. De surcroît, les poids moyens par commande pour ces deux jours sont supérieurs à ceux des autres jours de la semaine. Dans une telle situation, la contrainte sur la charge utile des camions joue un rôle essentiel et peut entraver la découverte de solutions admissibles nettement meilleures que celles, non admissibles, de la société.

4.2.6 Conclusion

Dans la première partie de ce chapitre, nous avons présenté deux méthodes heuristiques pour la résolution d'un PTVFT réel auquel l'un des principaux producteurs suisses de farine et de nourriture pour bétail doit faire face. La première méthode est une simple procédure d'insertion dont les performances sont modestes. Cependant, comme son temps d'exécution n'est que de quelques secondes, nous avons la possibilité de l'invoquer plusieurs fois avec des jeux de paramètres différents afin d'améliorer la qualité des solutions proposées. Cela permet aussi de fournir une aide appréciable au responsable des transports car il peut ainsi avoir à disposition plusieurs plans de route et choisir l'un d'entre eux selon ses propres critères.

La seconde méthode est une adaptation de la méta-heuristique Tabou. Les points essentiels de notre approche sont la relaxation des contraintes et l'utilisation d'une stratégie d'intensification. Le premier point nous permet d'étendre l'espace des solutions en diversifiant la recherche puisque celle-ci peut examiner aussi bien des solutions admissibles que des solutions inadmissibles. Le processus d'intensification joue un rôle complémentaire. En effet, il conduit la recherche à visiter des solutions proches de la meilleure solution rencontrée en mettant un statut tabou sur certaines de ses tournées. Les expériences numériques effectuées sur des données représentatives de l'activité de la société nous permettent de conclure à la supériorité de notre adaptation de la méta-heuristique Tabou par rapport à l'heuristique d'insertion, même lorsque le nombre total d'itérations de cette heuristique est faible. Nous pouvons donc obtenir de très bonnes solutions qui respectent l'ensemble des contraintes en un temps de calcul très raisonnable (11 et 45 minutes en moyenne pour effectuer respectivement 30'000 et 120'000 itérations de RT).

5. Diversification et intensification probabilistes des méthodes de recherche locale pour le PTV

5.1 Introduction

De plus en plus fréquemment, les méthodes de recherche locale sont utilisées afin de déterminer de bonnes solutions pour des problèmes d'optimisation combinatoire. Tout au long de ce chapitre, nous utiliserons le terme de "recherche locale" comme synonyme de recherche par évaluation de voisinages. Les recherches locales sont parfois restreintes aux méthodes de descente (voir section 2.3.2.1.1) mais nous inclurons des procédures plus générales comme le recuit simulé (voir section 2.3.2.2.1) ou la méta-heuristique Tabou (voir section 2.3.2.2.2) dans cette classification. Ces techniques ont en commun les deux défauts majeurs suivants.

1. Elles peuvent être bloquées dans un optimum local de très mauvaise qualité. Cette situation peut être difficile à surmonter même en procédant à un fastidieux ajustage des paramètres de la recherche locale.
2. Elles requièrent généralement des temps de calcul importants. De nos jours, la voie la plus prometteuse pour réduire ces temps de calcul est l'utilisation des ordinateurs parallèles. Malheureusement, la recherche locale est un processus intrinsèquement séquentiel qui ne se parallélise que difficilement.

Dans ce chapitre, nous proposons une nouvelle méthode qui surmonte ces deux difficultés. Nous illustrerons cette technique sur deux méthodes de recherche avec tabous (RT) que nous avons développées pour l'élaboration de tournées de véhicules : d'une part pour le PTV élémentaire et d'autre part, pour le PTV avec contraintes de fenêtres de temps. Soulignons cependant que cette technique peut être appliquée à d'autres types de recherche locale ou d'autres variantes du PTV. Dans la section 5.2, nous décrivons brièvement les problèmes étudiés ainsi que les recherches locales utilisées pour les

résoudre. Puis, dans la section 5.3, nous présentons la nouvelle technique que nous avons développée. Cette méthode, située à la croisée entre la RT et les algorithmes génétiques, nous a permis dans un premier temps, de diversifier la recherche en visitant des solutions qui sont très différentes les unes des autres et dans un second temps, d'intensifier la recherche dans le but de trouver de meilleurs optima locaux dans une région prometteuse de l'espace des solutions admissibles. Nous montrons aussi comment il est possible de paralléliser notre approche. Nous décrivons ensuite une technique de post-optimisation qui conduit à des améliorations fréquentes des solutions obtenues avec le processus de diversification et d'intensification. Dans la section 5.4, nous comparons notre nouvelle technique aux recherches locales sur lesquelles elle est basée pour finalement convenir en section 5.5 de quelques conclusions.

5.2 PTV et recherche locale

5.2.1 Présentation des problèmes

Le premier problème étudié, décrit dans la section 1.2, est le PTV élémentaire : un groupe $X = \{x_1, ..., x_i, ..., x_n\}$ de n clients avec des demandes connues q_i ($i = 1, ...,n$) doit être desservi par une flotte homogène de véhicules de capacité fixe Q à partir d'un unique centre de distribution ou dépôt (x_0). Sur la base des distances c_{ij} séparant chaque paire (i,j) de clients ($0 \leq i, j \leq n$), l'objectif est de déterminer une séquence de livraison pour chaque véhicule de sorte que :

- la distance totale parcourue par la flotte de véhicules soit la plus petite possible,
- chaque client soit desservi une seule fois et par un seul véhicule,
- la capacité maximale Q de chaque véhicule ne soit pas dépassée.

Le second problème est le PTVFT académique, décrit dans la section 1.3, pour lequel Solomon [Sol87] a proposé un ensemble de 56 problèmes de référence différents.

5.2.2 Recherche locale pour le PTV

De manière très générale, une recherche locale peut se formuler comme suit :

1. Choisir une solution initiale $s_0 \in S$; poser k:= 0 ;
2. **Tant que** (un critère d'arrêt n'est pas satisfait) **faire**
   a. Choisir une solution $s_{k+1} \in N(s_k)$, les solutions voisines de s_k.
   b. k := k + 1.

Tableau 5-1 : Une méthode générale de recherche locale.

En fonction des choix réalisés lors de ces diverses étapes, nous obtenons des méthodes de recherche itérative différentes. Généralement, la solution initiale est choisie de sorte que son élaboration soit simple et rapide. Le choix du critère d'arrêt est souvent lié au type de recherches locales retenu : par exemple, une méthode de descente s'arrête dès qu'il n'y a pas de meilleure solution que s_k dans $N(s_k)$ alors qu'une RT est généralement stoppée lorsque k est supérieur à un certain nombre d'itérations. Sur la base de la formulation adoptée ici, les points les plus délicats pour la mise au point d'une recherche locale efficace sont la définition du voisinage d'une solution et le choix d'une solution dans celui-ci. Pour obtenir plus de détails quant à ces choix pour le problème de l'élaboration de tournées de véhicules, le lecteur intéressé se référera aux travaux de Taillard [Tai93a] et Gendreau et al. [Gen94, Gen95].

La méthode de Taillard [Tai93a] est l'une des plus efficaces pour le PTV élémentaire. Sa caractéristique principale est de partitionner les problèmes en sous-problèmes et d'optimiser chacun de ceux-ci indépendamment. Une partition est généralement constituée de sous-problèmes composés de 4 à 8 tournées (ou 30 à 60 clients) qui sont spatialement proches les unes des autres. Une fois que les sous-problèmes ont été optimisés, les tournées qui les constituent sont regroupées pour former une solution du problème original et le processus de partitionnement est réitéré à partir de cette nouvelle solution. Comme ce processus contient une composante aléatoire, l'algorithme peut fournir des solutions très différentes d'une exécution à l'autre. La RT que nous avons utilisée est légèrement différente de la méthode originale de Taillard par le fait que le processus de partitionnement a été amélioré et que la procédure exacte d'optimisation des tournées a été remplacée par une heuristique. La plupart du temps, ces modifications améliorent le comportement de la méthode lorsqu'elle est appliquée à des problèmes où les clients ne sont par répartis uniformément dans le plan autour du dépôt.

La RT que nous avons utilisée pour le PTVFT académique est dérivée de l'adaptation de Rochat et Semet [Roc94] pour un PTVFT réel complexe (voir section 4.2). C'est donc une simplification d'une RT sophistiquée plutôt qu'une nouvelle RT développée spécialement pour les problèmes académiques de Solomon [Sol87]. Malgré cela, la nouvelle technique que nous proposons est efficace puisqu'elle atteint ou améliore 47 des meilleures solutions publiées pour les 56 problèmes de Solomon. Parmi celles-ci, 16 nouvelles meilleures solutions sont proposées. De manière similaire à la RT utilisée pour le PTV élémentaire, notre RT pour le PTVFT académique comporte une composante aléatoire. Par conséquent, deux exécutions de l'algorithme fourniront généralement deux solutions différentes.

5.3 Amélioration des méthodes de recherche locale appliquées au PTV

5.3.1 Une technique probabiliste de diversification et d'intensification

Un principe fondamental de la RT est d'exploiter l'effet combiné de la diversification (processus qui a pour but d'amener la recherche à visiter des régions inexplorées de l'espace des solutions) et de l'intensification (phénomène opposé dont le but est d'intensifier l'exploration de certaines régions de l'espace des solutions préalablement identifiées comme prometteuses). Pour exploiter cet effet, notre approche est basée sur deux concepts fondamentaux que nous décrivons dans les paragraphes suivants.

Le premier de ces deux concepts provient de la RT probabiliste. Ce type de méthodes est fondé sur l'idée consistant à traduire l'information récoltée au cours de la recherche (sur les solutions visitées, leur valeur ainsi que les régions de l'espace dans lesquelles elles se situent) en termes d'évaluation auxquels des probabilités de sélection sont assignées. Dans le cadre de l'exploration du voisinage d'une solution, ce type de méthodes fera un choix parmi un ensemble d'alternatives possibles en fonction des probabilités de sélection qui auront été associées à ces alternatives. Les valeurs de ces probabilités sont déterminées de façon à favoriser le choix d'une alternative de bonne qualité. Cette approche, proposée par Glover [Glo90], est motivée par la prémisse suivante :

> une utilisation intelligente de la randomisation (lorsque celle-ci n'est pas simplement réalisée de manière uniforme mais par le biais de probabilités tenant compte de l'historique de la recherche et de la qualité des solutions visitées) est un type de diversification très utile qui remplace avantageusement d'autres techniques de diversification plus complexes.

Les gains d'efficacité procurés par cette utilisation intelligente de la randomisation proviennent du fait que cette approche est très simple à mettre en oeuvre et qu'il n'est pas nécessaire, comme pour les autres techniques de diversification, de mémoriser et d'évaluer une grande quantité d'information au cours de la recherche.

Le second concept est dérivé d'une des premières et des plus fondamentales stratégies d'intensification. Le coeur de cette stratégie réside dans la détermination de solutions par référence aux notions de variables *fortement déterminées* et *consistantes*. Une variable fortement déterminée est une variable dont la valeur ne peut pas être modifiée sans provoquer des modifications importantes de la valeur de la fonction objectif ou des valeurs d'autres variables du problème considéré. L'identification des variables fortement déterminées se réalise par référence aux meilleures solutions rencontrées préalablement au cours de la recherche. Nous mesurons donc les forces de détermination des variables d'un problème par rapport à la qualité des solutions que l'on obtient lorsqu'on leur fixe certaines valeurs. Ce procédé correspond à la pratique précitée de mise en évidence des bonnes alternatives dans la RT probabiliste, ce qui nous permet d'exploiter aisément ces deux approches simultanément.

Une variable consistante est une variable qui est fréquemment qualifiée de fortement déterminée pour des valeurs comprises dans un intervalle très précis. En d'autres termes, plus une variable reçoit fréquemment une valeur particulière dans les meilleures solutions d'un problème donné (lorsque ces solutions sont évaluées par rapport à leur valeur de la fonction objectif), plus elle est qualifiée de consistante. Le raisonnement pour identifier de telles variables et la stratégie pour les exploiter sont les suivants. Premièrement, une variable très consistante sera presque sûrement fixée à sa valeur préférée dans la solution optimale ou dans de très bonnes solutions. Deuxièmement, une fois que certaines variables ont été fixées à des valeurs particulières nous observons que d'autres variables qui n'étaient pas considérées précédemment comme consistantes peuvent le devenir. Finalement, le fait d'imposer des restrictions sévères sur les valeurs que peuvent prendre certaines variables permet de mesurer avec beaucoup plus de précision la consistance relative des autres variables. La stratégie d'intensification basée sur ces dernières remarques, proposée par Glover [Glo77], peut se résumer comme suit :

1. Sélectionner une ou plusieurs variables parmi celles qui sont les plus consistantes et les fixer à leur valeur préférée.
2. Déterminer les nouvelles consistances relatives des variables sur la base des restrictions imposées au point 1.
3. Répéter les étapes 1 et 2 jusqu'à ce que toutes les variables aient été fixées à une certaine valeur.

Ce processus est ensuite couplé avec une procédure heuristique d'amélioration pour transformer cette affectation de valeurs aux variables en une nouvelle solution du problème étudié. Nous obtenons ainsi une méthode itérative qui nous permettra d'obtenir des solutions dont la qualité s'améliorera progressivement.

Pour appliquer les deux concepts exposés ci-dessus dans le cadre de l'élaboration de tournées de véhicules, nous considérons un ensemble $\mathcal{T}$ constitué de tournées extraites de certaines des solutions visitées au cours de la recherche. Notons que nous n'interdisons pas la présence de tournées identiques dans cet ensemble. La première étape de la stratégie proposée par Glover est réalisée en sélectionnant les variables par blocs. Pour notre problème, cela consiste à choisir (selon un critère à définir) une tournée $t \in \mathcal{T}$. La détermination des nouvelles consistances relatives des variables (point 2 dans ce qui précède) est effectuée en tenant compte du fait que les valeurs des variables d'une même solution ne peuvent pas être modifiées une fois qu'elles ont été fixées. Dans notre cas, cela signifie que nous ne pouvons pas sélectionner une tournée $t \in \mathcal{T}$ si elle contient des clients déjà desservis dans d'autres tournées préalablement retenues. Les heuristiques d'amélioration qui sont les moteurs de notre technique de diversification et d'intensification sont celles de Taillard [Tai93a] et de Rochat et Semet [Roc94].

Une méthode de recherche locale doit être capable de fournir des solutions qui sont très différentes les unes des autres mais pas nécessairement de très bonne qualité pour prétendre jouer un rôle de diversification dans l'exploration de l'espace des solutions. Cette condition est généralement remplie par les recherches locales qui sont initialisées avec une solution générée au hasard ou qui contiennent une composante aléatoire dans leur processus de recherche. Dans ce cas, différentes exécutions de ces méthodes fourniront généralement des solutions différentes. Cependant, en incorporant le principe de la RT probabiliste pour guider les choix à réaliser dans notre méthode, nous avons créé un type de diversification plus stratégique qu'une simple randomisation. Nous indiquons précisément dans les paragraphes suivants les détails de notre approche.

Dans une première phase (l'initialisation), la recherche est diversifiée en générant, avec l'aide de la recherche locale, I solutions initiales qui sont différentes les unes des autres. Pour le PTV élémentaire, cette génération est réalisée en considérant diverses décompositions initiales du problème traité. Pour le PTVFT académique, ainsi que pour des PTV élémentaires de petite taille qui ne peuvent pas être décomposés, les caractéristiques non déterministes de notre recherche locale favorisent la diversité des solutions produites.

En générant plusieurs solutions initiales avec la recherche locale, nous espérons que toute l'information nécessaire à la création de solutions de très bonne qualité se trouve présente dans les solutions initiales même si c'est sous une forme non apparente.

Dans le cadre de l'élaboration de tournées de véhicules, cette information est contenue dans les tournées. Nous tablons dès lors sur le fait que la phase d'initialisation de notre méthode crée un ensemble de tournées dont certaines ne sont pas trop différentes de celles qui forment une bonne solution. Le défi est donc de trouver un ensemble de tournées qui sont simultanément de bonne qualité pour la totalité des clients à desservir.

La génération de I solutions initiales crée notre ensemble $\mathcal{T}$ de tournées. La Figure 5.1 représente un sous-ensemble de $\mathcal{T}$ après l'exécution de la phase d'initialisation pour un problème de Christofides et al. [Chr79] avec 199 clients. Cette figure est à comparer avec la Figure 5.2 qui montre une très bonne solution pour le même problème. Dans ces deux figures, les conventions suivantes ont été retenues :

- les cercles vides représentent les clients à desservir et la surface d'un cercle est proportionnelle à la quantité commandée par le client qu'il représente,
- le cercle noir est le dépôt et sa surface est proportionnelle à la charge utile d'un camion,
- le premier et le dernier trajet de chaque véhicule ne sont pas dessinés pour améliorer la lisibilité.

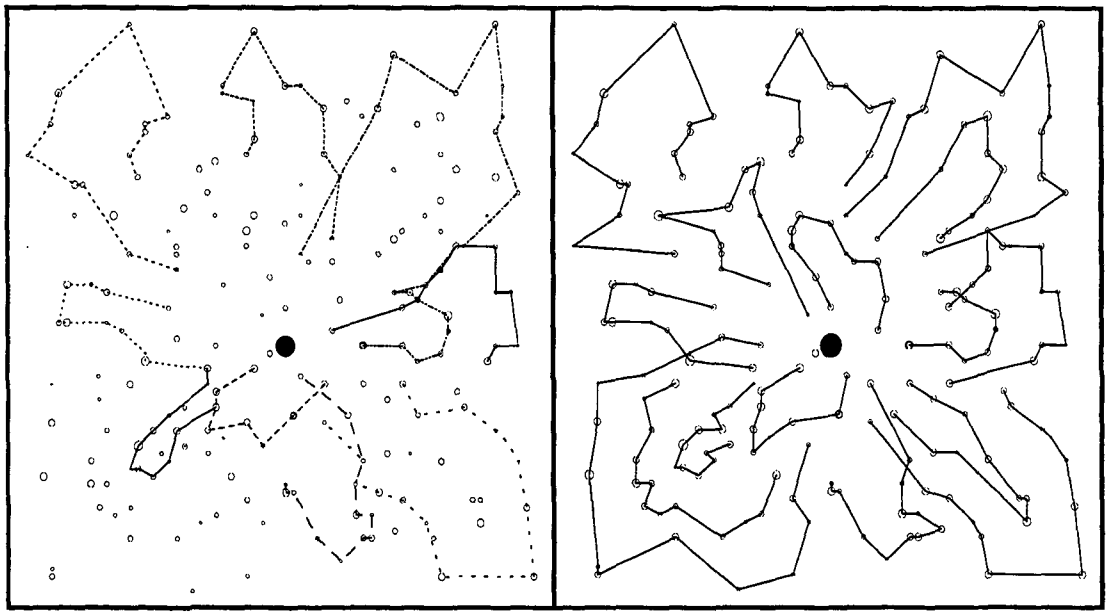


Figure 5.1: Un sous-ensemble de tournées appartenant à $\mathcal{T}$ après la phase d'initialisation.

Figure 5.2: La meilleure solution connue pour le problème de Christofides et al. avec 199 clients.

En observant attentivement ces deux figures, nous remarquons que plusieurs tournées sont similaires et deux d'entre elles sont identiques. Cependant, l'ensemble $\mathcal{T}$ contient plusieurs autres tournées si bien que les bonnes tournées ne peuvent pas être directement identifiées après la phase d'initialisation.

Dans une seconde phase, notre but est d'extraire puis d'améliorer les bonnes tournées qui se trouvent dans $\mathcal{T}$. Nous sommes partis du principe que si une solution quelconque contient de telles tournées, alors la valeur de sa fonction objectif est probablement meilleure que celle d'une autre solution qui ne les contient pas. Par conséquent, lors de cette seconde phase, nous allons favoriser l'extraction de tournées qui appartiennent aux meilleures solutions générées au cours de la recherche sans toutefois totalement exclure le choix de tournées provenant de mauvaises solutions. Pour mettre en œuvre cette seconde phase, chaque tournée reçoit une marque correspondant à la valeur de la solution à laquelle elle appartient. Comme la fonction objectif de notre problème s'exprime en termes de minimisation, une bonne tournée est une tournée dont la valeur de la marque associée est faible. Les tournées de l'ensemble $\mathcal{T}$ sont ensuite triées par ordre croissant des valeurs de leur marque afin d'obtenir un classement de la meilleure à la plus mauvaise des tournées. Lors de ce tri, les tournées composées d'un seul client sont ôtées de $\mathcal{T}$ car elles ne contiennent pas de caractéristiques intéressantes sur la structure du problème étudié.

Puis, nous construisons un ensemble $\mathcal{K}$ de tournées en choisissant de manière probabiliste des tournées dans $\mathcal{T}$. Ce choix est réalisé en donnant la préférence à des tournées dont la marque associée est de faible valeur et qui ne contiennent pas des clients déjà desservis par des tournées préalablement choisies. Cette sélection est réitérée jusqu'à ce qu'il ne soit plus possible d'extraire de nouvelles tournées de l'ensemble $\mathcal{T}$.

Comme il est fort possible que l'ensemble $\mathcal{K}$ ainsi construit ne contienne pas la totalité des clients du problème, $\mathcal{K}$ est une solution partielle qu'il s'agit de rendre admissible. Dans ce but, les clients qui n'appartiennent pas aux tournées de $\mathcal{K}$ sont considérés comme un PTV indépendant (de petite taille) qui peut être résolu par la méthode de recherche locale. Les tournées de ce PTV indépendant seront ensuite ajoutées à $\mathcal{K}$ pour créer une solution admissible pour le problème original. Dans nos implémentations, nous avons utilisé une procédure légèrement différente mais plus efficace pour compléter la solution partielle $\mathcal{K}$. La différence réside dans le fait de d'abord essayer d'ajouter les clients non desservis sur des tournées de $\mathcal{K}$ avant d'envisager la création d'une nouvelle tournée. Avec ce procédé nous cherchons donc à minimiser le nombre de camions nécessaire pour desservir la totalité des clients. La solution admissible que nous obtenons est alors considérée comme la solution initiale de la recherche locale qui va essayer d'améliorer sa qualité.

Lorsqu'une nouvelle solution est obtenue après l'exécution de la recherche locale, les tournées qui la composent sont marquées avec la valeur de cette solution et introduites dans l'ensemble $\mathcal{T}$. Les trois étapes de cette seconde phase (la construction d'une solution à partir des tournées de $\mathcal{T}$, l'optimisation de cette solution avec la recherche locale et l'insertion des nouvelles tournées obtenues dans $\mathcal{T}$) sont réitérées jusqu'à ce qu'un critère d'arrêt soit satisfait. De façon plus formelle, notre algorithme de diversification et d'intensification peut se formuler comme suit :

1. **Phase d'initialisation**
   a. Générer I solutions initiales différentes avec la recherche locale,
   b. Marquer chaque tournée avec la valeur de la solution à laquelle elle appartient ;
   c. Ôter les tournées qui ne contiennent qu'un seul client ;
   d. Introduire les tournées restantes dans un ensemble $\mathcal{T}$;
   e. Trier $\mathcal{T}$ dans l'ordre croissant des marques des tournées ;
2. **Phase de diversification et d'intensification**
   (à réitérer jusqu'à ce qu'un critère d'arrêt soit satisfait)
   a. Poser $\mathcal{T}' := \mathcal{T}$; $\mathcal{K} := \emptyset$;
   b. **Tant que** $\mathcal{T}' \neq \emptyset$ **faire**
      (1) Choisir $t \in \mathcal{T}'$ de manière probabiliste ;
      (2) Poser $\mathcal{K} := \mathcal{K} \cup \{t\}$;
      (3) Enlever de $\mathcal{T}'$ toutes les tournées contenant un ou plusieurs clients appartenant à t ;
   c. **Si** certains clients ne sont pas desservis par les tournées de $\mathcal{K}$
      alors Construire une solution admissible $\mathcal{K}'$ à partir de $\mathcal{K}$
      sinon $\mathcal{K}' := \mathcal{K}$;
   d. Améliorer la solution $\mathcal{K}'$ avec la recherche locale ;
   e. Marquer les tournées de la nouvelle solution ainsi obtenue ;
      Introduire dans $\mathcal{T}$ les tournées contenant plus d'un client ;
      Trier $\mathcal{T}$ comme à l'étape e de la phase d'initialisation ;

Tableau 5-2 : Algorithme probabiliste de diversification et d'intensification.

Dans le but de favoriser la création de bonnes solutions, les tournées ne sont pas choisies de manière uniforme au pas 2b(1) mais de sorte que la probabilité de sélection d'une tournée de $\mathcal{T}'$ soit proportionnelle à son classement. Pour des raisons pratiques évidentes, il est nécessaire de limiter la taille de l'ensemble $\mathcal{T}$ à une valeur maximale L.

Ainsi donc après chaque tri de $\mathcal{T}$, les $|\mathcal{T}| - L$ plus mauvaises tournées sont ôtées de $\mathcal{T}$ si le cardinal de $\mathcal{T}$ est supérieur à L.

La probabilité de choisir la $i^{ième}$ tournée la mieux classée dans $\mathcal{T}$ est donnée par l'expression suivante :

$$\boxed{\frac{2\cdot(L+1-i)}{L\cdot(L+1)},\quad i=1,2,\ldots,L}$$

Ce sont bien des probabilités puisque les valeurs que peuvent prendre cette expression sont comprises entre 0 et 1 et que de plus,

$$\sum_{i=1}^{L}\frac{2\cdot(L+1-i)}{L\cdot(L+1)}=\frac{2}{L\cdot(L+1)}\cdot\sum_{i=1}^{L}(L+1-i)=\frac{2}{L\cdot(L+1)}\cdot\sum_{j=1}^{L}j=\frac{2\cdot L\cdot(L+1)}{L\cdot(L+1)\cdot 2}=1$$

La Figure 5.3 représente les probabilités biaisées de sélection d'une tournée dans $\mathcal{T}$ lorsque L vaut 20.

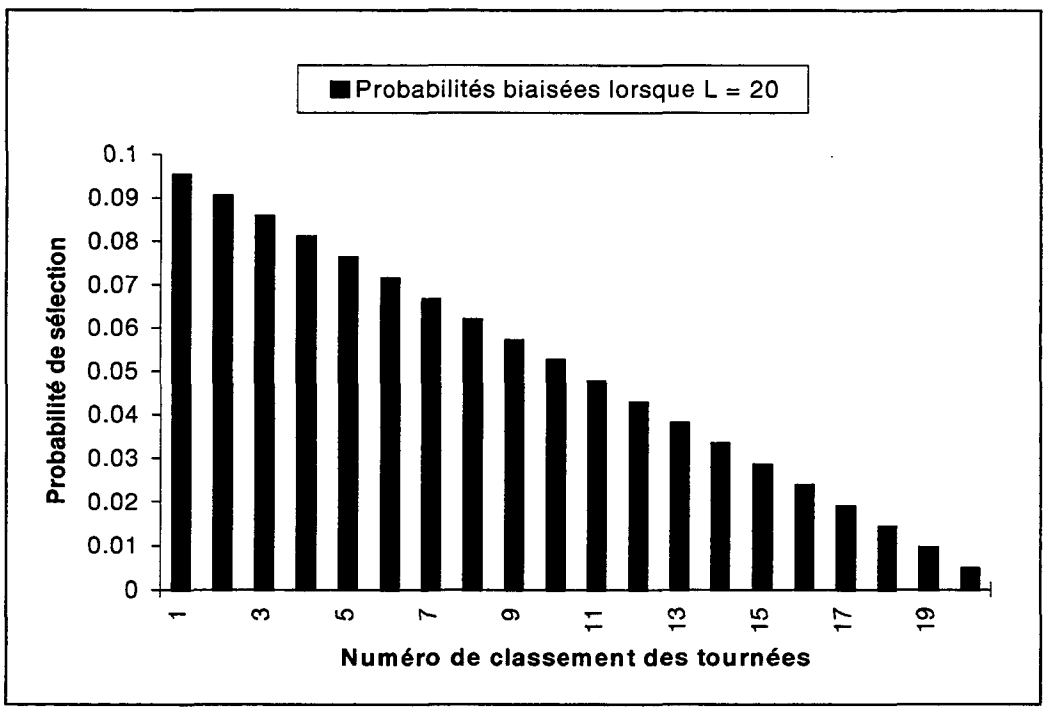


Figure 5.3 : Illustration des probabilités biaisées de sélection d'une tournée de $\mathcal{T}$.

Un aspect implicite de notre méthode est de "combiner" des solutions puisque nous en créons de nouvelles à partir des composantes de plus anciennes. Dans ce sens, notre nouvelle approche incarne une partie de l'esprit des algorithmes génétiques (voir par exemple les articles de Holland [Hol75] et Davis [Dav87]). Cette technique de combinaison implicite couplée avec un processus d'intensification, dont les origines sont

approximativement de la même époque que celles des algorithmes génétiques, est une contrepartie intéressante à la combinaison avec des opérateurs génétiques.

Les principes de fonctionnement de notre algorithme peuvent s'expliquer de la manière suivante. Si les valeurs des paramètres I et L sont suffisamment grandes, alors la phase d'initialisation non déterministe de notre méthode garantit que des régions très diverses de l'espace des solutions seront explorées. La création de nouvelles solutions partielles au début de la seconde phase (point 2b) permet d'augmenter la diversité des solutions visitées par la recherche locale. Au fur et à mesure que le processus évolue, la taille de l'ensemble $\mathcal{T}$ augmente, si bien que les solutions partielles construites à partir de cet ensemble sont de plus en plus complètes. À la fin, ces solutions sont souvent admissibles et parfois de meilleure qualité que la meilleure solution trouvée jusque-là. Après avoir exécuté plusieurs fois la seconde phase de notre méthode, le processus intensifie automatiquement sa recherche dans des régions prometteuses de l'espace des solutions puisque lors du point 2b(1) les tournées ne sont pas choisies uniformément et que de plus, les tournées provenant de solutions de mauvaises qualités sont retirées de $\mathcal{T}$. Il est important de souligner les deux points suivants : premièrement notre processus n'interdit pas la présence de tournées identiques dans l'ensemble $\mathcal{T}$ et deuxièmement, au fur et à mesure que le processus évolue, il existe de plus en plus souvent des tournées qui ne sont pas modifiées par la recherche locale. Ces deux points ont pour conséquence directe le fait que les meilleures tournées de $\mathcal{T}$ sont de plus en plus fréquemment choisies lors de la construction d'une solution partielle si bien que, tout naturellement, la recherche change progressivement d'un processus de diversification à un processus d'intensification. C'est ce phénomène qui a donné le nom à notre nouvelle technique.

Dans le dernier paragraphe de cette section, nous allons montrer comment notre nouvelle approche pourrait être aisément parallélisée. Les étapes qui doivent être exécutées en parallèle sont les étapes 1a et 2d de notre algorithme (utilisation de la recherche locale) puisque ce sont celles qui nécessitent les temps de calcul les plus importants. Dans ce but, nous pouvons adopté une approche maître-esclaves illustrée dans la Figure 5.4. Dans cette figure, nous observons que :

- le processeur maître exécute les étapes 1b à 1e de la phase d'initialisation ainsi que les points 2a, 2b, 2c et 2e de la phase de diversification et d'intensification,
- chacun des I processeurs esclaves applique une recherche locale de l'étape 1a, transmet la solution initiale obtenue au processeur maître, se met en attente de réception d'une solution partielle en provenance du processeur maître, améliore cette solution avec la recherche locale (étape 2d) et finalement communique la nouvelle solution obtenue au processeur maître.

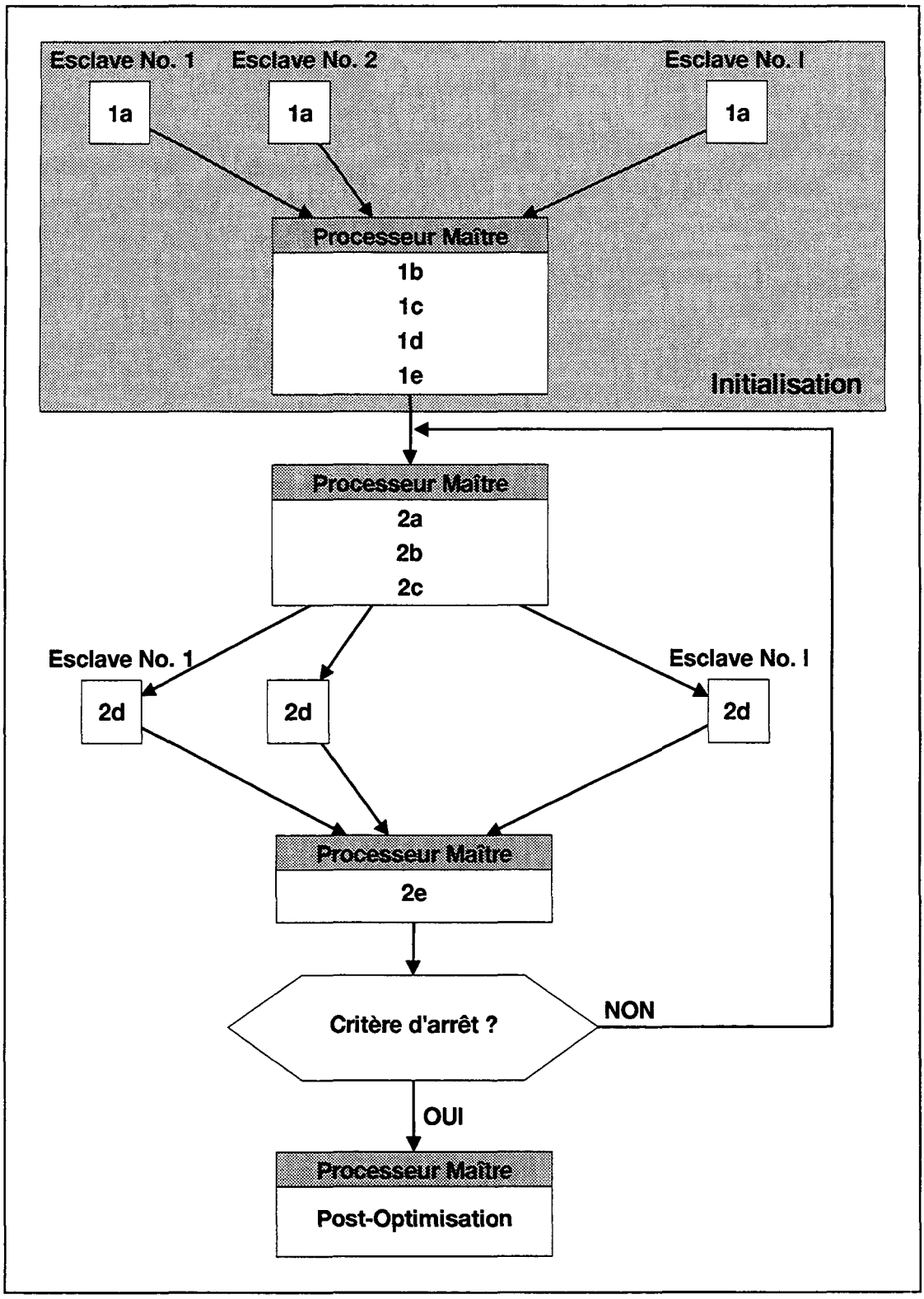


Figure 5.4 : Schéma de parallélisation.

Il est important de noter que le nombre de processeurs esclaves utilisés dans cette approche est indépendant de la taille du problème étudié.

5.3.2 Une technique de post-optimisation

En examinant le comportement empirique de notre technique au cours de la phase de diversification et d'intensification, nous avons remarqué que notre processus parvient parfois à améliorer la meilleure solution connue directement à partir de la solution construite à l'étape 2c, avant d'y appliquer la recherche locale du point 2d. Par conséquent, notre processus est capable de construire de meilleures solutions que celles déjà rencontrées en utilisant uniquement les tournées contenues dans l'ensemble $\mathcal{T}$. Ce constat nous a conduit à développer la procédure de post-optimisation suivante.

Soient L_j la longueur de la $j^{ième}$ tournée de $\mathcal{T}$ ($j = 1, \ldots, |\mathcal{T}|$) et z_{ij} ($i = 1, \ldots, n$ et $j = 1, \ldots, |\mathcal{T}|$) une variable binaire valant 1 si le client i appartient à la $j^{ième}$ tournée de $\mathcal{T}$ ou 0 sinon. Alors, la meilleure solution qui peut être construite en utilisant uniquement des tournées de $\mathcal{T}$ se détermine en résolvant le problème de partitionnement suivant :

$$\begin{aligned} \min \quad & \sum_{j=1}^{|\mathcal{T}|} L_j \cdot x_j \\ \text{s.c.} \quad & \sum_{j=1}^{|\mathcal{T}|} z_{ij} \cdot x_j = 1 \qquad i = 1, \ldots, n \\ & x_j \in \{0,1\} \qquad j = 1, \ldots, |\mathcal{T}| \end{aligned}$$

Une valeur de 1 pour la variable x_j indique que la $j^{ième}$ tournée de $\mathcal{T}$ doit être présente dans la solution. Notons qu'il n'est pas intéressant d'incorporer cette technique de post-optimisation dans la phase de diversification et d'intensification puisque cela ne permet par d'enrichir $\mathcal{T}$ avec des nouvelles tournées. Par conséquent, le moment opportun pour l'application d'une telle technique se situe à la fin de la méthode d'exploration de l'espace des solutions, raison pour laquelle nous parlons d'une technique de post-optimisation.

5.4 Résultats

5.4.1 Procédure de diversification et d'intensification

Dans cette section, nous analysons le comportement de notre technique probabiliste de diversification et d'intensification lorsqu'elle utilise les RT décrites dans la section 5.2 comme méthodes de recherche locale. Définissons tout d'abord les valeurs des paramètres utilisés dans nos expériences numériques :

- le nombre I de solutions initiales est fixé à 20,
- le nombre L de tournées conservées dans T est égal à 260,
- le nombre d'itérations effectuées par la recherche locale au point 1a et 2d est fonction du type de PTV étudié :
  - 14n où n est le nombre de clients pour le PTV élémentaire,
  - 2'000 pour le PTVFT académique.

Ces paramètres ont été choisis de manière relativement arbitraire, sans procéder à une détermination spécifique de leurs valeurs. Par conséquent, il est fort probable que les résultats que nous présentons puissent être encore améliorés. Tous nos tests numériques ont été réalisés sur des stations de travail Silicon Graphics Indigo (200 Mhz) et nos temps de calcul sont exprimés en secondes.

5.4.1.1 PTV élémentaire

Dans le Tableau 5-3 nous indiquons les temps de calcul requis par la méthode de recherche locale de Taillard [Tail93a] et par notre nouvelle méthode pour obtenir des solutions de trois niveaux de qualités moyennes différentes pour le PTV élémentaire. Plus précisément, nous donnons les temps de calcul nécessaires pour fournir des solutions dont la valeur moyenne sur cinq exécutions est 5%, 2% et 1% au-dessus des meilleures solutions connues pour certains problèmes-tests proposés par Christofides et al. [Chr79] (sans la contrainte de durée maximale des tournées), Fisher [Fis94] (avec 71 et 134 clients) et Taillard [Tai93a] (avec 385 clients). Un temps de calcul est imprimé en caractères gras s'il est significativement plus petit (plus que 1.5 fois) que celui requis par l'autre méthode présentée.

	Recherche Locale			Diversification et Intensification		
Taille du problème	5%	2%	1%	5%	2%	1%
50	2.8	7.3	11	2.8	7.3	11
75	1.5	9.4	**27**	1.5	11	68
100	15	94	**420**	15	**57**	900
100b	**33**	**140**	**180**	51	280	350
120	>5000	—	—	**690**	**2100**	**2700**
150	30	**250**	>3900	25	670	**1800**
199	110	**1100**	>14000	89	>3700	—
71	**58**	>1400	—	190	**1080**	**1130**
134	32000	>35000	—	**520**	**2300**	**3100**
385	**370**	8800	>23000	1300	**5400**	**18000**

Tableau 5-3 : Temps de calcul requis pour obtenir des solutions aux problèmes de Christofides et al. [Chr79], Fisher [Fis94] et Taillard [Tai93a] qui sont, en moyenne, un certain pourcentage au-dessus des meilleures solutions connues.

Nous remarquons, pour la plupart des problèmes de Christofides et al. [Chr79], que notre nouvelle technique n'est pas compétitive (du moins pour les paramètres que nous avons choisis) par rapport à la méthode de recherche locale. Cependant, les solutions des problèmes qui s'apparentent le plus avec des problèmes réels (ceux avec 120, 71, 134 et 385 clients) sont généralement de meilleure qualité lorsqu'elles ont été obtenues avec notre nouvelle approche.

Comme l'a observé Taillard pour des problèmes d'affectation quadratique [Tail94], les problèmes réels peuvent se révéler compliqués à résoudre pour des méthodes de recherche locale efficaces pour d'autres types de problèmes. Cela provient du fait que les méthodes de recherche locale sont souvent bloquées dans un optimum local de mauvaise qualité, si bien que les valeurs moyennes des solutions qu'elles fournissent ne sont pas très bonnes. Nous pensons qu'un phénomène similaire a lieu pour le PTV, comme le laissent présager les résultats obtenus pour les problèmes avec 120, 71, 134 et 385 clients. Afin d'étudier le comportement de notre nouvelle technique sur ces problèmes non uniformes, nous avons généré des problèmes de ce type avec une procédure similaire à celle employée par Taillard [Tai94] pour créer des problèmes d'affectation quadratique non uniformes. Ainsi, les clients sont localisés dans le plan et répartis en plusieurs groupes distincts. Le nombre ainsi que la compacité de ces groupes sont très variés. De plus, les poids des commandes à desservir sont exponentiellement distribués si bien que le poids de la commande d'un client pourrait très bien être aussi grand que la charge utile d'un camion. Nous avons généré quatre problèmes (a, b, c et d) pour chacune des catégories suivantes : 75 clients (9 ou 10 véhicules), 100 clients (11 ou 12 véhicules) et 150 clients (14 ou 15 véhicules).

La Figure 5.5 représente la meilleure solution que nous avons trouvée pour le problème 150a qui possède une structure similaire aux problèmes réels proposés par Fisher [Fis94]. En la comparant avec la Figure 5.2 qui montre la meilleure solution connue pour le problème de Christofides et al. avec 199 clients, nous remarquons que les structures de ces deux problèmes sont très différentes.

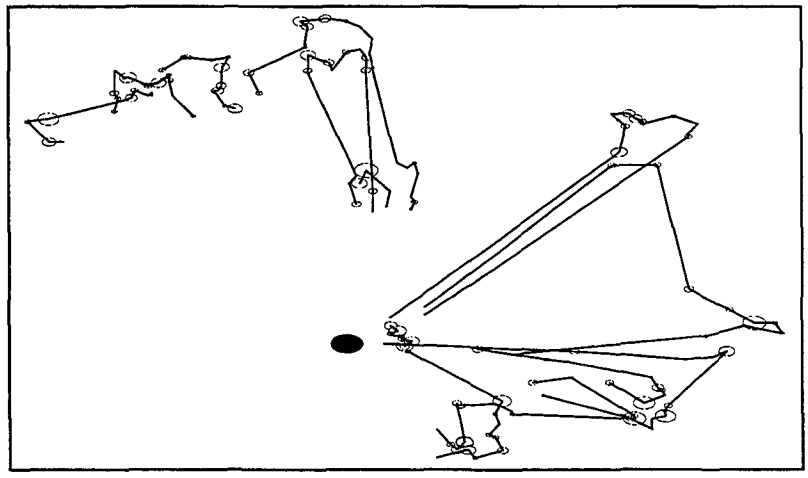


Figure 5.5 : Meilleure solution connue pour un problème non uniforme avec 150 clients.

Dans le Tableau 5-4, nous représentons pour ces problèmes non uniformes des informations similaires à celles illustrées dans le Tableau 5-3. Nous remarquons distinctement que notre nouvelle technique améliore très largement les performances de la recherche locale, spécialement lorsqu'il faut obtenir des solutions de très bonne qualité. Les problèmes non uniformes que nous avons générés ne sont pas seulement difficiles à résoudre pour la méthode de recherche locale mais aussi pour la RT de Gendreau et al. [Gen94]. Pour illustrer cette remarque, nous reportons donc dans ce tableau le temps de calcul requis par cette dernière méthode ainsi que la qualité de la solution qu'elle fournit, exprimée en pourcentage d'écart par rapport aux meilleures solutions connues. Nous remarquons que cet algorithme est très souvent bloqué dans un optimum local de mauvaise qualité, même pour les problèmes avec 75 clients alors qu'il est supérieur (voir Tai[93a]) à notre recherche locale pour quelques uns des problèmes de Christofides et al. [Chr79].

	Recherche Locale			Diversification et Intensification			Gendreau et al. [Gen94]	
Taille du problème	5%	2%	1%	5%	2%	1%	Temps	%
75a	**3.3**	30	170	6.5	**18**	**48**	1900	1.2
75b	**2.8**	14	170	6.4	13	**17**	1700	0.7
75c	15	>2100	—	14	**120**	**170**	1600	9.6
75d	**4.4**	29	52	7.5	**13**	**17**	1700	1.1
100a	13	**53**	>2000	10	220	**720**	3600	4.6
100b	21	110	460	**15**	**38**	**150**	2900	2.3
100c	**35**	190	>6800	55	**140**	**770**	2900	5.1
100d	25	**120**	>2100	28	180	**680**	3300	4.7
150a	>3700	—	—	**480**	**1600**	**2800**	7000	4.3
150b	78	>3100	—	**25**	**390**	**1100**	8600	2.8
150c	**33**	>2800	—	84	**420**	**820**	6400	8.1
150d	100	2000	>3800	96	**980**	**1700**	5800	5.0

Tableau 5-4 : Temps de calcul requis pour obtenir des solutions aux problèmes non uniformes qui sont, en moyenne, un certain pourcentage au-dessus des meilleures solutions connues et présentation des performances de l'algorithme de Gendreau et al. [Gen94].

5.4.1.2 PTV avec contraintes de fenêtres de temps

Nous présentons les résultats obtenus sur un ensemble de 56 problèmes proposés par Solomon [Sol87]. Chacun de ces 56 problèmes est constitué de n=100 clients et le temps de parcours t_{ij} entre deux clients i et j ($0 \leq i, j \leq n$) est égal à la distance euclidienne c_{ij} correspondante. Ils se différencient les uns des autres par les valeurs données aux éléments suivants :

- la capacité des véhicules,
- la durée maximale d'une tournée,
- la distribution spatiale des clients,
- les temps de services survenant lors de la livraison chez un client,
- la densité ainsi que la taille des fenêtres de temps.

Ces problèmes se répartissent dans trois catégories distinctes qui sont : la catégorie R (distribution spatiale uniforme des clients), la catégorie C (distribution groupée des clients) et la catégorie RC (un mélange des catégories R et C). Deux ensembles de problèmes sont proposés pour chacune de ces trois catégories : les ensembles R1, C1 et RC1 caractérisés par un *horizon de planification étroit* et les ensembles R2, C2 et RC2 qui ont un *horizon de planification large*. Les problèmes avec un horizon de planification étroit ont des véhicules d'une faible capacité et une durée maximale par tournée qui est courte si bien que seulement quelques clients peuvent être desservis par un même véhicule. À l'opposé, les problèmes avec un horizon de planification large se caractérisent par une durée maximale par tournée qui est longue et des véhicules de grande capacité de sorte qu'un grand nombre de clients peuvent être desservis par un même véhicule. Le Tableau 5-5 résume les différentes caractéristiques de ces 56 problèmes.

Type de problèmes	Capacité des véhicules	Durée maximum des tournées	Temps de service	Horizon de planification
R1	200	230	10	Étroit
C1	200	1'236	90	Étroit
RC1	200	240	10	Étroit
R2	1'000	1'000	10	Large
C2	700	3'390	90	Large
RC2	1'000	960	10	Large

Tableau 5-5 : Récapitulation des caractéristiques des problèmes de Solomon.

L'objectif à atteindre pour ces 56 problèmes est double :

1. minimiser le nombre de véhicules nécessaires pour desservir les 100 clients,
2. minimiser la distance totale parcourue par ce nombre minimum de véhicules.

Dans le Tableau 5-6, nous comparons les valeurs des solutions obtenues pour le PTVFT académique après cinq exécutions différentes de la RT avec celles produites après une exécution de notre méthode de diversification et d'intensification. Pour chaque type de problèmes, nous indiquons les informations suivantes :

- le temps CPU moyen nécessaire pour effectuer 50'000, 150'000 et 300'000 itérations pour la RT. Pour la méthode de diversification et d'intensification, ce temps CPU correspond respectivement à 25, 75 et 150 appels à la RT effectuant 2000 itérations à chaque fois.

- la valeur moyenne des meilleures solutions obtenues sur cinq exécutions,
- la valeur moyenne de l'intégralité des solutions obtenues,
- la valeur moyenne des plus mauvaises solutions obtenues sur cinq exécutions.

Les valeurs des solutions sont représentées par deux nombres qui sont le nombre moyen de véhicules utilisés et la distance moyenne parcourue.

Type	Temps CPU	Méthode de recherche locale						Diversification et Intensification	
		Minimum		Moyenne		Maximum			
R1	450	**12.75**	**1204.81**	13.00	1225.62	13.33	1255.71	12.83	1208.43
	1300	12.67	1206.54	12.92	1223.74	13.17	1259.10	**12.58**	**1202.31**
	2700	12.67	1204.76	12.92	1222.36	13.17	1257.72	**12.58**	**1197.42**
C1	540	**10.00**	**831.24**	10.00	838.93	10.00	863.10	10.00	832.59
	1600	10.00	831.01	10.00	838.00	10.00	861.41	**10.00**	**829.01**
	3200	10.00	830.32	10.00	836.87	10.00	858.75	**10.00**	**828.45**
RC1	430	**12.63**	**1383.06**	13.00	1418.58	13.25	1475.49	12.75	1381.33
	1300	**12.25**	**1396.48**	12.88	1417.92	13.25	1470.99	12.50	1368.03
	2600	**12.25**	**1385.28**	12.77	1418.36	13.25	1470.99	12.38	1369.48
R2	1600	3.36	991.20	3.62	996.03	3.91	1015.97	**3.18**	**999.63**
	4900	3.36	988.75	3.62	992.48	3.91	1010.86	**3.09**	**969.29**
	9800	3.36	988.43	3.62	990.09	3.91	1004.07	**3.09**	**954.36**
C2	1200	**3.00**	**594.64**	3.10	616.44	3.50	654.99	3.00	595.38
	3600	3.00	591.43	3.00	611.25	3.00	654.91	**3.00**	**590.32**
	7200	3.00	591.43	3.00	610.28	3.00	653.38	**3.00**	**590.32**
RC2	1300	4.00	1171.18	4.18	1249.80	4.38	1422.50	**3.62**	**1207.37**
	3900	4.00	1166.78	4.18	1245.06	4.38	1417.02	**3.62**	**1155.47**
	7800	4.00	1166.78	4.18	1244.77	4.38	1417.02	**3.62**	**1139.79**

Tableau 5-6 : Valeurs moyennes des solutions obtenues après des temps de calcul fixés pour la recherche locale et la technique de diversification et d'intensification. Problèmes de Solomon [Sol87].

Comme pour le PTV élémentaire, nous remarquons que notre nouvelle technique améliore grandement les solutions produites uniquement avec la recherche locale. De surcroît, la valeur de la meilleure solution obtenue après cinq exécutions différentes de la recherche locale est souvent de moins bonne qualité que la valeur moyenne des solutions fournies par notre technique de diversification et d'intensification. Nous avons aussi observé que lorsque nous exécutions plusieurs fois une méthode sur le même problème, la différence entre la meilleure et la plus mauvaise solution obtenue est nettement plus faible pour notre nouvelle approche que pour la recherche locale. Par conséquent, la technique de diversification et d'intensification est plus efficace et robuste, si bien qu'il n'était pas nécessaire de calculer des valeurs moyennes sur plusieurs exécutions dans le Tableau 5-6.

5.4.2 Procédure de post-optimisation

La procédure de post-optimisation nous a permis d'améliorer la qualité de plusieurs des meilleures solutions connues pour des exemples de problèmes de la littérature. Les améliorations qui peuvent être obtenues ne sont pas très grandes, généralement inférieures à 1 %, mais elles le sont avec seulement une très légère augmentation du temps de calcul. Nous illustrerons ces propos sur les nouveaux exemples de problèmes qui ont été générées pour le PTV élémentaire. Nous considérons pour cela cinq exécutions de notre nouvelle technique pour chaque exemple des problèmes non uniformes avec 100 et 150 clients. Pour chacune de ces exécutions, nous utilisons I = 20 solutions initiales et 50 ou 70 appels à notre procédure de diversification et d'intensification selon que le nombre de clients est de 100 ou 150. Toutes les solutions obtenues, c'est-à-dire 350 ou 450 pour chaque problème, ont été stockées. Parmi les meilleures solutions produites, nous avons choisi 250 tournées différentes pour créer l'ensemble T. Finalement, nous avons résolu de façon exacte le problème de partitionnement induit par T en utilisant le logiciel "Cplex Mix Integer Library" [Cpl93].

Dans le Tableau 5-7, nous indiquons les informations suivantes pour chaque exemple des problèmes non uniformes avec 100 et 150 clients :

- la valeur de la meilleure solution obtenue après cinq exécutions de la méthode de diversification et d'intensification,
- le temps moyen d'une exécution de notre nouvelle technique,
- la valeur de la solution produite par notre procédure de post-optimisation,
- le temps de calcul de la procédure de post-optimisation,
- la valeur de la meilleure solution connue.

Taille des problèmes	Diversification et Intensification		Post-optimisation		Meilleure solution connue
	Valeur	Temps CPU	Valeur	Temps CPU	
100a	2062.27	1000	**2047.90**	5.5	2047.90
100b	**1940.61**	1000	**1940.61**	3.9	1940.61
100c	1421.59	1000	**1407.44**	71	1407.44
100d	**1581.25**	1000	**1581.25**	1.2	1581.25
150a	3077.77	2100	3070.91	0.43	3055.23
150b	2735.16	2100	2733.60	1.9	2727.99
150c	2367.65	2100	2364.31	18	2362.79
150d	2668.34	2100	2663.20	22	2655.67

Tableau 5-7 : Comparaison des résultats fournis par la technique de diversification et d'intensification et la procédure de post-optimisation.

Dans ce dernier tableau, nous remarquons que l'algorithme de diversification et d'intensification a trouvé les meilleures solutions connues pour la moitié des problèmes avec 100 clients. De plus, la procédure de post-optimisation a réussi à améliorer les autres solutions en trouvant les meilleures solutions connues. Pour les problèmes avec 150 clients, nous observons que la différence moyenne entre les meilleures solutions fournies par notre nouvelle approche et les meilleures solutions connues a été réduite de près de 40 % par la post-optimisation. Les temps de calcul nécessaires à l'exécution de la procédure de post-optimisation sont souvent très faibles bien que l'écart–type pour un problème donné puisse être très élevé. Nous en concluons donc que la procédure de post-optimisation permet d'améliorer rapidement les solutions produites par la recherche locale mais que la difficulté du problème de partitionnement sous-jacent varie énormément.

5.4.3 Les meilleures solutions connues

Notre technique de diversification et d'intensification, couplée avec la procédure de post-optimisation, nous a permis d'améliorer la qualité de plusieurs des meilleures solutions connues de divers problèmes-tests de la littérature. Pour le PTV élémentaire, le Tableau 5-8 indique pour chaque problème étudié : la référence où il est décrit, son numéro d'identification (lorsqu'il existe), sa taille, la valeur de la meilleure solution publiée avec la référence de l'article dans lequel elle a été publiée et finalement, la valeur de la meilleure solution que nous avons obtenue dans le cadre de notre étude.

Origine du problème	Numéro du problème	Taille	Meilleure solution publiée	Nouvelle meilleure solution
Fisher [Fis94]	12	134	1163.60[a]	1162.96
Christofides et al. [Chr79]	5	199	1298.79[b]	1291.45
Christofides et al. [Chr79]	10	199	1396.94[b]	1395.85
Taillard [Tai93a]	—	385	24599.6[b]	24435.5

Tableau 5-8 : Meilleures solutions publiées et nouvelles meilleures solutions pour le PTV élémentaire. Références : [a] = Fisher [Fis94], [b] = Taillard [Tai93a].

Pour le PTVFT académique, le Tableau 5-9 donne le même type d'information sur les problèmes de Solomon. L'origine ([Sol87]) ainsi que la taille (100) des problèmes ne sont pas indiqués. De plus, la valeur d'une solution est indiquée par une paire de nombres qui représente respectivement le nombre de véhicules utilisés et la distance parcourue.

Type	Meilleure solution publiée		Meilleure solution avec la recherche locale		Meilleure solution avec notre nouvelle technique	
R101	18	1607.7[a]	19	1656.20	*19*	*1650.80*
R102	17	1434.0[a]	18	1477.41	*17*	*1486.12*
R103	13	1207[b]	14	1222.90	*14*	*1213.62*
R104	10	1048[b]	**10**	**1013.26**	**10**	**982.01**
R105	14	1420.94[c]	**14**	**1404.75**	14	**1377.11**
R106	12	1350[b]	**12**	**1293.92**	12	**1252.03**
R107	11	1146[b]	**11**	**1085.77**	**10**	**1159.85**
R108	10	989[b]	**10**	**965.28**	**9**	**980.95**
R109	12	1205[c]	**12**	**1186.41**	**11**	**1235.68**
R110	11	1105[b]	11	1107.90	**10**	**1080.36**
R111	10	1151[b]	11	1070.90	**10**	**1129.88**
R112	10	992[b]	**10**	**965.66**	**10**	**953.63**
C101	10	827.3[a]	**10**	**828.94**	**10**	**828.94**
C102	10	827.3[a]	**10**	**828.94**	**10**	**828.94**
C103	10	835[b]	**10**	**828.06**	**10**	**828.06**
C104	10	840[b]	10	841.59	**10**	**824.78**
C105	10	828.94[c]	**10**	**828.94**	**10**	**828.94**
C106	10	827.3[a]	**10**	**828.94**	**10**	**828.94**
C107	10	827.3[a]	**10**	**828.94**	**10**	**828.94**
C108	10	827.3[a]	**10**	**828.94**	**10**	**828.94**
C109	10	828.94[c]	**10**	**828.94**	**10**	**828.94**
RC101	14	1669[b]	15	1737.03	*15*	*1623.58*
RC102	13	1557[b]	**13**	**1480.66**	**13**	**1477.54**
RC103	11	1110[b]	11	1264.30	*11*	*1262.02*
RC104	10	1204.07[c]	**10**	**1157.23**	**10**	**1135.83**
RC105	14	1602[b]	15	1543.16	**13**	**1733.56**
RC106	12	1485.67[c]	**12**	**1415.62**	**12**	**1384.92**
RC107	11	1274.71[c]	**11**	**1262.43**	**11**	**1230.95**
RC108	10	1281[b]	11	1149.64	**10**	**1170.70**
R201	4	1354[b]	4	1485.36	**4**	**1281.58**
R202	3	1530.49[c]	4	1101.49	*4*	*1088.07*
R203	3	1126[b]	4	912.98	**3**	**948.74**
R204	2	914[d]	3	824.62	**2**	**869.29**
R205	3	1128[b]	3	1205.58	**3**	**1063.24**
R206	3	833[b]	3	956.05	*3*	*912.97*
R207	3	904[b]	**3**	**814.84**	**3**	**814.78**
R208	2	759.21[c]	3	708.78	**2**	**738.60**
R209	2	855[b]	4	901.88	*3*	*944.64*
R210	3	1052[b]	3	1087.32	**3**	**967.50**
R211	3	816[b]	**3**	**794.46**	**2**	**949.49**
C201	3	591.56[c]	3	591.56	**3**	**591.56**
C202	3	591.56[c]	**3**	**591.56**	**3**	**591.56**
C203	3	591.55[c]	**3**	**591.17**	**3**	**591.17**
C204	3	590.60[c]	3	597.76	**3**	**590.60**
C205	3	588.88[c]	**3**	**588.88**	**3**	**588.88**
C206	3	588.49[c]	**3**	**588.49**	**3**	**588.49**
C207	3	588.32[c]	3	588.49	**3**	**588.29**
C208	3	588.49[c]	**3**	**588.49**	**3**	**588.32**
RC201	4	1249[b]	5	1469.73	*4*	*1438.89*
RC202	4	1221[b]	4	1443.66	**4**	**1165.57**
RC203	3	1203[b]	4	1013.99	**3**	**1079.57**
RC204	3	897[b]	**3**	**843.12**	**3**	**806.75**
RC205	4	1389[b]	5	1286.70	**4**	**1333.71**
RC206	3	1213[b]	4	1207.76	**3**	**1212.64**
RC207	3	1181[b]	4	1079.07	**3**	**1085.61**
RC208	3	919[b]	3	919.83	**3**	**833.97**

Tableau 5-9 : Résultats pour les problèmes académiques de Solomon [Sol87].
Références : a = Desrochers et al. [Des92], b = Thangiah et al. [Tha94], c = Potvin et Bengio [Pot96] et d = Chiang et Russell [Chi93].

Dans le Tableau 5-9, nous observons (en caractères gras) que la RT que nous avons utilisée (cinq solutions initiales et 400'000 itérations par exécution) a amélioré ou atteint pas moins de 27 des 56 meilleures solutions publiées. Les distances reportées dans la littérature sont parfois inexactes suite aux façons utilisées pour les calculer (distances arrondies et / ou tronquées). Les résultats obtenus dans le cadre de notre étude sont des distances réelles avec double précision où la dernière décimale a été arrondie. La dernière colonne du Tableau 5-9 reporte la meilleure solution obtenue lors de nos expériences numériques pour chaque problème de Solomon. Lorsque celle-ci est la meilleure solution connue, nous indiquons sa valeur en caractères gras. Dans le cas contraire, sa valeur est représentée en caractères italiques. Nous observons que seulement 9 des 56 meilleures solutions publiées pour ces problèmes n'ont pas été atteintes ou améliorées. Ces résultats sont donc très prometteurs et démontrent bien la puissance de notre nouvelle technique de diversification et d'intensification.

Pour terminer la présentation de nos résultats, nous reportons encore, dans le Tableau 5-10, les résultats obtenus sur les exemples D417 et E417 des problèmes proposés par Russell [Rus95] pour le PTVFT académique. Le nombre de clients à desservir pour ces problèmes est égal à 417. Les meilleures solutions connues ont été publiées par Thangiah et al. [Tha94].

Type	Meilleure solution connue		Diversification et Intensification			
D417	**54**	**4866**	54	6264.80	55	3467.83
E417	55	4149	**54**	**7211.83**	**55**	**3693.24**

Tableau 5-10 : Résultats pour les problèmes de Russell avec 417 clients.

Comme nous n'avons pas réussi à diminuer de manière significative les longueurs totales des tournées obtenues avec 54 véhicules, nous reportons aussi les meilleures solutions trouvées en utilisant 55 camions.

5.5 Conclusion

Nous avons présenté une technique probabiliste qui permet de diversifier, d'intensifier et de paralléliser n'importe quelle méthode de recherche locale pour un très grand nombre de variantes du PTV. Cette technique rend la recherche locale plus robuste puisqu'elle converge plus souvent vers des solutions dont la qualité est proche ou

meilleure que les meilleures solutions connues à ce jour pour des problèmes-tests. De surcroît, notre nouvelle technique a plusieurs avantages que nous détaillons dans les paragraphes suivants.

Généralement, il est relativement aisé de développer une méthode de recherche locale qui construit des bonnes tournées pour un sous-ensemble des clients à desservir alors qu'il est difficile de concevoir une recherche qui détermine des bonnes tournées pour la totalité des clients simultanément. Notre technique permet de surmonter cette difficulté et ainsi de concevoir plus aisément des méthodes de recherche qui sont robustes.

Un deuxième avantage de notre approche est qu'elle peut être appliquée à une grande variété de problèmes d'élaboration de tournées de véhicules, par exemple ceux qui tiennent compte des contraintes suivantes :

- flotte hétérogène de véhicules,
- fenêtres de temps pour les livraisons,
- longueur et / ou durée maximale sur les tournées,
- accessibilité de la flotte de véhicules,
- livraison et / ou ramassage chez chaque client.

De plus, notre technique peut être parallélisée aisément avec un nombre arbitraire de processeurs, c'est-à-dire dont le nombre ne dépend pas de la taille du problème étudié. Cependant, nous avons clairement montré son intérêt et sa puissance même lorsqu'elle est utilisée sur un ordinateur mono–processeur.

Finalement, la procédure de post-optimisation proposée permet d'améliorer, dans la grande majorité des cas et sans augmentation sensible du temps de calcul, les solutions obtenues à la fin de la phase de diversification et d'intensification.

5.6 Annexe : quelques meilleures solutions

Problème de Fisher [Fis94] avec 134 clients

Nombre de clients	Tournée	Distance	Charge
41	91-21-25-26-27-28-92-29-93-94-45-44-43-40-3-41-42-2-4-5-6-7-8-9-10-12-11-14-88-15-13-16-90-89-87-86-85-84-83-20-82	187.06	2145
27	72-75-1-62-53-102-104-101-35-36-99-100-98-37-95-39-38-96-97-105-57-56-103-55-54-61-60	64.89	2066
20	78-133-68-70-69-10-111-125-112-126-124-123-122-121-127-128-129-113-81-17	225.29	2167
15	47-32-34-48-49-50-51-52-58-30-31-59-23-24-22	55.63	2140
13	73-74-134-76-77-64-63-79-67-80-33-71-66	89.42	1864
11	46-118-18-132-116-131-117-1119-130-65-19	205.33	2029
7	120-109-108-107-106-114-115	335.34	2209
134		1162.96	14620

Problème de Christofides et al. [Chr79] avec 199 clients

Nombre de clients	Tournée	Distance	Charge
15	89-166-60-84-17-113-86-140-38-14-43-15-57-144-137	117.07	194
14	59-151-92-37-98-100-193-91-85-93-104-99-96-6	57.16	197
14	82-46-124-168-47-36-143-49-64-11-175-107-19-123	130.76	200
14	101-122-20-188-66-71-65-136-35-135-164-34-78-50	122.72	200
14	195-134-163-24-29-121-169-129-79-185-158-3-77-184	83.23	195
13	2-178-115-145-41-22-133-75-74-171-73-21-105	70.37	197
13	54-130-165-55-25-170-67-23-186-56-197-72-40	99.52	200
13	189-10-108-90-126-63-181-32-131-160-128-30-70	97.09	200
13	5-173-61-16-141-191-44-119-192-142-42-172-87	83.14	200
12	146-88-148-159-62-182-48-7-194-106-153-52	73.43	200
11	26-149-179-155-4-139-187-39-110-198-180	71.21	194
11	76-196-116-68-80-150-177-109-12-138-154	49.69	199
11	27-167-127-190-31-162-69-132-176-111-28	47.26	198
10	147-118-83-199-125-45-174-8-114-18	66.13	195
10	1-51-103-161-9-120-81-33-157-102	77.09	199
10	112-183-94-95-97-117-13-58-152-53	41.08	199
1	156	4.47	19
199		1291.45	3186

Problème R107

Nombre de clients	Tournée	Distance	Attente	Durée	Charge
11	52-7-11-62-88-31-10-63-90-32-70	116.34	0.79	227.13	138
11	98-85-91-44-14-38-86-16-100-37-97	103.56	6.56	220.13	197
11	42-43-15-41-22-56-4-74-72-73-58	113.55	4.27	227.82	128
11	60-83-45-46-8-84-5-17-61-93-96	112.19	0.30	222.48	121
10	94-95-92-59-99-6-87-57-2-13	75.71	9.72	185.43	152
10	28-27-69-76-40-53-26-68-24-80	121.70	3.97	225.66	136
9	21-75-39-23-67-55-25-54-12	117.83	14.43	222.26	159
9	1-33-30-20-9-66-71-35-81	132.76	0.00	222.77	141
9	47-36-64-49-19-48-82-18-89	126.63	0.00	216.63	167
9	51-65-79-78-34-29-3-77-50	139.57	0.00	229.57	119
100		1159.84	40.04	2199.88	1458

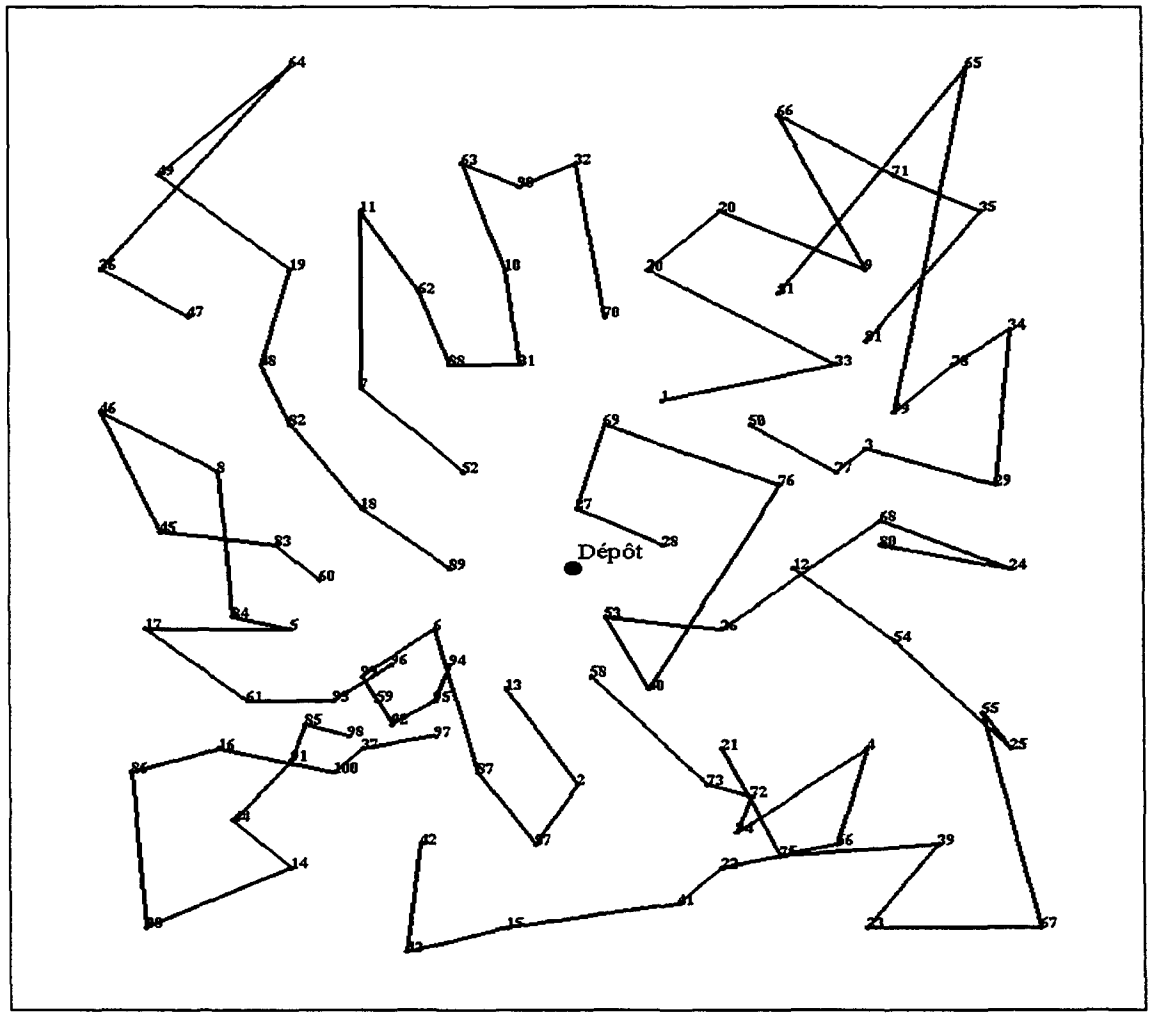


Figure 5.6 : Nouvelle meilleure solution pour R107 : 1159.84 avec 10 camions.

Problème R108

Nombre de clients	Tournée	Distance	Attente	Durée	Charge
13	95-97-59-96-99-93-5-84-17-45-83-60-89	86.43	4.74	221.17	182
12	27-1-69-50-76-28-53-26-40-13-94-6	96.27	0.00	216.27	167
12	92-98-91-44-14-38-86-16-61-85-100-37	106.08	0.00	226.08	200
12	2-57-15-43-42-87-41-22-74-73-21-58	107.37	0.00	227.37	129
11	80-24-29-79-81-9-51-33-3-68-12	108.04	5.39	223.44	172
10	72-75-56-23-67-39-55-4-25-54	125.95	0.00	225.95	179
10	52-7-82-8-46-36-47-19-48-18	108.10	21.55	229.66	137
10	70-30-20-66-65-71-35-34-78-77	119.65	0.00	219.65	134
10	31-88-62-11-49-64-63-90-32-10	123.06	0.00	223.05	158
100		980.95	31.68	2012.64	1458

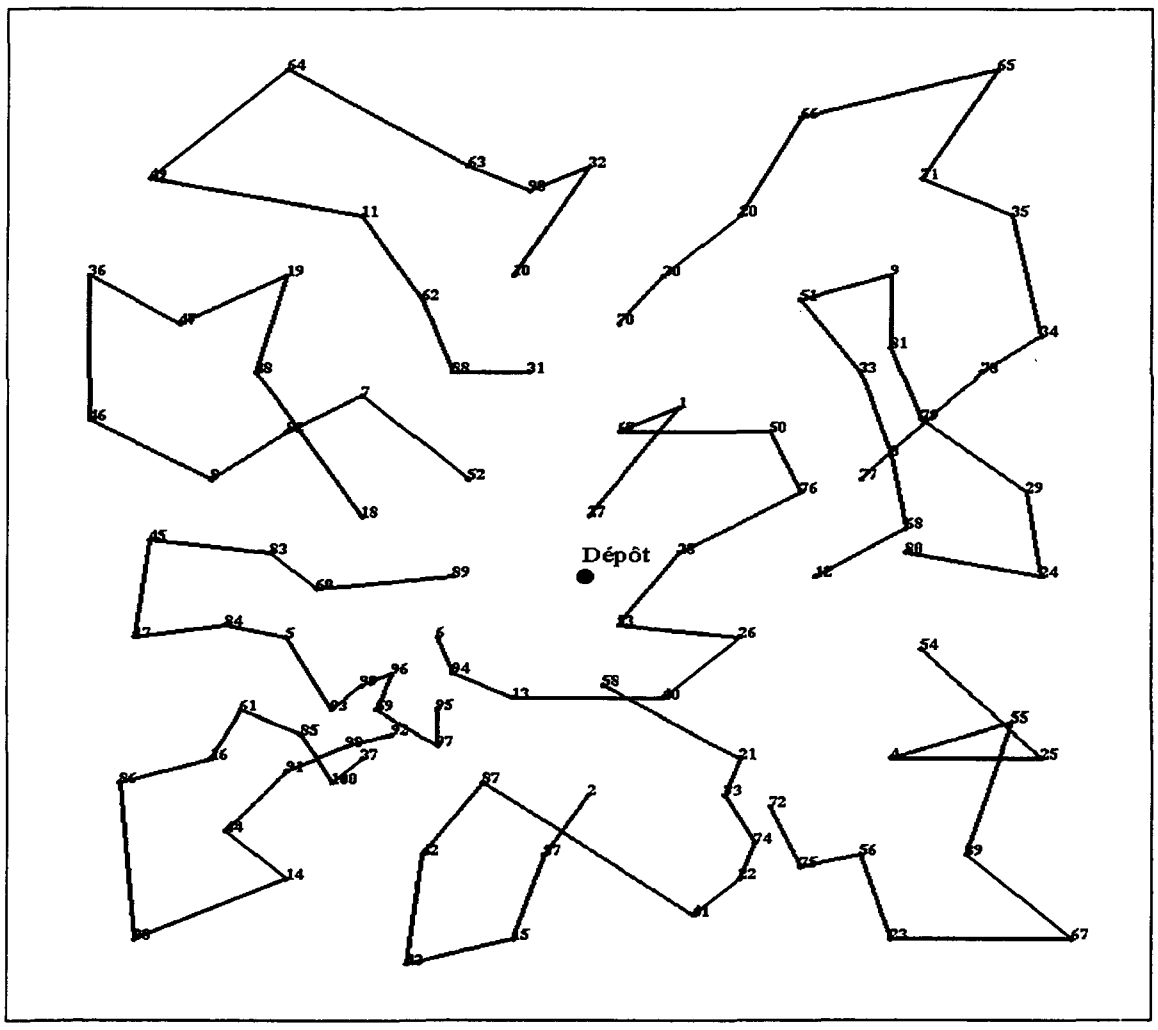


Figure 5.7 : Nouvelle meilleure solution pour R108 : 980.95 avec 9 camions.

Problème R109

Nombre de clients	Tournée	Distance	Attente	Durée	Charge
11	95-92-98-42-15-57-87-97-43-13-58	116.22	0.00	226.22	138
10	59-14-44-38-86-84-93-37-100-91	123.29	0.00	223.29	172
10	12-76-33-81-78-79-3-50-68-80	94.64	0.00	194.64	163
10	27-69-31-30-51-9-66-20-70-1	106.28	0.00	206.27	145
10	2-72-73-21-40-53-26-54-55-25	115.88	0.00	215.88	118
9	52-88-7-18-8-46-17-60-89	105.52	4.94	200.47	65
9	39-67-23-75-22-41-74-56-4	120.25	5.13	215.38	159
9	83-5-61-16-85-99-94-6-96	80.08	0.13	170.21	160
8	28-29-71-65-35-34-24-77	143.44	0.00	223.44	99
7	45-82-19-47-36-49-48	124.37	0.00	194.37	147
7	62-11-64-63-90-32-10	105.71	0.00	175.71	92
100		1235.68	10.20	2245.88	1458

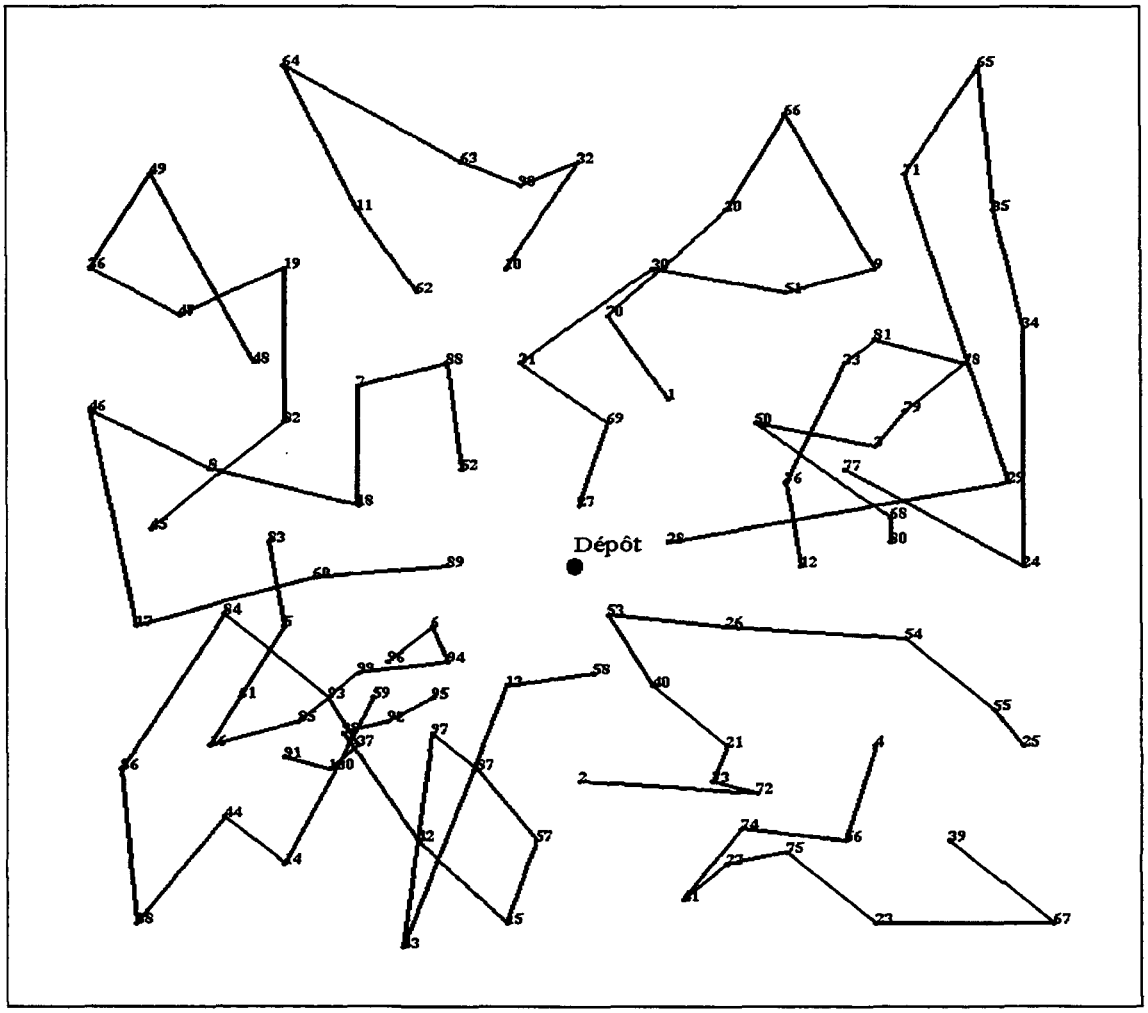


Figure 5.8 : Nouvelle meilleure solution pour R109 : 1235.68 avec 11 camions.

Problème C104

Nombre de clients	Tournée	Distance	Attente	Durée	Charge
13	43-42-41-40-44-46-45-48-51-50-52-49-47	64.81	0.00	1234.81	160
11	20-24-25-27-29-30-28-26-23-22-21	50.80	0.00	1040.80	170
11	5-3-7-8-11-9-6-4-2-1-75	56.18	65.24	1111.42	170
10	81-78-76-71-70-73-77-79-80-63	128.04	0.00	1028.04	200
10	67-65-62-74-72-61-64-68-66-69	57.79	64.02	1021.81	150
10	90-87-86-83-82-84-85-88-89-91	76.07	0.00	976.07	170
9	13-17-18-19-15-16-14-12-10	96.04	0.00	906.04	200
9	34-36-39-38-37-35-31-33-32	97.23	163.61	1070.84	200
9	98-96-95-94-92-93-97-100-99	95.94	0.00	905.94	190
8	55-54-53-56-58-60-59-57	101.88	0.62	822.50	200
100		824.78	293.49	10118.27	1810

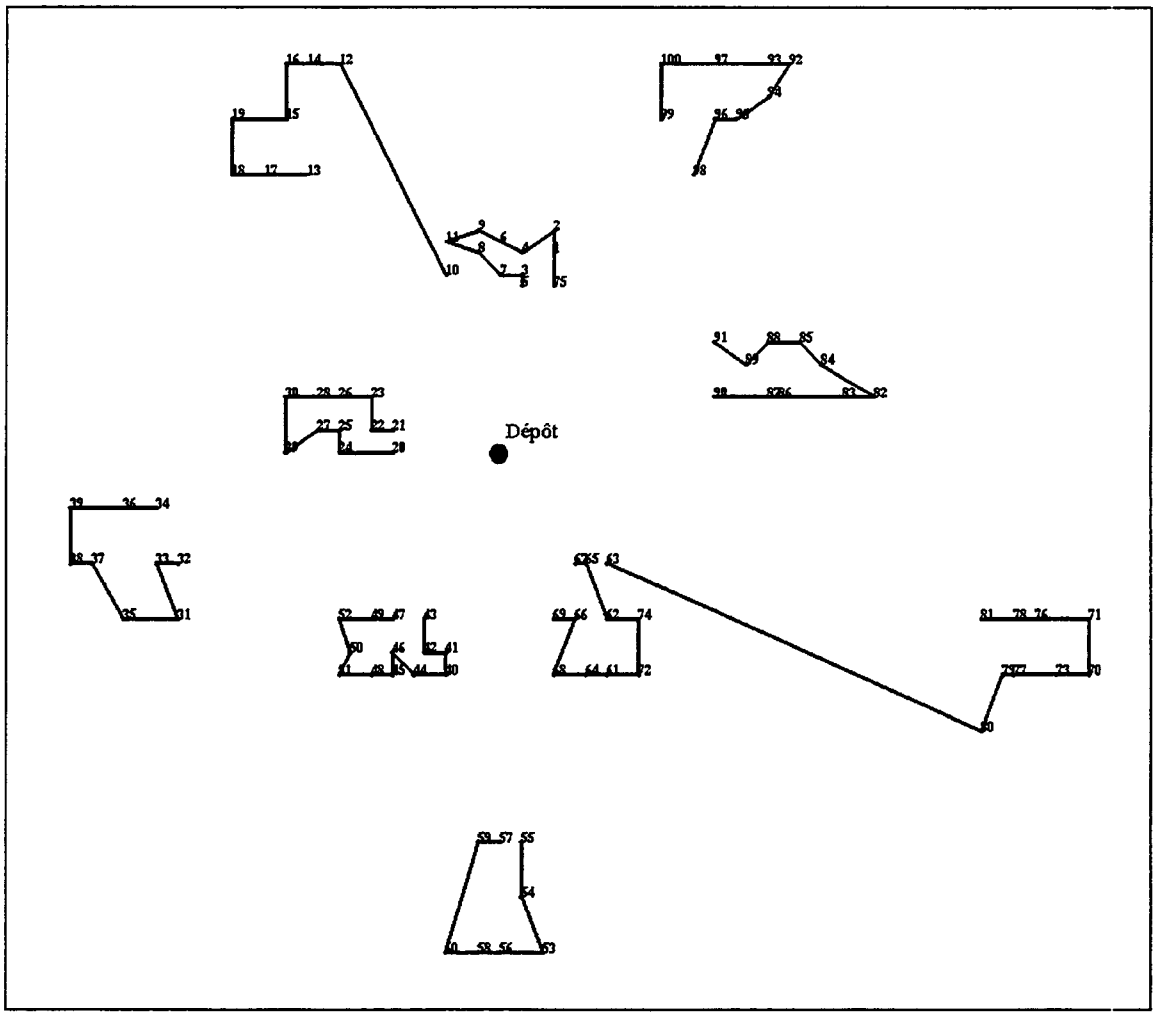


Figure 5.9 : Nouvelle meilleure solution pour C104 : 824.78 avec 10 camions.

Problème RC105

Nombre de clients	Tournée	Distance	Attente	Durée	Charge
10	29-27-26-30-28-32-31-34-50-80	134.65	0.00	234.65	183
9	21-19-23-18-22-49-20-48-25	142.44	0.00	232.44	190
9	39-36-44-38-40-37-35-43-70	132.93	2.33	225.26	193
8	69-88-79-46-4-3-1-100	104.60	11.22	195.81	159
8	72-71-81-41-54-96-94-93	127.54	6.06	213.61	120
8	92-95-64-99-52-86-74-24	124.33	20.17	224.50	101
8	12-14-47-15-16-9-57-77	152.00	0.00	232.00	130
8	98-82-11-87-59-97-75-58	141.11	11.08	232.19	161
7	83-65-90-53-10-13-17	125.70	27.79	223.49	101
7	63-62-67-84-51-89-91	157.50	0.11	227.62	93
7	5-45-7-73-78-55-68	121.33	0.00	191.33	119
6	42-61-8-6-2-60	138.42	0.00	198.42	96
5	33-76-85-56-66	131.01	0.00	181.01	78
100		1733.56	78.76	2812.33	1724

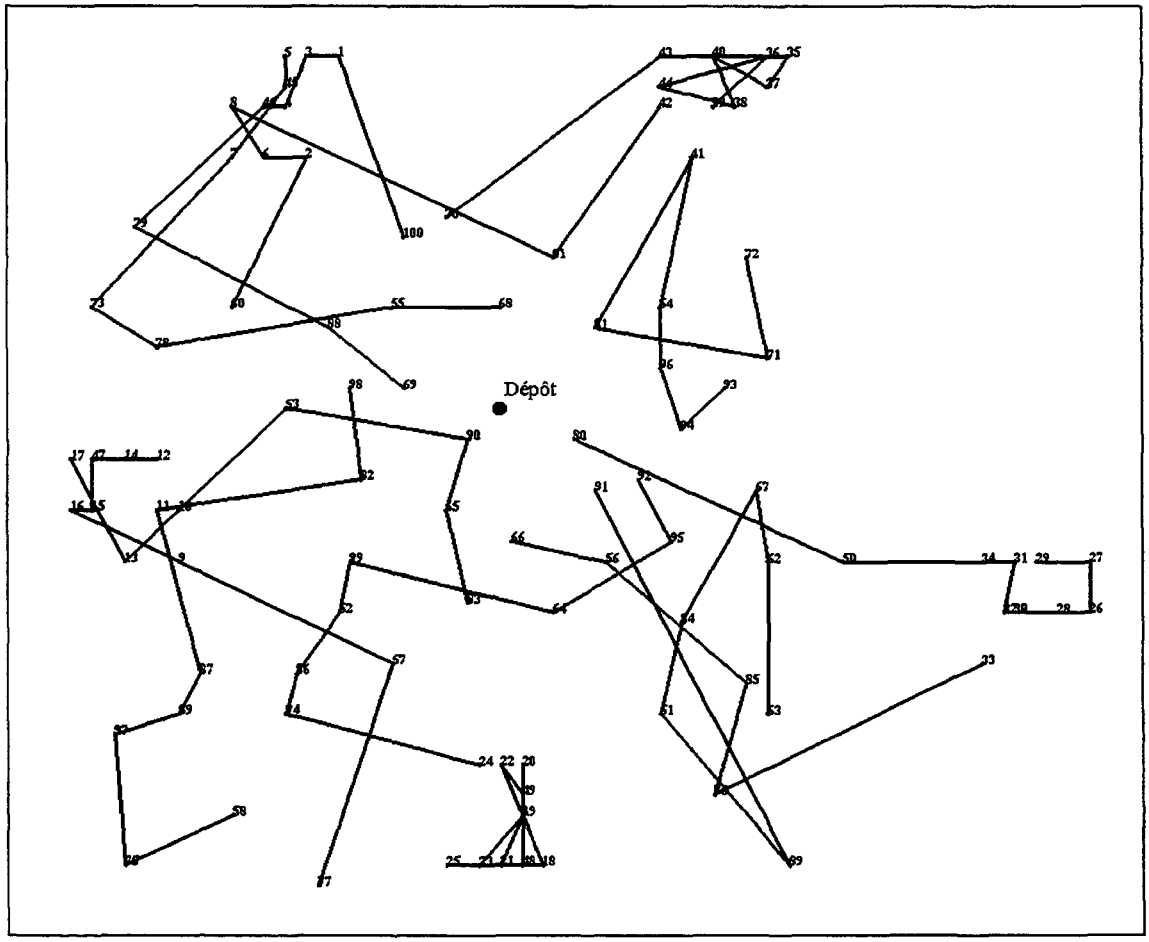


Figure 5.10 : Nouvelle meilleure solution pour RC105 : 1733.56 avec 13 camions.

Problème RC106

Nombre de clients	Tournée	Distance	Attente	Durée	Charge
11	2-45-5-8-7-6-46-4-3-1-100	109.75	0.00	219.75	193
10	11-12-14-47-15-16-9-10-13-17	126.27	0.00	226.27	200
10	42-44-39-38-36-40-41-43-37-35	132.65	3.46	236.10	200
9	83-64-19-23-21-18-48-25-77	131.07	1.31	222.38	168
9	92-67-33-30-32-34-50-93-80	120.78	19.19	229.97	129
9	95-62-63-85-76-51-84-56-66	109.98	0.00	199.97	120
8	65-52-87-59-75-97-58-74	133.80	15.36	229.16	130
8	82-99-86-57-22-49-20-24	103.77	16.94	200.71	145
7	31-29-27-26-28-89-91	153.83	7.71	231.54	128
7	72-71-94-61-81-54-96	105.00	11.71	186.72	106
7	69-98-88-78-73-79-60	91.79	0.00	161.79	134
5	90-53-55-70-68	66.24	7.30	123.54	71
100		1384.93	82.98	2467.90	1724

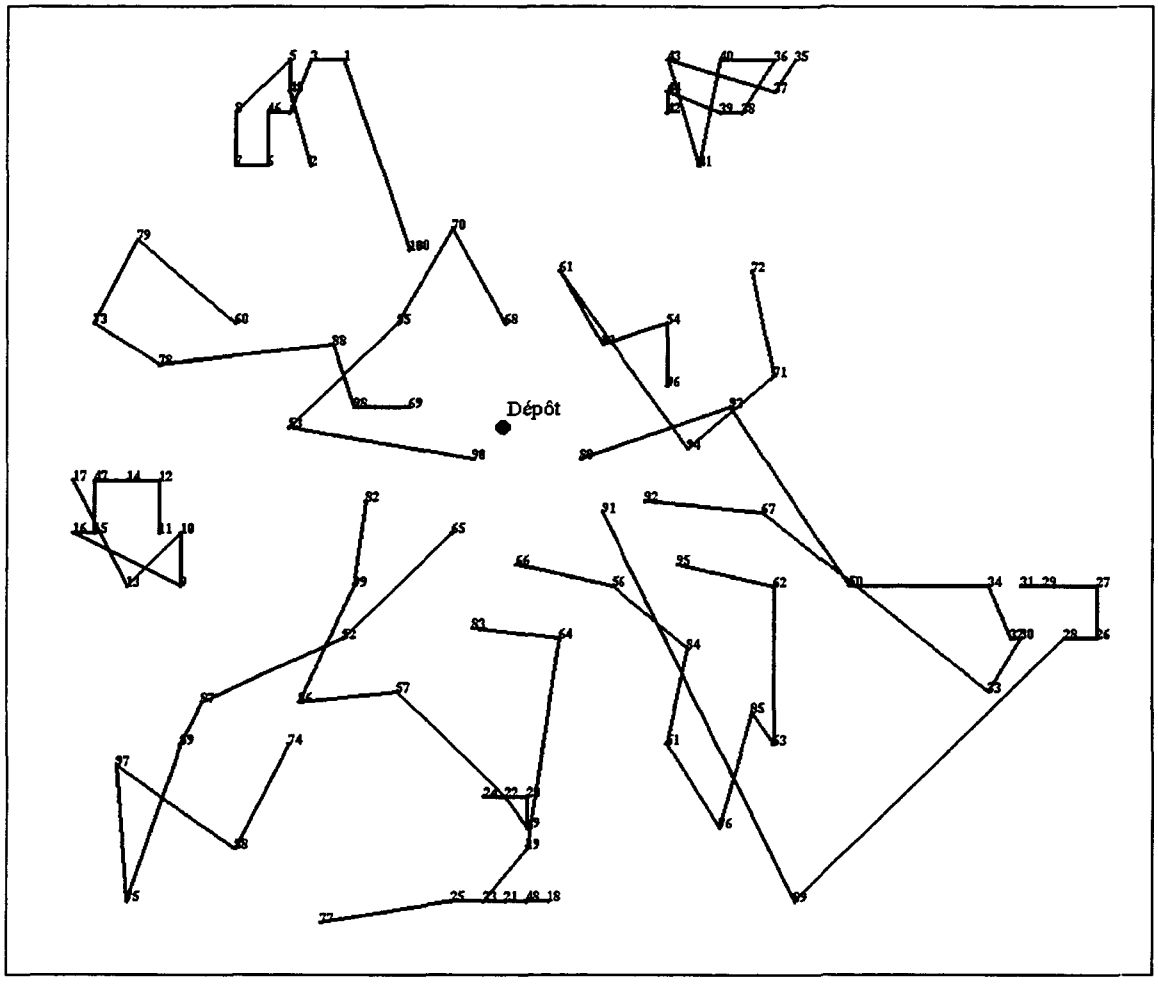


Figure 5.11 : Nouvelle meilleure solution pour RC106 : 1384.93 avec 12 camions.

Problème R201

Nombre de clients	Tournée	Distance	Attente	Durée	Charge
29	5-83-45-82-52-31-69-12-29-76-30-71-9-51-81-79-78-34-50-3-68-26-54-4-55-25-24-80-77	337.08	267.16	894.24	427
25	33-65-63-62-47-36-64-11-19-7-88-90-18-6-94-96-97-37-43-100-91-93-60-17-48	372.04	298.62	920.66	308
25	95-59-92-42-15-14-98-61-16-44-38-86-85-99-84-8-49-46-10-20-32-66-35-70-1	334.96	169.21	754.16	387
21	27-28-2-72-39-75-23-67-21-73-40-53-87-57-41-22-56-74-13-58-89	237.50	344.95	792.46	336
100		1281.58	1079.94	3361.52	1458

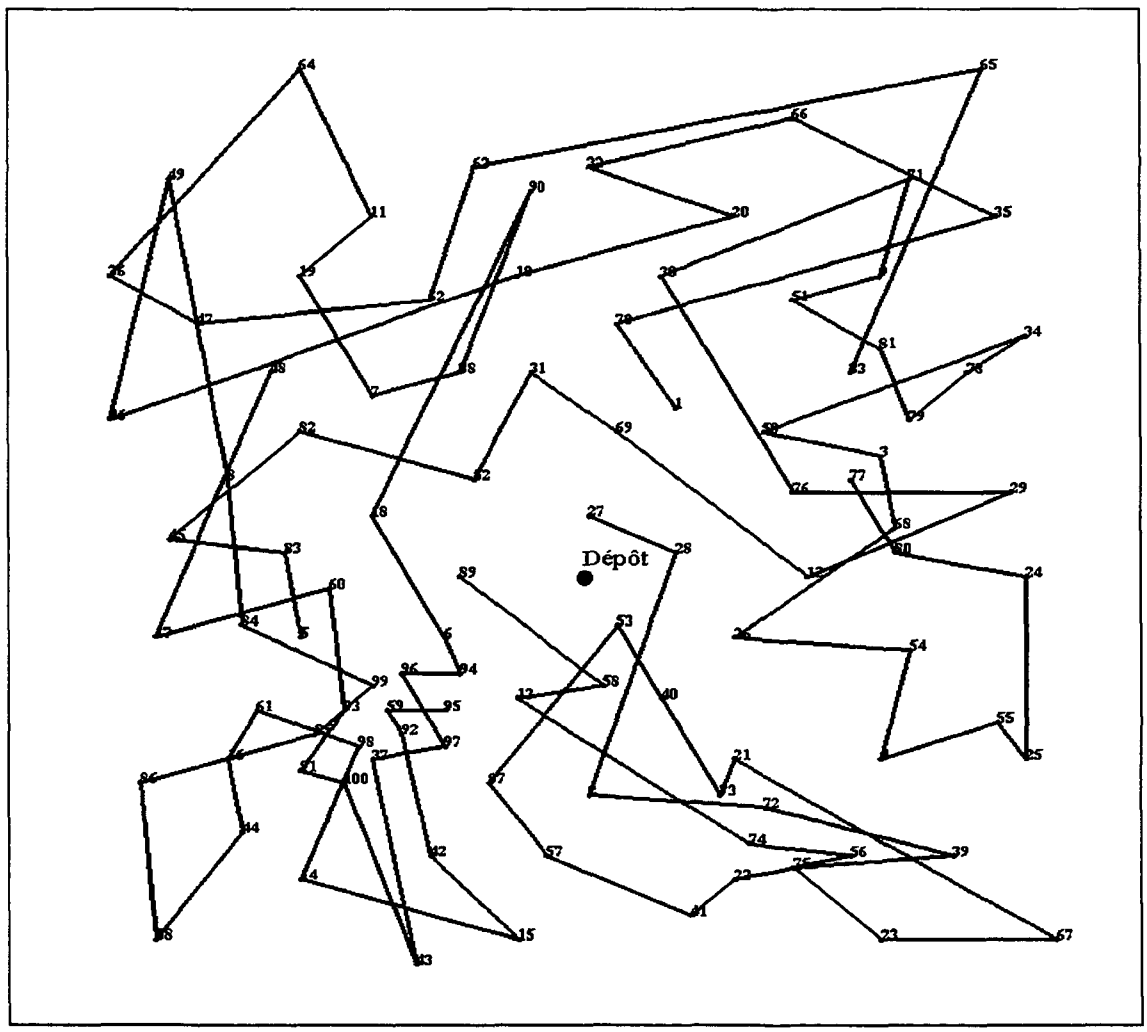


Figure 5.12 : Nouvelle meilleure solution pour R201 : 1281.58 avec 4 camions.

Problème 211

Nombre de clients	Tournée	Distance	Attente	Durée	Charge
51	95-2-21-72-75-23-67-39-12-76-29-79-33-81-9-65-71-51-30-90-63-64-49-36-47-48-46-8-45-84-5-6-94-96-59-97-13-58-26-74-56-4-25-55-54-24-80-68-77-1-70	476.05	0.00	986.06	800
49	28-27-69-52-82-19-11-62-31-88-7-18-83-61-16-86-38-14-44-85-98-99-92-87-42-43-15-57-41-22-73-40-53-3-78-34-35-66-20-32-10-50-37-93-100-91-17-60-89	473.44	16.59	980.03	658
100		949.49	16.59	1966.09	1458

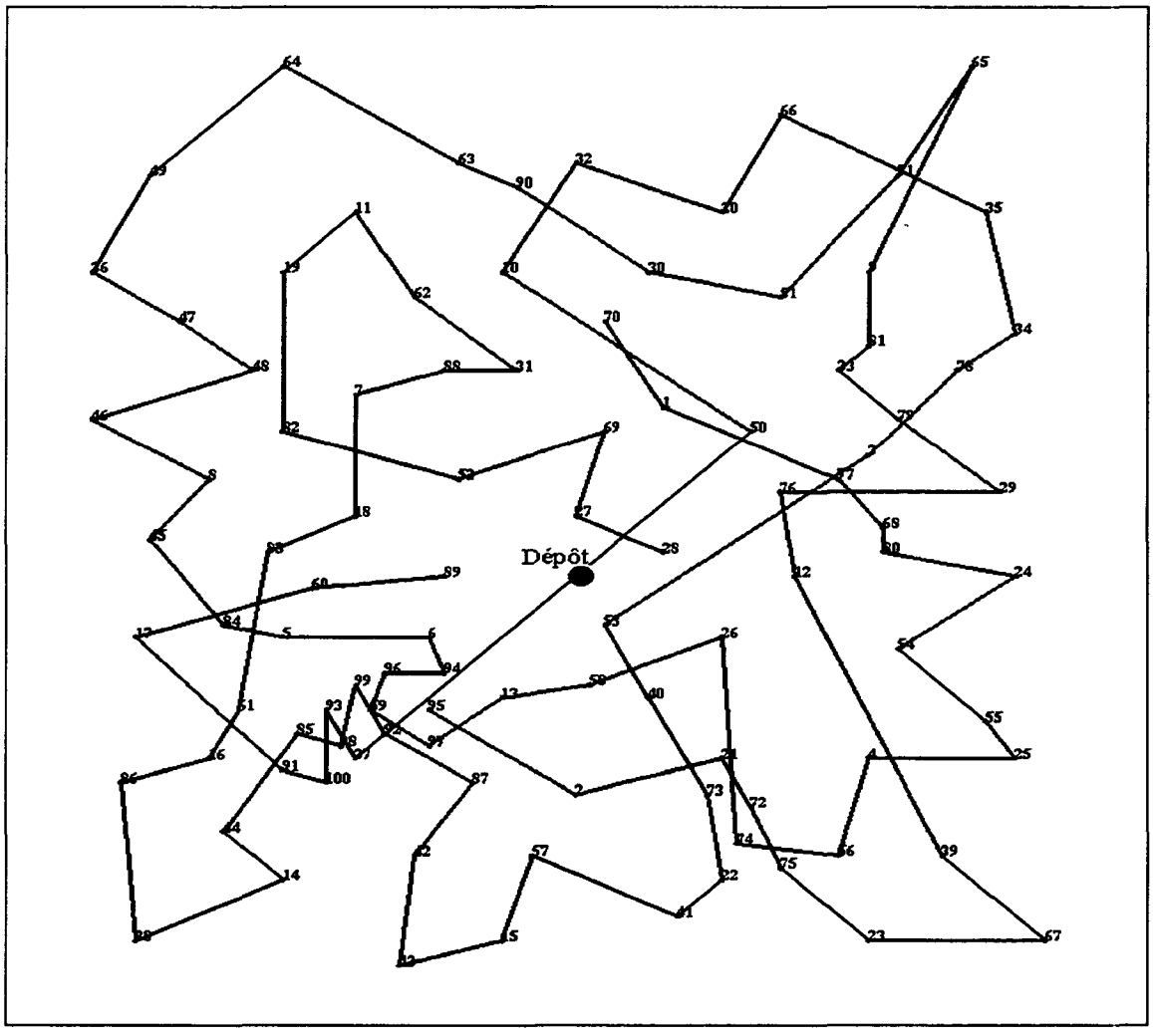


Figure 5.13 : Nouvelle meilleure solution pour R211 : 949.49 avec 2 camions.

Problème C207

Nombre de clients	Tournée	Distance	Attente	Durée	Charge
35	93-5-75-2-1-99-100-97-92-94-95-98-7-3-4-89-91-88-86-84-83-82-85-76-71-70-73-80-79-81-78-77-96-87-90	235.53	0.00	3385.53	620
33	20-22-24-27-30-29-6-32-33-31-35-37-38-39-36-34-28-26-23-17-18-19-16-14-12-15-13-25-9-11-10-8-21	195.09	72.12	3237.21	630
32	67-63-62-74-72-61-64-66-69-68-65-49-55-54-53-56-58-60-59-57-40-44-46-45-51-50-52-47-42-41-43-48	157.67	0.00	3037.67	560
100		588.29	72.12	9660.41	1810

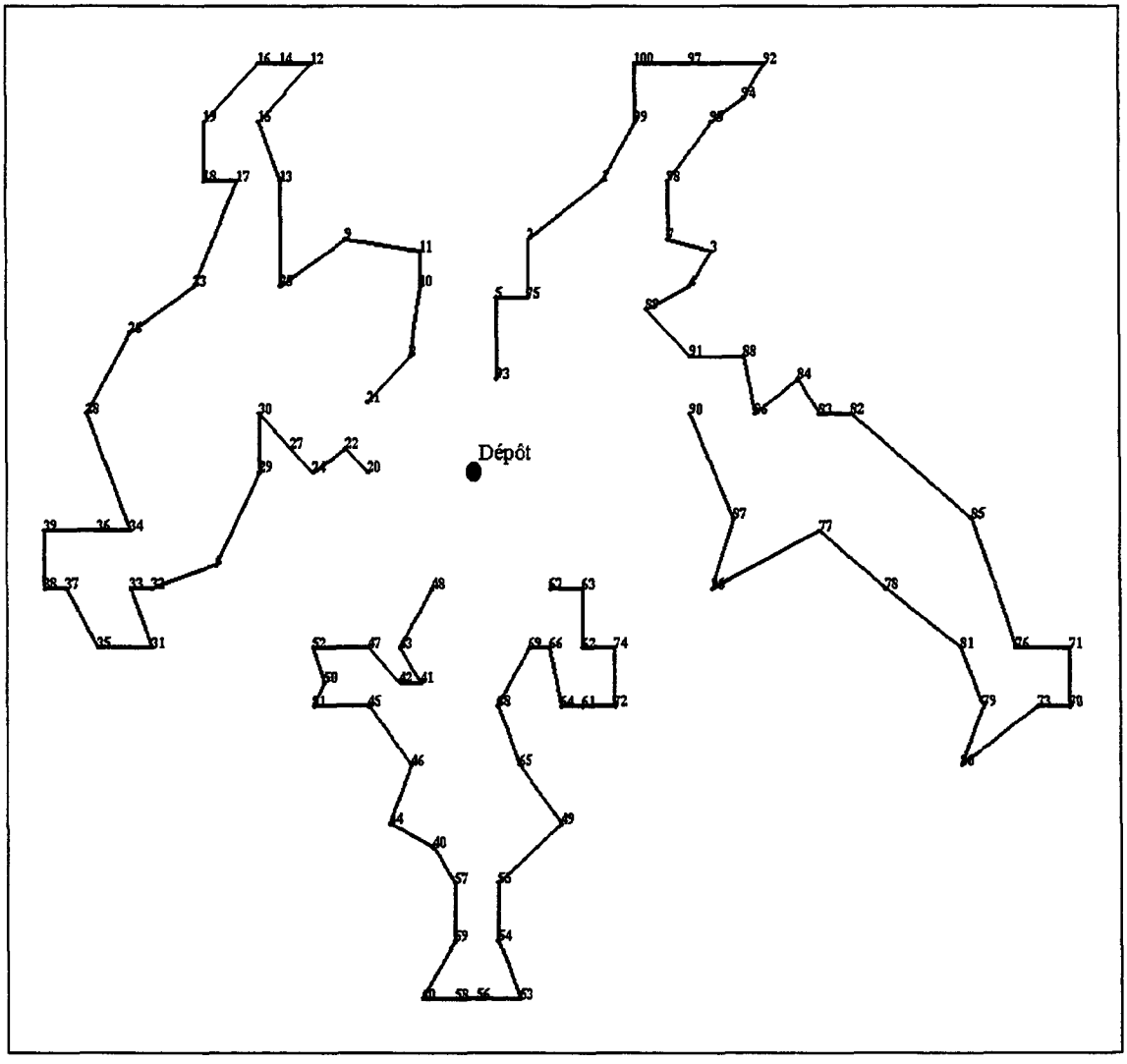


Figure 5.14 : Nouvelle meilleure solution pour C207 : 588.29 avec 3 camions.

Problème RC208

Nombre de clients	Tournée	Distance	Attente	Durée	Charge
44	92-95-64-83-65-82-99-52-11-12-14-47-16-15-9-87-75-58-77-25-23-21-48-18-19-49-20-22-24-57-86-74-59-97-13-10-17-60-46-4-70-100-55-68	381.00	67.35	888.35	793
29	69-98-88-53-78-73-79-7-6-2-8-45-5-3-1-42-44-43-40-36-35-37-38-39-41-72-54-93-96	213.92	179.51	683.44	515
27	90-61-81-71-94-67-62-50-34-31-29-27-26-28-30-32-33-76-89-63-85-51-84-56-66-91-80	239.04	0.00	509.04	416
100		833.97	246.86	2080.83	1724

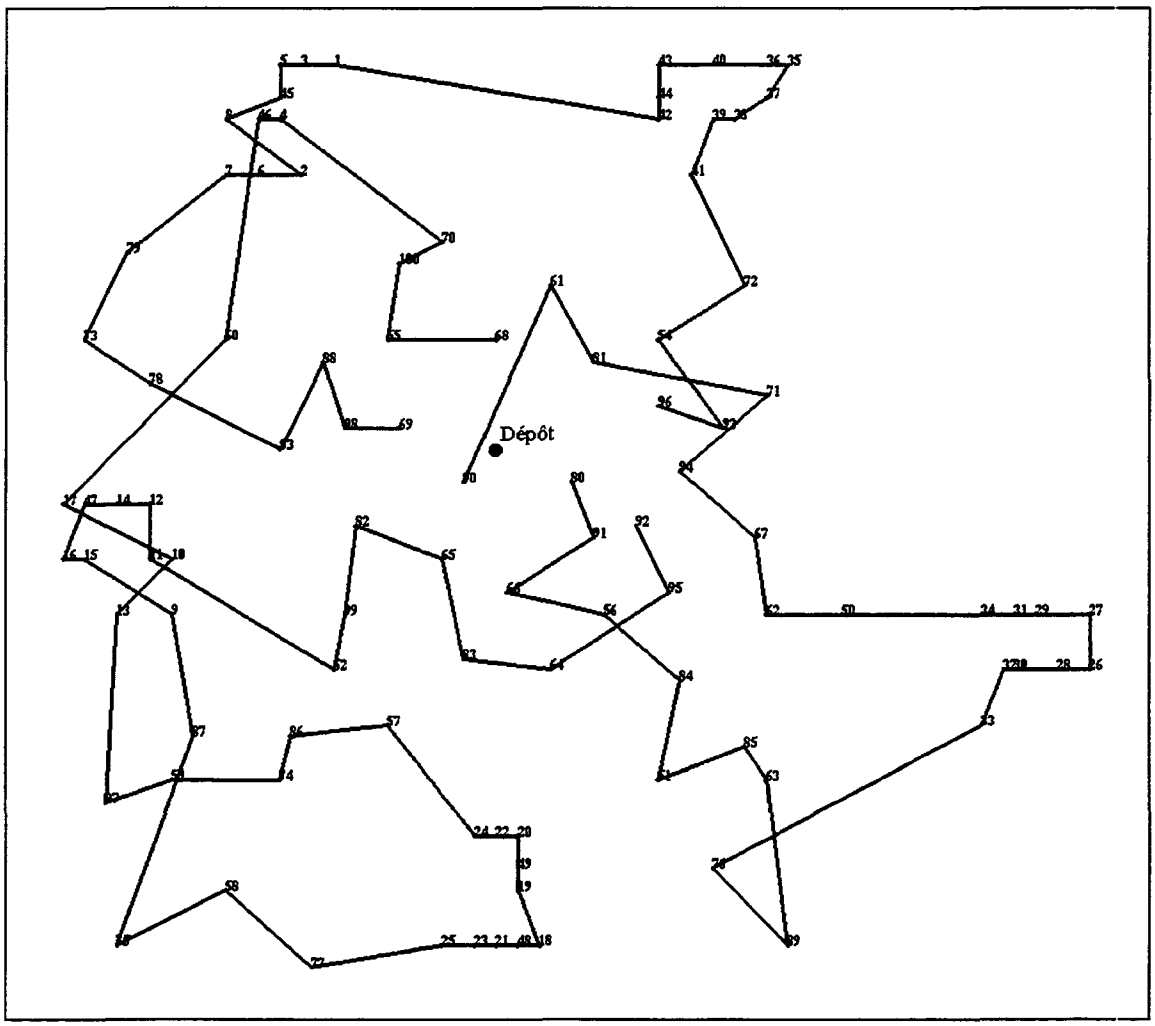


Figure 5.15 : Nouvelle meilleure solution pour RC208 : 833.97 avec 3 camions.

Bibliographie relative au PTV

[Aga89] Agarwal Y., Mathur K. et Salkin H.M. (1989), "A set-partitioning-based algorithm for the vehicle routing problem", *Networks* 19, 731-749.

[Bak86] Baker E. et Schaffer J. (1986), "Computational experience with branch exchange heuristics for vehicle routing problem with time window constraints", *American Journal of Mathematical and Management Sciences* 6, 261-300.

[Bal64] Balinski M. et Quandt R. (1964), "On an integer program for a delivery problem", *Operations Research* 12, 300-304.

[Bea83] Beasley J.E. (1983), "Route first-cluster second methods for vehicle routing", *Omega* 11, 403-408.

[Bod83] Bodin L.D., Golden B.L., Assad A.A. et Ball M.O. (1983), "Routing and scheduling of vehicles and crews : the state of the art", *Computers and Operations Research* 10, 63-211.

[Bov86] Bovet J. (1986), "Une amélioration de la méthode de Dijkstra pour la recherche d'un plus court chemin dans un réseau", *Discrete Applied Mathematics* 13, 93-96.

[Cap91] Capelle R.B. et Chow G. (1991), "U.S. and Canadian statistical sources for transborder trucking information", *Evolution in Transportation*, 215-227.

[Cer85] Cerny V. (1985), "Thermodynamical approach to the traveling salesman problem : an efficient simulation algorithm", *Journal of Optimization Theory and Applications* 45, 41-51.

[Chi93] Chiang W.C. et Russell R.A. (1993), "Simulated annealing metaheuristics for the vehicle routing problem with time windows", Rapport technique, Department of Quantitative Methods, University of Tulsa, Tulsa, États-Unis.

[Chr76] Christofides N. (1976), "The vehicle routing problem", *RAIRO* 10, 55-70.

[Chr79] Christofides N., Mingozzi A. et Toth P. (1979), "The vehicle routing problem" dans *Combinatorial Optimization* , Christofides N., Mingozzi A., Toth P. et Sandi C. (éditeurs), Wiley, Chichester, 315-338.

[Chr81a] Christofides N., Mingozzi A. et Toth P. (1981), "State space relaxation procedures for the computation of bounds to routing problems", *Networks* 11, 145-164.

[Chr81b] Christofides N., Mingozzi A. et Toth P. (1981), "Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations", *Mathematical Programming* 20, 255-282.

[Chr85a] Christofides N. (1985), "Vehicle routing" dans *The Traveling Salesman Problem*, Lawler E.L., Lenstra J.K., Rinnooy Kan A.H.G. et Shmoys D. (éditeurs), Wiley, Chichester, Royaume-Uni, 431-448.

[Chr85b] Christofides N. (1985), "Vehicle scheduling and routing", 12th International symposium on Mathematical Programming, Cambridge.

[Chr89] Christofides N. et Mingozzi A. (1989), "Vehicle routing : practical and algorithmic aspects" dans *Logistics : Where the Ends Have to Meet*, van Rijn C.F.H. (éditeur), Pergamon.

[Cla64] Clarke G. et Wright J. W. (1964), "Scheduling of vehicles from a central depot to a number of delivery points", *Operations Research* 12, 568-581.

[Col91] Colorni A., Dorigo M. et Maniezzo V. (1991), "Distributed optimization by ant colonies", Proceedings of the first European conference on artificial life, Paris, 134-142.

[Col92] Colorni A., Dorigo M. et Maniezzo V. (1992), "An investigation of some properties of an ant algorithm", Proceedings of the second conference on parallel problem solving from nature, Bruxelles, 509-520.

[Cpl93] Cplex Optimisation, Inc. (1993), "Using the CPLEX callable library and CPLEX mixed integer library, Incline Village, Cplex, NV.

[Dan54] Dantzig G.B., Fulkerson D.R. et Johnson S.M. (1954), "Solution of a large-scale travelling salesman problem", *Operations Research* 2, 393-410.

[Dar02] Darwin C.R. (1902), *The Origin of Species by Means of Natural Selection*, Murray J. (éditeur), London.

[Dav87] Davis L. (1987), *Genetic Algorithms and Simulated Annealing*, Pitman, London.

[Des88] Desrochers M., Lenstra J.K., Savelsberg M.W.P. et Soumis F. (1988), "Vehicle routing with time windows : optimization and approximation" dans *Vehicle Routing : Methods and Studies*, Golden B.L. et Assad A.A. (éditeurs), Amsterdam, Pays-Bas, 65-84.

[Des91] Desrochers M. et Verhoog T.W. (1991), "A new heuristic for the fleet size and mix vehicle routing problem", *Computers and Operations Research* 18, 263-274.

[Des92] Desrochers M., Desrosiers J. et Solomon M.M. (1992), "A new optimization algorithm for the vehicle routing problem with time windows", *Operations Research 40*, 342-354.

[Des83] Desrosiers J., Pelletier P. et Soumis F. (1983), "Plus court chemin avec contraintes d'horaires", *RAIRO* 17, 357-377.

[Des84] Desrosiers J., Soumis F. et Desrochers M. (1984), "Routing with time windows by column generation", *Networks* 14, 545-565.

[Des94] Desrosiers J., Dumas Y., Solomon M.M. et Soumis F. (1994), "Time constrained routing and scheduling" dans *Handbooks of Operations Research and Management Science : Network Routing* , Amsterdam, Pays-Bas, 35-139.

[Dro86] Dror M. et Levy L. (1986), "A vehicle routing improvement algorithm : comparison of a greedy and a matching implementation for inventory routing", *Computers and Operations Research* 13, 33-45.

[Eil71] Eilon S., Watson-Gandy C.D.T. et Christofides N. (1971), "Mathematical modelling and practical analysis" dans *Distribution Management*, Griffin, Londres.

[Fai88] Faigle U. et Schraeder R. (1988), "On the convergence of stationary distribution in simulated annealing algorithms", *Information Processing Letters* 27, 189-194.

[Fai92] Faigle U. et Kern W. (1992), "Some convergence results for probabilistic tabu search", *ORSA Journal on Computing* 4, 32-37.

[Fis78] Fisher M.L. (1978), "A decomposition algorithm for large-scale vehicle routing", Rapport 78-11-05, Department of Decision Sciences, University of Pennsylvania, États-Unis.

[Fis81] Fisher M.L., Jaikumar R. (1981), "A generalized assignment heuristic for vehicle routing", *Networks* 11, 109-124.

[Fis94] Fisher M.L. (1994), "Optimal solution of vehicle routing problems using minimum K-trees", *Operations Research* 42, 626-642.

[Fis95] Fisher M.L. (1995), "Vehicle routing" dans *Handbooks on Operations Research and Management Science : Network Routing*, Amsterdam, Pays-Bas, 1-33.

[Gar79] Garey M.R. et Johnson D.S. (1979), *Computers and Intractability : A Guide to the Theory of NP-Completeness*, Freeman & Co. (éditeurs), New York.

[Gas67] Gaskel T.J. (1967), "Bases for vehicles fleet scheduling", *Operational Research Quaterly* 18, 281-295.

[Gen92] Gendreau M., Hertz A. et Laporte G. (1992), "New insertion and post-optimization procedures for the traveling salesman problem", *Operations Research* 40, 1086-1094.

[Gen94] Gendreau M., Hertz A. et Laporte G. (1994), "A tabu search heuristic for the vehicle routing problem", *Management Science* 40, 1276-1290.

[Gen95] Gendreau M., Laporte G. et Potvin J.-Y. (1995), "Local search algorithms for the vehicle routing problem" dans *Local Search Algorithms*, Lenstra J.K. et Aarts E.H.L. (éditeurs), Wiley, Chichester, Royaume-Uni.

[Gil74] Gillett B. et Miller L. (1974), "A heuristic algorithm for the vehicle dispatch problem", *Operations Research* 22, 340-349.

[Glo75] Glover F. (1975), "Surrogate constraint duality in mathematical programming", *Operations Research* 23, 434-451.

[Glo77] Glover F. (1977), "Heuristic for integer programming using surrogate constraints", *Decision Sciences* 8, 1, 156-166.

[Glo86] Glover F. (1986), "Future paths for integer programming and links to artificial intelligence", *Computers and Operations Research* 13, 5, 533-549.

[Glo89] Glover F. (1989), "Tabu Search – Part I", *ORSA Journal on Computing* 1, 190-206.

[Glo90] Glover F. (1990), "Tabu Search – Part II", *ORSA Journal on Computing* 2, 4-32.

[Glo93a] Glover F., Taillard É. et de Werra D. (1993), "A user's guide to tabu search", *Annals of Operations Research* 41, 3-28.

[Glo93b] Glover F. et Laguna M. (1993), "Tabu search" dans *Modern Heuristics Techniques for Combinatorial Problems*, Reeves C.R. (éditeur), Blackwell, Oxford, Royaume-Uni, 70-150.

[Glo93c] Glover F. (1993), "Scatter search and star-paths : beyond the genetic metaphor", Rapport technique, University of Colorado, School of Business, Boulder, États-Unis.

[Glo94] Glover F. (1994), "Genetic algorithms and scatter search : unsuspected potentials", *Statistics and Computing* 4, 131-140.

[Gol85] Golden W.R. et Stewart J. (1985), "Empirical analysis of heuristics" dans *The Traveling Salesman Problem*, Lawler E.L., Lenstra J.K., Rinnooy Kan A.H.G. et Shmoys D. (éditeurs), Wiley, Chichester, Royaume-Uni, 207-249.

[Gol88] Golden B.L. et Assad A.A. (éditeurs) (1988), *Vehicle Routing : Methods and Studies*, Amsterdam, Pays-Bas.

[Hai85] Haimovich M. et Rinnooy Kan A.H.G. (1985), "Bounds and heurisitics for capacitated routing problems", *Mathematics of Operations Research* 10, 527-542.

[Hai88] Haimovich M., Rinnooy Kan A.H.G. et Stougie L. (1988), "Analysis of heuristics for vehicle routing problems" dans *Vehicle Routing : Methods and Studies* , Golden B.L. et Assad A.A. (éditeurs), Amsterdam, Pays-Bas, 47-61.

[Haj85] Hajek B. (1985), "A tutorial survey of theory and applications of simulated annealing", Proceedings of 24th conference on decision and control, Landerdale Ft., 755-760.

[Han86] Hansen P. (1986), "The steepest ascent mildest descent heuristic for combinatorial programming", Congress on numerical methods in combinatorial optimization, Capri, Italie.

[Her92] Hertz A., Taillard É. et de Werra D. (1992), "Tabu search", à paraître dans *Local Search in Combinatorial Optimization*, Lenstra J.K. (éditeur), Rapport ORWP 92/18, Département de Mathématiques, École Polytechnique Fédérale de Lausanne, Suisse.

[Hol75] Holland J.H. (1975), *Adaptation in Natural and Artificial Systems*, Ann Arbor, University of Michigan Press.

[Hoo88] Hooban J.M. (1988), "Marketing a vehicle routing package" dans *Vehicle Routing : Methods and Studies* , Golden B.L. et Assad A.A. (éditeurs), Amsterdam, Pays-Bas, 447-468.

[Kir83] Kirkpatrick S., Gelatt C.D. Jr. et Vecchi M.P. (1983), "Optimization by simulated annealing", *Science* 220, 671-680.

[Lap85] Laporte G., Nobert Y. et Desrochers M. (1985), "Optimal routing under capacity and distance restrictions", *Operations Research* 33, 1050-1073.

[Lap86] Laporte G., Mercure H. et Nobert Y. (1986), "An exact algorithm for the asymmetrical capacitated vehicle routing problem", *Networks* 16, 33-46.

[Lap87] Laporte G. et Nobert Y. (1987), "Exact algorithms for the vehicle routing problem", dans *Surveys in Combinatorial Optimization* , Martello S., Laporte G., Minoux M. et Ribeiro C. (éditeurs), Amsterdam, Pays-Bas.

[Lap92] Laporte G. (1992), "The vehicle routing problem : an overview of exact and approximate algorithms", *European Journal of Operational Research* 59, 345-358.

[Lap95] Laporte G. et Osman I.H. (1995), "Routing problems : a bibliography", *Annals of Operations Research* 61, 227-262.

[Law76] Lawler E. L. (1976), *Combinatorial Optimization : Networks and Matroids*, Helt, Rinehart et Winston (éditeurs), New York.

[Len81] Lenstra J.K. et Rinnooy Kan A.H.G. (1981), "Complexity of vehicle routing and scheduling problems", *Networks* 11, 221-228.

[Lin65] Lin S. (1965), "Computer solutions of the traveling salesman problem", *Bell System Technical Journal* 44, 2245-2269.

[Lin73] Lin S. et Kernighan B.W. (1973), "An effective heuristic algorithm for the TSP", *Operations Research* 21, 498-516.

[Men32] Menger K. (1932), "Das Botenproblem" dans *Ergebnisse eines Mathematischen Kolloquiums* 2, Menger K. (éditeur), Teubner, Leipzig.

[Met53] Metropolis N., Rosenbluth A., Rosenbluth M., Teller A. et Teller E. (1953), "Equation of state calculations by fast computing machines", *Journal of Chemical Physics* 21, 1087-1091.

[Mil60] Miller C., Tucker A. et Zemlin R. (1960), "Integer programming formulations and traveling salesman problems", *J.A.C.M.* 7, 326-329.

[Mol76] Mole R.H. et Jameson S.R. (1976), "A sequential route building algorithm employing a generalized savings criterion", *Operational Research Quaterly* 27, 503-511.

[Or76] Or I. (1976), "Traveling salesman type combinatorial optimization problems and their relation to the logistics of regional blood banking", Ph.D. Thesis, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, États-Unis.

[Osm91] Osman I.H. (1991), "Metastrategy simulated annealing and tabu search algorithms for combinatorial optimization problems", Ph.D. Thesis, Imperial College, The Management School, University of London, Royaume-Uni.

[Osm93a] Osman I.H. (1993), "Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem", *Annals of Operations Research* 41, 421-451.

[Osm93b] Osman I.H. (1993), "Vehicle routing and scheduling : applications, algorithms and developments", Proceedings of the international conference on industrial logistics, Rennes, France.

[Owo92] Owoc M. et Sargious M.A. (1992), "The role of transportation in free trade competition", dans *Canadian Transportation : Competing in a Global Context*, Waters N. (éditeur), Banff, Alberta, Canada, 23-32.

[Pae88] Paessens H. (1988), "The savings algorithms for the vehicle routing problem", *European Journal of Operational Research* 34, 336-344.

[Pot95a] Potvin J.-Y. (1995), "Genetic algorithms for the travelling salesman problem", à paraître dans *Annals of Operations Research.*

[Pot95b] Potvin J.-Y., Dubé D. et Robillard C. (1995), "A hybrid approach to vehicle routing using neural networks and genetic algorithm", à paraître dans *Applied Intelligence.*

[Pot96] Potvin J.-Y. et Bengio S. (1996), "The vehicle routing with time windows – Part II : Genetic", *INFORMS Journal on Computing* 8, 165-172.

[Pur91] Pureza V.M. et França P.M. (1991), "Vehicle routing problems via tabu search metaheuristic", Rapport CRT-747, Centre de recherche sur les transports, Université de Montréal, Canada.

[Roc94] Rochat Y. et Semet F. (1994), "A tabu search approach for delivering pet food and flour in Switzerland", *Journal of the Operational Research Society* 45, 1233-1246.

[Roc95] Rochat Y. et Taillard É. (1995), "Probabilistic diversification and intensification in local search for vehicle routing", *Journal of Heuristics* 1, 147-167.

[Ron88] Ronen D. (1988), "Perspectives on practical aspects of truck routing and scheduling", *European Journal of Operational Research* 35, 137-145.

[Ros77] Rosenkrantz D.J., Stearns R.E. et Lewis P.M. (1977), "An analysis of several heuristics for the traveling salesman problem", *SIAM Journal on computing* 6, 563-581.

[Rus95] Russell R.A. (1995), "Hybrid heuristics for the vehicle routing problem with time windows", *Transportation Science* 29, 156-166.

[Sav85] Savelsbergh M.W.P. (1985), "Local search in routing problems with time windows", *Annals of Operations Research* 4, 285-305.

[Sem93] Semet F. et Taillard É. (1993), "Solving real-life VRP efficiently using tabu search", *Annals of Operations Research* 41, 469-488.

[Sil80] Silver E.A., Vidal R.V. et de Werra D. (1980), "A tutorial on heuristic methods", *European Journal of Operational Research* 5, 153-162.

[Sol83] Solomon M.M. (1983), "The vehicle routing and scheduling problem with time window constraints : models and algorithms", Ph.D. Thesis, Department of Decision Sciences, University of Pennsylvania, Philadelphia, États-Unis.

[Sol86] Solomon M.M. (1986), "On the worst-case a performance of some heuristics for the vehicle routing and scheduling problem with time window constraints", *Networks* 16, 161-174.

[Sol87] Solomon M.M. (1987), "Algorithms for the vehicle routing and scheduling problems with time window constraints", *Operations Research* 35, 254-265.

[Sol88a] Solomon M.M., Baker E. et Schaffer J. (1988), "Vehicle routing and scheduling problems with time window constraints : efficient implementations of solution improvement procedures" dans *Vehicle Routing : Methods and Studies*, Golden B.L. et Assad A.A. (éditeurs), Amsterdam, Pays-Bas, 85-105.

[Sol88b] Solomon M.M. et Desrosiers J. (1988), "Time windows constraints routing and scheduling problems", *Transportation Science* 22, 1-13.

[Tai93a] Taillard É. D. (1993), "Parallel iterative search methods for vehicle routing problems", *Networks* 23, 661-676.

[Tai93b] Taillard É. D. (1993), "Recherches itératives dirigées parallèles", Thèse No. 1153, Département de Mathématiques, École Polytechnique Fédérale de Lausanne, Suisse.

[Tai94] Taillard É. D. (1994), "Comparison of heuristic methods for the quadratic assignment problem", Rapport CRT-989, Centre de recherche sur les transports, Université de Montréal, Canada.

[Tha91] Thangiah S.R., Nygard K.E. et Juell P.L. (1991), "GIDEON : a genetic algorithm system for vehicle routing with time windows", Proceedings IEEE of 7th conference on artificial intelligence applications, IEEE Computer Society Press, Los Alamitos.

[Tha93] Thangiah S.R. (1993), "Vehicle routing with time windows using genetic algorithms", Rapport SRU-SpSc-TR-93-23, Slippery Rock University, Pennsylvanie, États-Unis.

[Tha94] Thangiah S.R., Osman I.H. et Sun T. (1994), "Hybrid genetic algorithm, simulated annealing and tabu search methods for vehicle routing problems with time windows", Rapport UKC/IMS/OR94/4, Institute of Mathematics and Statistics, University of Kent, Canterbury, Royaume-Uni.

[Tho88] Thompson P.M. (1988), "Local search algorithms for vehicle routing and other combinatorial problems", Ph.D. Thesis, Massachusetts Institute of Technology, États-Unis.

[Tot84] Toth P. (1984), "Heuristic algorithms for the VRP", Workshop on routing problems, Hamburg, Allemagne.

[Wil89] Willard J.A.G. (1989), "Vehicle routing using r-optimal tabu search", M. Sc. Dissertation, The Management School, Imperial College, Londres, Royaume-Uni.

[Wre71] Wren A. (1971), *Computers in Transport Planning and Operation*, Ian Allon, Londres.

[Wre72] Wren A. et Holliday A. (1972), "Computer scheduling of vehicles from one or more depots to a number of delivery points", *Operational Research Quaterly* 23, 333-344.

[Yel70] Yellow P. (1970), "A computational modification of the savings method of vehicle scheduling", *Operational Research Quaterly* 21, 281-283.

PARTIE 2

ORDONNANCEMENT DANS UN ENVIRONNEMENT MATÉRIEL AUTOMATISÉ

Introduction à l'ordonnancement

La seconde partie de cette thèse aura pour objet l'étude des problèmes d'ordonnancement relatifs à l'optimisation de traitements d'échantillons dans un laboratoire robotisé d'analyses chimiques. Il s'agira de développer des modèles et des algorithmes performants pour la gestion de tels systèmes. Avant de présenter ce problème en détail dans le chapitre suivant, nous allons rappeler succinctement les éléments importants qui interviennent dans les problèmes d'ordonnancement. Le lecteur intéressé par une présentation détaillée des divers problèmes appartenant à ce domaine est invité à consulter les références suivantes : [Con67], [Bak74], [Cof76], [Len77], [Gra79], [Fre82], [Law82a], [Law82b], [Len85], [Bla87], [Car88], [Got93], [Cra94b] et [Cra95].

Les deux notions fondamentales qui interviennent dans un problème d'ordonnancement sont les *ressources* et les *travaux* (ou *jobs*). Une ressource est un moyen matériel ou humain à disposition pour la réalisation d'un travail. Une ressource possède une certaine disponibilité (limitée ou non) qui est connue a priori. Un travail se définit comme un ensemble de *tâches* ou d'*opérations*, généralement une par ressource, qui doivent être exécutées dans un ordre prédéfini. La réalisation d'une tâche nécessite un certain nombre d'unités de temps (sa *durée de traitement*) et d'unités d'une ou plusieurs ressources. Une ressource est dite *renouvelable* si après avoir exécuté une tâche elle est à nouveau disponible pour les autres. C'est par exemple le cas d'une machine, d'un processeur ou d'un fichier informatique. Dans le cas contraire, par exemple pour les matières premières ou l'argent, on parlera de ressource *consommable*. Une tâche est dite *préemptive* ou *morcelable* si elle peut être exécutée en plusieurs fois. Généralement, les tâches sont liées entre elles par des *contraintes d'antériorité*. Dans le cas contraire, on parlera de tâches *indépendantes*.

La résolution d'un problème classique d'ordonnancement consiste en l'allocation de ressources dans le temps pour exécuter un ensemble de travaux. Habituellement, on rencontre ce type de problème dans le monde industriel où il s'agit de répartir du travail sur les différentes machines d'un atelier de fabrication. Mais, comme le soulignent Carlier et Chrétienne [Car82], les problèmes d'ordonnancement apparaissent dans bien d'autres

domaines tels l'informatique (allocation de processeurs pour l'exécution des programmes), la construction (suivi d'un projet) ou l'administration (gestion de l'emploi du temps). Généralement, la fonction objectif qui permet d'évaluer la qualité d'une solution d'un problème d'ordonnancement fait intervenir des variables telles que la date de fin d'exécution ou le retard d'un travail. Les critères d'optimisation les plus courants sont la durée totale de traitement (connu sous le nom de *makespan* dans la littérature et représentant la date de fin d'exécution du dernier travail réalisé) ou le plus grand retard que l'on cherche à minimiser.

Dans un problème d'ordonnancement cyclique, l'ensemble des travaux est composé de travaux *génériques*, c'est-à-dire de travaux qui donnent naissance chacun à une infinité d'instances. Le problème consiste donc à ordonnancer toutes les instances des travaux génériques. Ce problème, qui n'a attiré que récemment l'attention des chercheurs, admet de nombreuses applications pratiques dans le monde industriel (production en série, ordonnancement de lignes de galvanoplastie) ou informatique (calculs répétitifs). Généralement, le critère d'optimisation d'un tel problème est la maximisation du taux de production ou, ce qui revient au même, la minimisation du temps moyen qui sépare la production successive de deux travaux.

De nombreux articles scientifiques traitant des problèmes d'ordonnancement ont été publiés ces trente dernières années. On pourrait donc être en droit de penser que tout a été fait dans ce domaine. Cependant, en analysant de plus près cette abondante littérature, force est de constater que le sujet qui a attiré le plus les chercheurs jusqu'ici est le problème de la *chaîne de montage* ou *flowshop*. Il s'agit du problème de production de base dans lequel n jobs ou pièces doivent être exécutés suivant le même ordre sur chacune des m machines qui composent l'atelier. Ces pièces ont donc toutes la même gamme opératoire, mais des temps de traitement éventuellement différents. Ce problème est NP-difficile dès que le nombre de ressources est supérieur à 3. Il existe plusieurs variantes de ce problème de production de base comme par exemple le flowshop de permutations dans lequel le même ordre de traitement des pièces sur les machines doit être respecté (aucune pièce ne devance une autre).

Pour un atelier plus général, où les pièces suivent des cheminements différents déterminés uniquement par leur gamme opératoire propre, nous parlons d'un problème de *jobshop* qui est généralement difficile à résoudre (NP-difficile au sens fort [Gar79]). Dans ce cas, la littérature scientifique se fait plus rare. Si l'on considère un problème où l'ordre d'exécution des opérations d'un job est libre, on est en présence d'un problème d'atelier moins contraint que les deux précédemment cités que l'on appelle *openshop*.

Celui-ci modélise par exemple les problèmes de gestion des examens médicaux dans un hôpital (job = patient, opération = examen médical et machine = appareil médical ou salle). Un grand nombre de problèmes d'openshop sont NP-difficiles et par conséquent il n'existe que peu de variantes pour lesquelles des méthodes de résolution polynomiales sont connues.

La plupart des algorithmes et des heuristiques d'ordonnancement développés à ce jour l'ont été pour des applications de l'industrie mécanique, où le fait qu'un job attende avant d'être usiné (stockage avant traitement) ne portait pas à conséquence. Il n'existait que quelques rares situations, par exemple dans la sidérurgie, où un nouveau type de contraintes devait être pris en compte : la contrainte *pas d'attente autorisée* ou *no-wait*. Ce type de contrainte apparaît lorsqu'un job doit être traité sans aucune interruption (que ce soit sur ou entre les diverses machines de l'atelier) entre le début et la fin de sa gamme opératoire. Si cette contrainte n'est pas respectée, la situation n'est généralement pas désespérée car il est souvent possible de retraiter le job concerné. De nos jours, l'intérêt pour les problèmes d'ordonnancement avec contrainte de no-wait et pour des algorithmes capables de gérer ce type de contrainte est de plus en plus grand (voir [Red72], [Goy75], [Pan79], [Sah79], [Pan80], [Roc84], [Goy88], [Kub89] et [Hal93a]).

Par la suite, nous allons nous intéresser exclusivement aux problèmes d'ordonnancement qui apparaissent dans un environnement robotisé. Ce type de problème n'est pas standard car en plus des ressources statiques de l'atelier, il faut gérer un ou plusieurs appareils automatisés de transport (AAT) qui alimentent ces différents éléments statiques. Les modèles classiques d'ordonnancement ignorent la plupart du temps les contraintes imposées par les AAT sur la performance d'ensemble de l'atelier. À titre d'exemple, rappelons que les temps de transport entre les différentes machines d'un atelier sont supposés nuls dans un problème de flowshop. De nos jours, la plupart des opérations de manutention dans un atelier (transfert d'une pièce d'une machine à une autre, chargement et déchargement des pièces sur les machines, gestion des outils) sont réalisées grâce à des AAT tels qu'un robot ou un véhicule autoguidé. Comme ces AAT font partie intégrante de l'atelier, il s'agit dorénavant d'en tenir compte explicitement dans les nouveaux modèles mathématiques d'ordonnancement développés pour ces systèmes modernes de traitement et/ou de fabrication. Différents auteurs se sont récemment intéressés à ces nouveaux problèmes. Citons les travaux de Blazewicz et al. [Bla91], Sethi et al. [Set92], King et al. [Kin93], Hall et al. [Hal93a, Hal93b et Hal93c], Ioachim et Soumis [Ioa93], Blazewicz et Finke [Bla94], Levner et al. [Lev95] et Hertz et al. [Her96b]. Dans le but de présenter quelques résultats récents dans ce domaine, nous définissons dans le chapitre suivant un modèle général de représentation d'un atelier automatisé.

6. État actuel de la recherche

Le problème d'ordonnancement dans un atelier automatisé consiste à construire des ordonnancements pour les jobs et les AAT, de manière à optimiser une quelconque mesure de performance du système de fabrication (makespan, taux de production, temps moyen de production d'un job). L'atelier automatisé est généralement constitué d'une ressource de stockage d'entrée I (appelée Input), d'une ressource de stockage de sortie O (appelée Output), d'un ensemble R = {1, 2, ..., r, ..., m} de m ressources de traitement et de k AAT identiques. On notera J = {1, 2, ..., j, ..., n} l'ensemble des n jobs à traiter. Chaque job doit être exécuté soit une fois soit une infinité de fois selon la situation à considérer. Les n jobs sont initialement disponibles dans l'Input et doivent se retrouver dans l'Output après leur traitement. Chaque ressource peut traiter un seul job à la fois et le temps de traitement du job j sur la ressource r est égal à d_{jr} (j = 1, 2, ..., n ; r = 1, 2, ..., m). Chaque AAT se déplace de façon autonome dans l'atelier le long de rails prévus à cet effet et ne peut transporter qu'un seul job à la fois. Son déplacement d'une ressource r vers une ressource s requiert un temps égal à w_{rs} qu'il soit chargé ou vide. Le rôle d'un AAT dans l'atelier consiste à effectuer les manoeuvres suivantes :

- charger les jobs sur les ressources avant leur traitement,
- décharger les jobs des ressources après leur traitement,
- transporter les jobs d'une ressource à l'autre.

Les sections suivantes présentent succinctement l'état de l'art dans ce domaine.

6.1 Flowshop automatisé

Dans un flowshop automatisé, on suppose que les ressources sont disposées en ligne ou en cercle (sans perte de généralité dans l'ordre I, 1, 2, ..., m, O) et chaque job doit être successivement traité par les machine 1, 2, ..., m. De surcroît, les AAT peuvent seulement se déplacer de la ressource r vers la ressource t en passant devant toutes les ressources intermédiaires. Cette dernière contrainte peut se modéliser en utilisant la définition suivante sur les temps de transport :

$$\boxed{w_{rt} = w_{rs} + w_{st} \quad \text{si} \quad r < s < t}$$

Les travaux de recherche menés dans ce domaine peuvent être classifiés en deux groupes distincts : le flowshop dans une cellule robotisée et le problème d'ordonnancement cyclique dans un environnement automatisé, connu dans la littérature sous le nom de *hoist scheduling problem* (HSP).

6.1.1 Flowshop dans une cellule robotisée

Les problèmes de flowshop dans un atelier robotisé se caractérisent par la présence d'une unique ressource non statique, généralement un robot ou un véhicule autoguidé, qui achemine les jobs entre les différentes ressources de l'atelier. Usuellement, les ressources sont disposées sur un arc de cercle à équidistance les unes des autres et le robot se trouve au centre de ce cercle. Les temps de trajet du robot entre deux ressources consécutives quelconques est égal à δ et les temps de chargement ou de déchargement d'une pièce sur n'importe quelle ressource sont égaux à $\varepsilon > 0$. La Figure 6.1 illustre le cas où $m = 2$, c'est-à-dire lorsque l'atelier est constitué d'une ressource de stockage d'entrée I, de deux ressources de traitement R_1 et R_2 et d'une ressource de stockage de sortie, O. Ce type d'environnement, appelé *cellule robotisée*, se rencontre fréquemment dans les ateliers de fabrication où un certain nombre de ressources desservies par un unique robot sont utilisées pour la production de pièces en série [Asf85]. L'objectif est généralement la détermination d'un ordonnancement qui minimise le temps moyen à long terme de production d'une pièce.

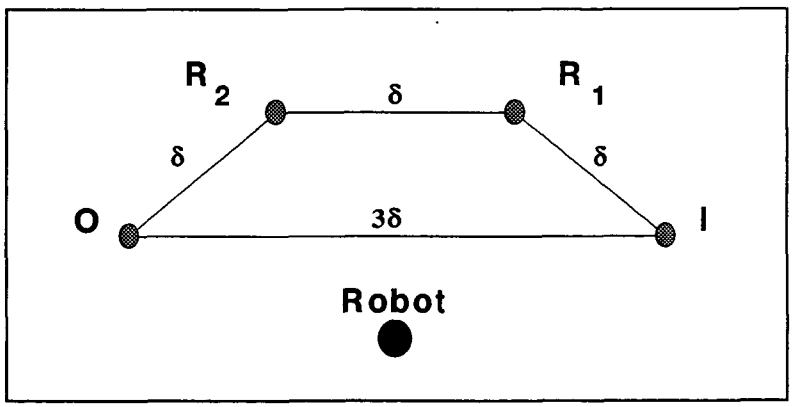


Figure 6.1 : Une cellule robotisée avec m = 2 ressources.

Les premiers travaux concernant ce problème sont dus à Sethi et al. [Set92] ainsi qu'à Kise et al. [Kis91b] lorsque l'objectif est la minimisation du makespan. Ces auteurs étudient des cellules robotisées de petite taille ($m = 2$ ou 3) et la majeure partie de leurs travaux portent sur l'établissement d'une séquence optimale de mouvements du robot, appelée *ℓ-cycle*, que l'on peut reproduire indéfiniment.

Un ℓ-cycle se définit comme une séquence de mouvements du robot dans laquelle chaque ressource est chargée et déchargée exactement ℓ fois, si bien qu'on peut la répéter indéfiniment. Lorsque ℓ = 1, Sethi et al. [Set92] ont montré qu'il existe exactement m! 1-cycle pour une cellule robotisée composée de m ressources de traitement.

Dans le cas où m = 2, les deux cycles possibles C_1 et C_2 se déterminent en étudiant les deux stratégies suivantes pour le déplacement du robot dans la cellule robotisée :

1. Le robot prend une pièce en I, se déplace vers R_1, charge la pièce sur R_1, attend la fin du traitement de la pièce en R_1, récupère la pièce sur R_1, se déplace en R_2, charge la pièce sur R_2, attend la fin du traitement de la pièce en R_2, récupère la pièce sur R_2, se déplace en O, charge la pièce sur O et retourne en I. Les mouvements du robot qui constituent le cycle C_1 sont illustrés dans la Figure 6.2.
2. Le robot prend une pièce (notée P_1) en I, se déplace vers R_1, charge la pièce P_1 sur R_1, attend la fin du traitement de la pièce P_1 en R_1, récupère la pièce P_1 sur R_1, se déplace en R_2, charge la pièce P_1 sur R_2, se déplace vers I, prend une autre pièce (notée P_2) en I, se déplace vers R_1, charge la pièce P_2 sur R_1, se déplace en R_2, si nécessaire attend la fin du traitement de la pièce P_1 en R_2, récupère la pièce P_1 en R_2, se déplace en O, charge la pièce P_1 sur O, se déplace en R_1, si nécessaire attend la fin du traitement de la pièce P_2 en R_1. Les mouvements du robot qui forment un cycle C_2 sont décrits dans la Figure 6.3.

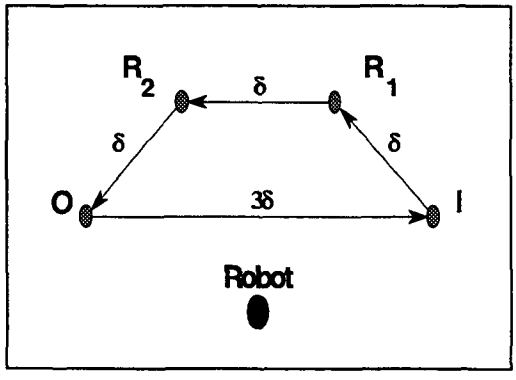


Figure 6.2 : Séquence cyclique C_1.

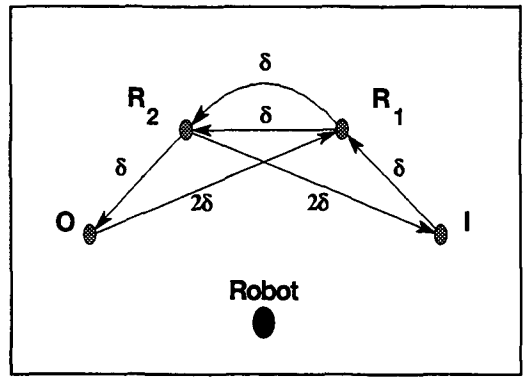


Figure 6.3 : Séquence cyclique C_2.

Toujours dans le cas m = 2, Sethi et al. [Set92] ont montré que, pour chaque 1-cycle, la séquence des jobs (i.e. la permutation de 1, 2, ..., n) qui minimise le temps de cycle à long terme peut se calculer en temps polynomial (O(n·log n)) en se ramenant à un cas spécial du PVC qui se résout avec l'algorithme de Gilmore et Gomory [Gil64]. Un résultat similaire a été obtenu par Kise et al. [Kis91b] lorsque l'objectif est la

minimisation du makespan. Ce problème avec deux ressources a aussi été étudié par Maggu et Dass [Mag80], Logendran et Sriskandarajah [Log92] et Hall et al. [Hal93b]. Notons cependant, comme l'ont observé Hall et al. [Hal93b], que le temps de cycle optimal ne s'obtient pas nécessairement en répétant indéfiniment le même 1-cycle, même lorsque m = 2.

Lorsque m ≥ 3, le flowshop dans une cellule robotisée devient un problème d'ordonnancement difficile à résoudre. Hall et al. [Hal93c] ont démontré que pour m = 3 le problème est NP-difficile au sens fort dès que n ≥ 2, que les mouvements du robot soient restreints à des 1-cycles ou non. Cependant, la complexité de ce problème n'est pas connue lorsque l'on désire produire des pièces identiques (i.e. lorsque n = 1). Sethi et al. [Set92] ont formulé la conjecture suivante : lorsque n = 1, les 1-cycle dominent les ℓ-cycles ($\ell > 1$) dans le sens que le temps de cycle à long terme peut être minimisé en répétant indéfiniment un même 1-cycle. Ces mêmes auteurs ont décrit une méthode optimale de résolution de ce problème pour m = 2. Ce problème a aussi été étudié par Crama et van de Klundert [Cra94a]. Ce dernier auteur a démontré récemment la conjecture précitée dans le cas m = 3 [Klu96].

Terminons cette section en rappelant que des variantes de ce problème de flowshop dans une cellule robotisée ont été étudiées par plusieurs chercheurs. Par exemple lorsque l'on suppose que chaque ressource a la possibilité de stocker un nombre non limité de pièces avant de les traiter, Kise [Kis91a] a montré que ce problème est NP-difficile lorsque m = 2. Dans [Kin93], King et al. proposent un algorithme de séparation et évaluation pour résoudre ce problème. D'autres variantes sont présentées dans les travaux de Matsuo [Mat88], Liu et Lin [Liu93], Kamoun et al. [Kam93] et Agnetis et al. [Agn94].

6.1.2 Hoist Scheduling Problems

Cette section présente le HSP, problème très similaire à celui présenté dans la section précédente mais qui possède les caractéristiques spécifiques suivantes :

- la plus importante est le fait que le temps de traitement d'une pièce sur une ressource n'est plus fixe mais sa valeur doit appartenir à un intervalle de temps défini par une valeur minimale et une valeur maximale,
- plusieurs AAT peuvent être présents dans l'atelier,
- les pièces à traiter sont supposées toutes identiques (i.e. n = 1).

Le problème consiste donc à traiter ou produire une infinité de pièces identiques tout en minimisant le temps moyen à long terme qui sépare la production successive de deux pièces. Dans le contexte industriel, ce type de problème se rencontre dans les systèmes de manutention d'une *ligne de galvanoplastie*. Une telle ligne, appelée aussi *ligne de traitement de surface*, est constituée d'un ensemble de cuves contenant des produits chimiques spécifiques et d'un ou plusieurs robots qui se déplacent sur un rail situé au-dessus des cuves (voir la Figure 6.4 qui est tirée d'un exposé donné par Baptiste et al. [Bap93] lors des journées thématiques de Besançon en 1993). Le rôle du robot est de transporter les plaques à traiter d'une cuve à l'autre. La gamme opératoire est l'ensemble des opérations à effectuer pour traiter une plaque, ce qui revient à indiquer une suite de cuves de traitement ainsi que les temps minimum et maximum d'immersion pour chaque cuve.

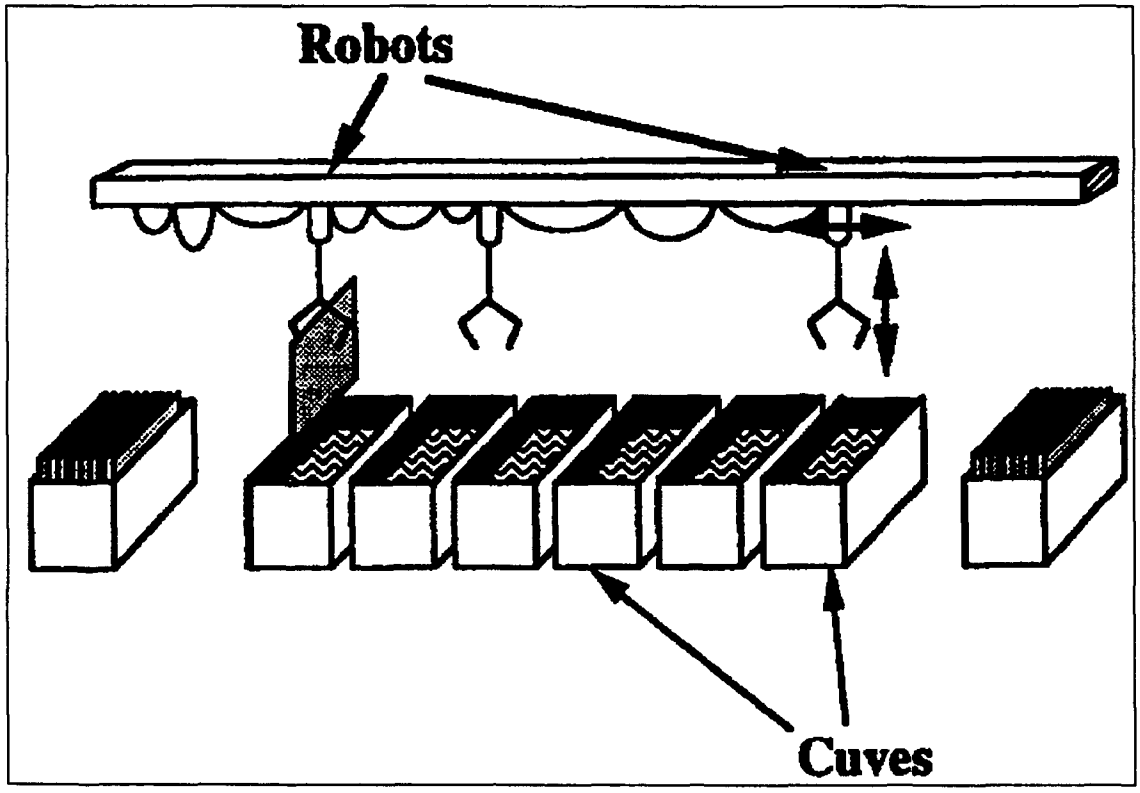


Figure 6.4 : Une ligne de galvanoplastie.

On s'intéresse généralement au problème du pilotage cyclique d'une telle ligne de galvanoplastie qui survient lorsque l'on traite un grand nombre de plaques ayant la même gamme opératoire. Pour ce problème, la recherche d'un ordonnancement périodique de période minimale pour un seul robot est déjà NP-difficile [Lei89]. Les premiers auteurs à s'intéresser à ce problème ont été Phillips et al. [Phi76] et Shapiro et al. [Sha88]. Les modèles et algorithmes de résolution qu'ils ont proposés, basés sur la programmation

linéaire mixte, ne sont pas spécifiques au problème considéré. Par contre, d'autres auteurs ont étudié la structure particulière sous-jacente des solutions de ce type de problèmes pour en tirer des algorithmes exacts (de type séparation et évaluation) ou des heuristiques. Citons les travaux de Matsuo et al. [Mat87], Baptiste et al. [Bap92], Hanen et Munier [Han93], Armstrong et al. [Arm94] et Lei et Wang [Lei94]. Dans le cas où plusieurs AAT sont utilisés simultanément dans l'atelier, diverses études ont été menées par Lei et Wang [Lei91], Hanen et Munier [Han94] et Manier et al. [Man94].

6.2 Flowshop automatisé avec ressources parallèles

Dans ce type de problèmes, les ressources sont supposées identiques, chaque pièce doit être traitée par une seule ressource et les AAT sont soit des robots soit des véhicules autoguidés. Ce genre de situation se rencontre fréquemment dans les ateliers flexibles de fabrication [Ste85]. Lorsqu'un unique AAT est présent dans l'atelier, citons les travaux de Jeng et al. [Jeng93] et Hall et al. [Hal94]. Dans le cas où l'atelier est constitué de plusieurs AAT opérant simultanément, Blazewicz et al. [Bla91] ont mené une étude sur un problème inspiré d'un cas réel. Récemment, une généralisation de ce problème a été proposée par Blazewicz et al. [Bla94].

6.3 Ordonnancement de traitements chimiques dans un laboratoire automatisé

Dans le problème qui va nous intéresser dans la seconde partie de cette thèse, plusieurs passages d'un même job peuvent être prévus sur des mêmes ressources (par exemple lorsqu'une séquence partielle de la gamme opératoire est répétée). C'est une caractéristique peu fréquente des problèmes classiques d'ordonnancement qu'il faudra intégrer dans les modèles mathématiques que nous développerons. Les interruptions entre les ressources sont autorisées mais leur durée est bornée supérieurement par des valeurs connues a priori. Toute interruption dont la durée serait supérieure à ces bornes peut avoir des conséquences catastrophiques, tant au niveau économique qu'humain, car les produits que nous traiterons sont des produits chimiques sensibles qui ne peuvent pas être retraités. Ce type de contrainte est donc une généralisation de la contrainte no-wait qu'il s'agira de prendre en compte dans les algorithmes que nous développerons.

De plus, les ressources peuvent traiter plusieurs jobs simultanément et les durées de traitement des tâches sur les ressources ne sont pas fixes mais, comme pour le HSP, bornées inférieurement et supérieurement. A notre connaissance, il n'existait pas de méthode satisfaisante pour résoudre ce nouveau type de problème.

7. Traitements analytiques d'échantillons dans un laboratoire chimique robotisé

Ce chapitre traite de l'étude des problèmes d'ordonnancement relatifs à l'optimisation de traitements analytiques d'échantillons dans un laboratoire robotisé, ci-après désignés par POTELR. Cette étude a été motivée par une application réelle : la préparation d'échantillons pour l'identification des bactéries par leurs acides gras membranaires. Les points suivants seront notamment abordés :

- modélisation et formulation du problème en termes mathématiques,
- développement d'une heuristique séquentielle constructive,
- mise au point d'une méthode d'amélioration basée sur la recherche d'un plus long chemin dans un graphe.

Nous considérons un laboratoire chimique automatisé qui effectue des traitements analytiques sur un ensemble de n échantillons identiques d'une application chimique donnée. Ce laboratoire se compose d'une ressource particulière, le robot, ainsi que d'un ensemble formé de ressources pour le traitement des échantillons. L'objectif de notre étude est de déterminer un ordonnancement pour n échantillons identiques d'une application chimique donnée lorsque ce nombre n n'est pas connu a priori. Nous devons donc développer des modèles et des algorithmes performants pour planifier intelligemment les déplacements du robot dans le laboratoire chimique afin de minimiser la durée de traitement de l'ensemble de ces n échantillons, tout en respectant un certain nombre de contraintes. Comme nous l'avons déjà mentionné dans le chapitre d'introduction sur les ordonnancements, notre POTELR n'appartient pas à une classe de problèmes d'ordonnancement traitée dans la littérature. La prise en compte simultanée des quatre points suivants nous a conduit à développer de nouveaux modèles mathématiques :

- plusieurs passages d'un même échantillon peuvent être prévus sur des mêmes ressources (par exemple lorsqu'une séquence partielle du traitement analytique de l'application chimique considérée est répétée),

- la capacité des ressources du laboratoire peut être supérieure à 1 si bien que plusieurs échantillons peuvent être traités simultanément dans la même ressource,
- les temps d'exécution des diverses tâches chimiques d'une application ne sont pas fixés à l'avance mais ils doivent être compris dans un intervalle borné inférieurement et supérieurement par des valeurs spécifiques à chaque tâche,
- les temps séparant la fin d'une tâche chimique et le début d'une autre (non nécessairement consécutive) pour le même échantillon sont bornés supérieurement par des valeurs connues a priori.

Notre POTELR peut être considéré comme une généralisation des problèmes de flowshop automatisé. En effet, le flowshop dans une cellule robotisée où n = 1 est un cas particulier de notre problème lorsque la capacité des ressources est égale à 1, la durée des tâches chimiques est fixe et aucun échantillon ne peut être traité plusieurs fois par la même machine. Rappelons que la complexité de ce problème n'est pas connue dès que le nombre de ressources de traitement est supérieur à 3. D'un autre point de vue, notre problème peut aussi être vu comme une généralisation du HSP avec un robot, problème pour lequel la recherche d'un ordonnancement périodique de période minimale est NP-difficile [Lei89]. Parmi les nombreuses techniques de résolution développées à ce jour il n'existe pas, à notre connaissance, d'algorithmes qui permettent de résoudre le POTELR de façon satisfaisante.

Dans la section 7.1, nous décrivons dans le détail le POTELR. Un algorithme heuristique d'ordonnancement est présenté puis évalué respectivement dans les sections 7.2 et 7.3. Des conclusions ainsi que des extensions possibles du modèle que nous avons développé sont discutées dans la dernière section. Il est important de noter que les méthodes qui font l'objet de ce chapitre ont été réellement implantées dans la pratique et ont prouvé leur efficacité.

7.1 Description du problème

Un laboratoire chimique robotisé est un ensemble matériel (généralement constitué d'un ordinateur, d'un robot et de diverses *ressources*) permettant d'effectuer une *application* automatisée sur des échantillons chimiques ou bactériologiques. Les éléments fondamentaux qui interviennent dans un problème d'ordonnancement dans un tel environnement robotisé sont donc les ressources, le robot, l'application à réaliser ainsi que la fonction objectif à optimiser.

7.1.1 Les ressources

Une ressource est définie comme un poste de traitement (machines, instruments, automates) ou de stockage des échantillons dans le laboratoire. Nous considérons un ensemble $R = \{r_0, r_1, \ldots, r_u, \ldots, r_m, r_{m+1}\}$ de ressources où r_0 et r_{m+1} sont respectivement les ressources de stockage d'entrée et de sortie des échantillons dans le laboratoire. Elles correspondent à l'Input et l'Output du modèle général de représentation d'un atelier automatisé (voir chapitre 6). Les ressources r_u, $u = 1, \ldots, m$, sont les postes de traitement où se déroulent les opérations chimiques effectuées sur les échantillons. A chacune de ces ressources est associée une fonction chimique unique correspondant à l'un des rôles caractéristiques joué par une ressource dans le laboratoire chimique (incubation, agitation, etc.). De plus, chaque ressource r_u, $u = 0, \ldots, m+1$, du laboratoire a une capacité limitée $c_u \geq 1$, c'est-à-dire qu'elle ne peut pas traiter simultanément plus de c_u échantillons. Nous supposons que les capacités des ressources de stockage sont supérieures ou égales au nombre d'échantillons à traiter, c'est-à-dire $c_0 \geq n$ et $c_{m+1} \geq n$.

Pour pouvoir utiliser une ressource du laboratoire, il faut l'activer. Cette activation peut être soit *explicite* soit *implicite*. Une activation implicite signifie que le traitement chimique sur un échantillon débute aussitôt après son introduction dans une ressource. C'est le cas, par exemple, lorsque un échantillon est introduit dans un bain. Dans une ressource à activation explicite comme par exemple une centrifugeuse, le traitement chimique ne commence que lorsque la ressource a été enclenchée, par un déclenchement manuel ou par l'intermédiaire d'un ordinateur. Pour les ressources de ce type qui auraient une capacité supérieure à un, nous supposons que chaque place de traitement de la ressource peut être activée ou désactivée de manière indépendante.

Certaines ressources ont la particularité supplémentaire d'être *bloquantes*. C'est une caractéristique spécifique d'une ressource qui peut être soit vraie (ressource bloquante) soit fausse (ressource non bloquante). Une ressource est dite bloquante si le robot est bloqué lors du traitement d'un échantillon dans l'une de ses places de traitement. Dans ce cas, le robot ne peut pas traiter un autre échantillon avant la fin de la tâche de l'échantillon considéré. Cette situation peut se présenter, par exemple, lorsqu'un réactif doit être ajouté dans une éprouvette. Dans ce cas, le robot doit placer l'éprouvette dans un distributeur de réactifs et enlever le bouchon de l'éprouvette qui reste dans son bras articulé. Il ne pourra traiter aucune autre éprouvette avant d'avoir replacé le bouchon sur l'éprouvette dans laquelle un réactif a été ajouté.

La Figure 7.1 illustre une configuration matérielle classique d'un laboratoire chimique robotisé.

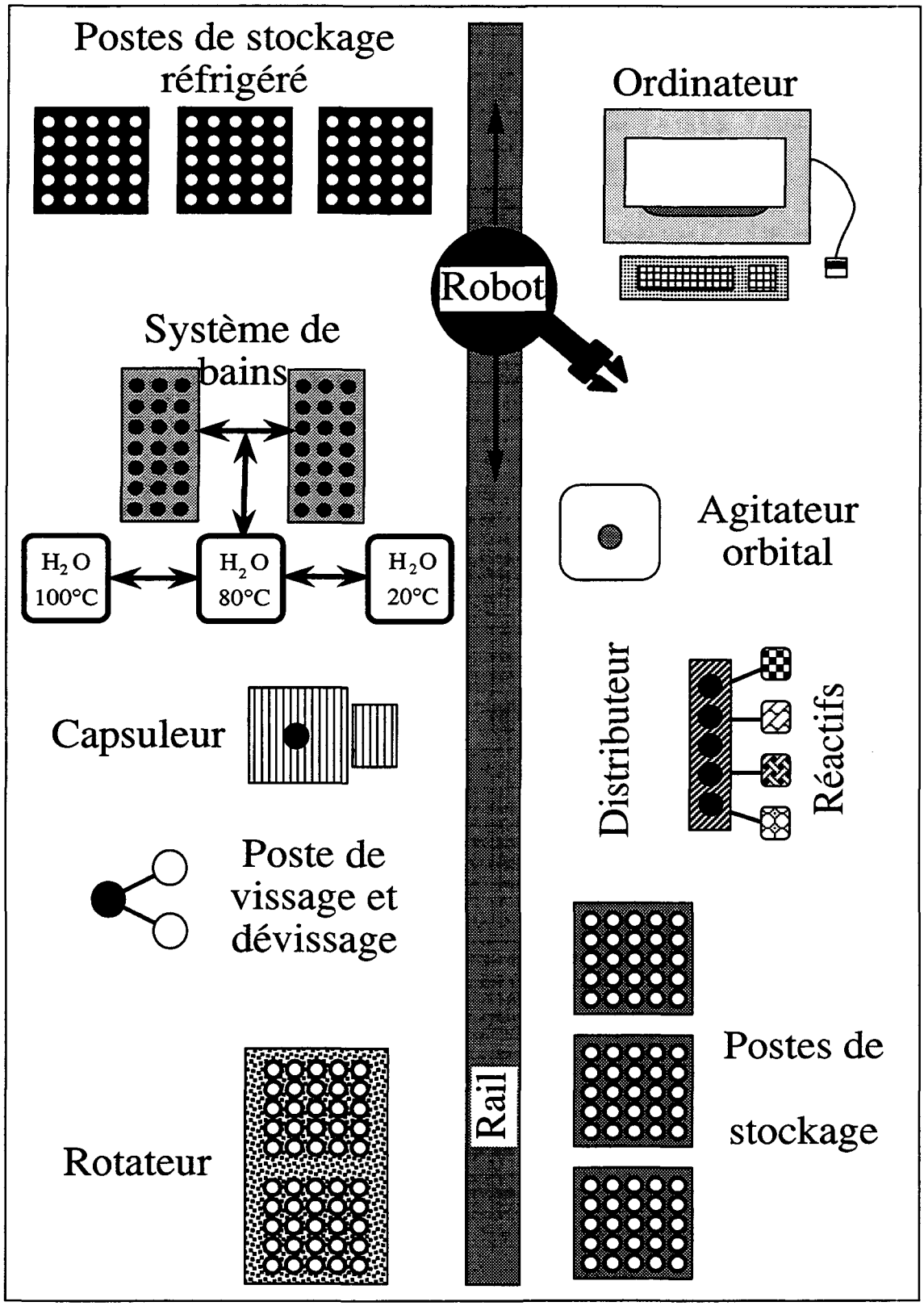


Figure 7.1: Un laboratoire chimique robotisé.

7.1.2 Le robot

En plus de cet ensemble R de ressources statiques, un robot peut se déplacer de façon autonome dans le laboratoire le long d'un rail prévu à cet effet. Son rôle consiste à transporter les échantillons dans le laboratoire d'une ressource à une autre ainsi qu'a effectuer les manoeuvres de chargement et de déchargement des échantillons sur ces ressources. Il ne peut transporter qu'un seul échantillon à la fois et le temps de déplacement du robot d'une ressource r_u vers une ressource r_s est égal à $w_{r_u r_s} \geq 0$.

7.1.3 Application

Dans le cadre de notre étude, on définit une application comme la préparation automatisée d'un type d'échantillons chimiques ou bactériologiques dans un laboratoire chimique robotisé. Une application se caractérise donc par un *processus analytique* et un certain nombre, n, d'échantillons identiques. Un processus analytique est une séquence ordonnée $\mathcal{T} = \{t_0, t_1, \ldots, t_i, \ldots, t_T\}$ de T+1 traitements chimiques, appelés *tâches*, qui doit être exécutée pour chacun des n échantillons de l'application considérée. Un échantillon est donc une entité qui subit des tâches (dans les ressources de traitement) lors de l'exécution d'une application et qui est déplacée par le robot. Sans perte de généralité, nous supposons qu'initialement :

- tous les échantillons se trouvent dans la ressource de stockage d'entrée r_0,
- le robot, positionné en face de la ressource r_0, est prêt à transporter n'importe lequel des n échantillons de r_0 vers une autre ressource du laboratoire.

Le stockage des échantillons dans la ressource r_0 est considéré comme la tâche t_0, dont la durée peut être limitée dans le temps, du processus analytique à exécuter. La tâche t_T de ce processus, dont la durée n'est pas limitée dans le temps, correspond au stockage des échantillons dans la ressource r_{m+1}. Chaque autre tâche t_i, $1 \leq i \leq T-1$, est une action réalisée par une ressource de traitement r_u, $1 \leq u \leq m$. Cette ressource r_u sera notée r(i) pour indiquer que c'est la ressource du laboratoire qui peut effectuer la fonction chimique requise par t_i. Cette action comprend les opérations chimiques, biologiques, mécaniques ou d'analyse que subit un échantillon. On peut voir une tâche chimique comme un certain intervalle de temps qu'un échantillon doit passer dans une certaine ressource du laboratoire. La part de ce temps durant laquelle cette ressource est active est appelée la *période active* de la tâche chimique. Celle-ci ne doit pas être interrompue et sa durée, appelée *temps de traitement*, n'est pas fixée à l'avance mais bornée par des valeurs minimale et maximale spécifiques à chaque tâche.

La Figure 7.2 représente un processus analytique composé de 21 tâches.

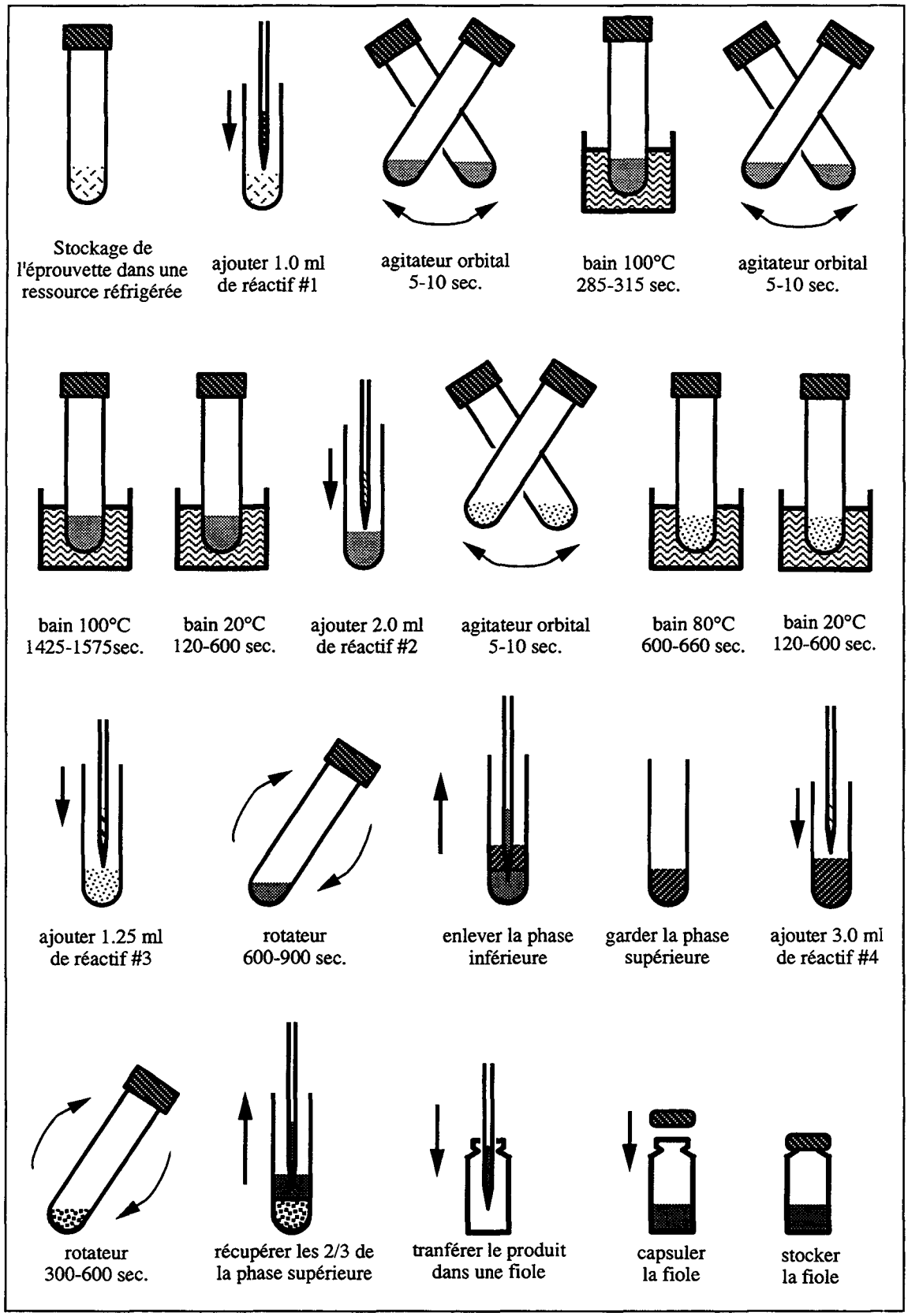


Figure 7.2 : Un exemple de processus analytique.

Considérons la situation où la tâche t_i d'un échantillon α se déroule dans une ressource r(i). Selon le type d'activation de r(i), il y a deux façons de procéder avec α lorsque la durée de la période active de t_i atteint la valeur maximale autorisée. Si r(i) est une ressource à activation explicite, elle peut être désactivée et α restera dans r(i) jusqu'à ce que le robot le déplace dans une autre ressource. Dans le cas contraire, le robot doit pouvoir retirer α de r(i) immédiatement sinon une des contraintes du POTELR est violée. Il se pourrait que la ressource r(i+1) dans laquelle α doit subir sa prochaine tâche t_{i+1} ne soit pas disponible pour des raisons de capacité. Le robot doit donc, dans un tel cas, déposer α dans une ressource de stockage intermédiaire pour autant qu'il en existe une. Dans notre étude, nous considérerons que de telles ressources de stockage ne sont pas disponibles. Par conséquent, si un échantillon α doit être ôté immédiatement d'une ressource r(i), le nombre d'échantillons en cours de traitement dans r(i+1) doit être strictement inférieur à $c_{r(i+1)}$.

Pour certaines paires (t_i, t_j) de tâches chimiques il est imposé, pour chaque échantillon, que le temps pouvant s'écouler entre la fin de la période active de t_i et le début de la période active de t_j (t_j n'est pas nécessairement un successeur immédiat de t_i) ne doit pas être supérieur à une valeur maximale donnée. Au-delà de cette valeur, aucune attente n'est autorisée. Rappelons que dans le cas de notre problème, toute attente peut avoir des conséquences catastrophiques tant au niveau économique qu'humain, par exemple si les produits traités sont des vaccins. Comme le font remarquer Goyal et Sriskandarajah [Goy88], les problèmes d'ordonnancement avec des contraintes de no-wait sont, exception faite de quelques cas spéciaux, des problèmes NP-difficiles. D'un certain côté, notre POTELR est plus général qu'un tel problème car la contrainte de no-wait peut être considérée comme un cas spécial où pour chaque paire (t_i, t_{i+1}) de tâches consécutives il est imposé que la tâche t_{i+1} de chaque échantillon doit commencer immédiatement après la fin de t_i. D'un autre côté, nous devons résoudre un problème d'ordonnancement où les échantillons sont tous identiques et par conséquent, la taille des données permettant de décrire un POTELR ne dépend pas du nombre n d'échantillons à traiter.

7.1.4 Objectif

Le but de notre étude est de déterminer une planification intelligente et efficace des traitements chimiques pour n échantillons d'une application robotisée donnée, lorsque ce nombre n n'est pas connu à l'avance. Il s'agit donc de planifier les déplacements du robot dans un laboratoire automatisé afin de minimiser le makespan, c'est-à-dire la durée totale de traitement de l'ensemble des n échantillons. Toutes les contraintes doivent être satisfaites car même une légère modification du processus analytique pourrait avoir des graves conséquences.

7.1.5 Un POTELR avec une ressource de traitement (m=1)

Considérons le POTELR illustré dans la Figure 7.3 où l'on considère qu'il n'y a qu'une ressource de traitement (m=1) dans le laboratoire. Chaque échantillon α se trouve initialement dans la ressource de stockage d'entrée r(0) où se déroule sa tâche t_0 et le robot est positionné en face de r(0). Chaque échantillon doit ensuite être déplacé par le robot dans la ressource r(1) où sera exécuté son unique tâche chimique, t_1. La ressource r(1) a une activation implicite et la durée d de la tâche t_1 de chaque échantillon α doit être comprise entre une borne inférieure $\underline{d}$ et une borne supérieure $\overline{d}$. Finalement, chaque échantillon α doit se retrouver dans la ressource de stockage de sortie r(2) après son traitement dans r(1). Les temps de déplacement du robot $w_{r(i),r(j)}$ ($0 \leq i, j \leq 2$) entre les ressources r(i) et r(j) sont symétriques. Cette application robotisée simple est constituée de n échantillons identiques auxquels on doit faire subir le processus analytique décrit ci-dessus. L'objectif est de minimiser l'heure à laquelle le dernier échantillon de l'application entre dans la ressource r(2).

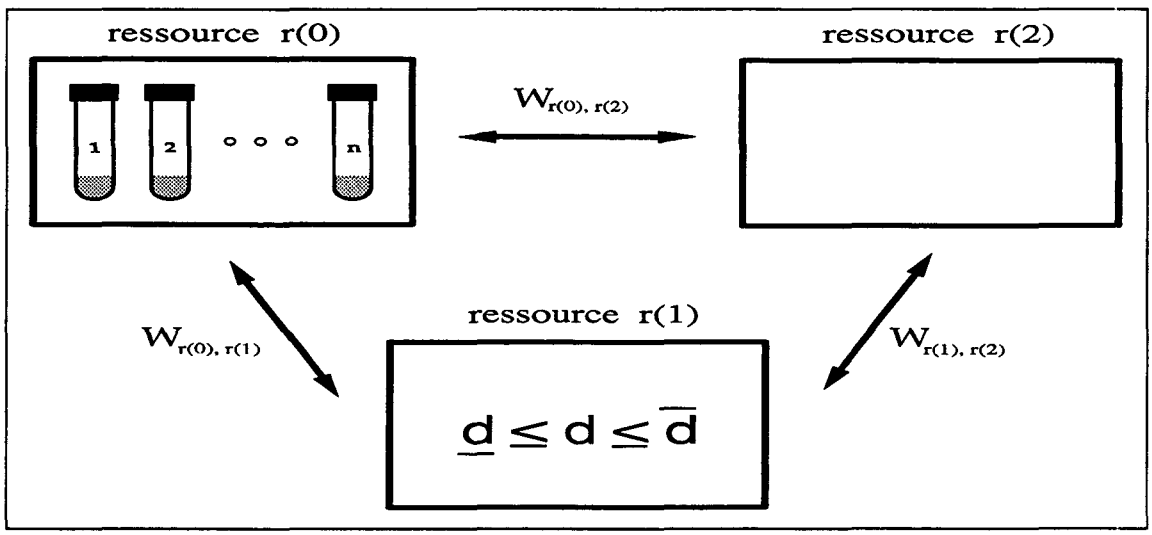


Figure 7.3 : Un POTELR avec une ressource de traitement.

Ce problème peut se décrire avec six nombres : n, $\underline{d}$, $\overline{d}$, $w_{r(0),r(1)}$, $w_{r(1),r(2)}$ et $w_{r(0),r(2)}$. A notre connaissance, la complexité de ce problème est ouverte. Nous ne savons d'ailleurs pas non plus s'il existe un algorithme, polynomial en n, pour le résoudre. Dans le but de convaincre le lecteur que ce problème est bien moins trivial qu'il n'y paraît à première vue, supposons dans un premier temps (comme c'est le cas dans la littérature classique) que la capacité de la ressource r(1) est égale à 1. Dans cette situation, un ordonnancement optimal consiste simplement à traiter complètement chaque échantillon l'un après l'autre. Le makespan de cet ordonnancement optimal S est le suivant :

$$\text{makespan}(S) = n \cdot (w_{r(0),r(1)} + \underline{d} + w_{r(1),r(2)}) + (n-1) \cdot w_{r(0),r(2)}$$

Comme exemple, considérons l'exemple de ce problème illustrée dans la Figure 7.4 où $w_{r(0),r(1)} = 2$, $w_{r(1),r(2)} = 3$, $w_{r(0),r(2)} = 4$, $n = 4$, $\underline{d} = 10$ et $\overline{d} = 15$. L'ordonnancement optimal que nous obtenons (voir Figure 7.5) a un makespan égal à 72 unités de temps.

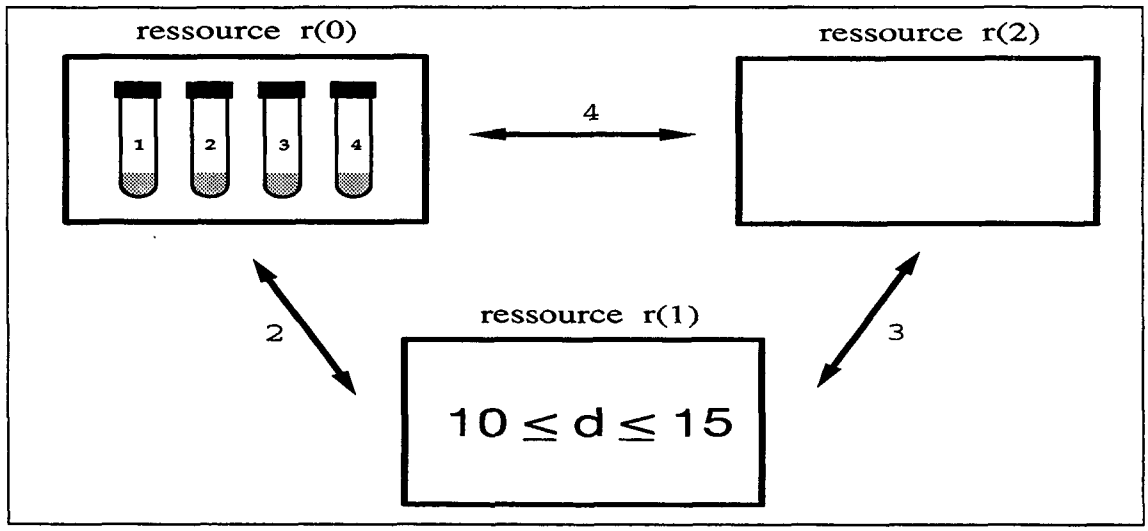


Figure 7.4 : Un exemple du POTELR avec une ressource de traitement.

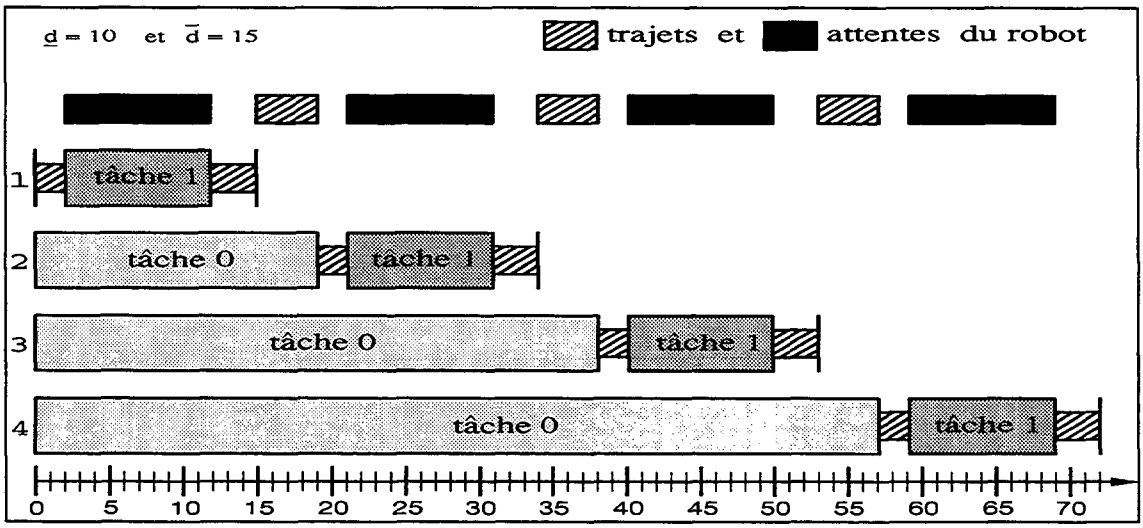


Figure 7.5 : Un ordonnancement en 72 unités de temps.

Supposons maintenant que la capacité de r(1) est supérieure ou égale à n et que les durées minimale $\underline{d}$ et maximale $\overline{d}$ de la tâche t_1 sont égales à 10 (c'est-à-dire que la durée d est fixe). Au lieu d'attendre $\underline{d}$ unités de temps en face de r(1), le robot peut dorénavant déplacer des échantillons additionnels de r(0) vers r(1). La première idée intuitive qui vient à l'esprit pour construire un ordonnancement dans cette situation est d'introduire les échantillons dans r(1) aussitôt que possible. Une fois qu'un échantillon est introduit dans r(1), le robot doit se déplacer vers r(0) puis revenir vers r(1) pour introduire un échantillon additionnel dans r(1). Ces déplacements prennent $2 \cdot w_{r(0),r(1)}$

unités de temps. De surcroît, une fois qu'un échantillon quitte r(1), le robot doit aller vers r(2) puis revenir près de r(1) pour ôter un échantillon supplémentaire de r(1). Par conséquent, comme la durée de t_1 est fixe, au moins $2 \cdot w_{r(1),r(2)}$ unités de temps doivent séparer deux introductions d'un échantillon dans r(1). En résumé, le temps qui sépare deux introductions consécutives d'un échantillon dans r(1) doit être au moins deux fois aussi important que la valeur maximale entre $w_{r(0),r(1)}$ et $w_{r(1),r(2)}$.

En considérant le même exemple du problème que précédemment, un premier échantillon est introduit dans r(1) au temps $2 = w_{r(0),r(1)}$. Le second entre dans r(1) au temps $8 = 2 + 2 \cdot w_{r(1),r(2)}$. Comme le premier échantillon doit quitter r(1) au temps $12 = 2 + \underline{d}$, aucun autre échantillon ne peut être introduit dans r(1) à ce moment car $12 - 8 = 4 < 6 = 2 \cdot w_{r(1),r(2)}$. Le robot va par conséquent attendre 4 unités de temps près de r(1) puis il va déplacer successivement les échantillons 1 et 2 dans r(2). L'ordonnancement complet pour les $n = 4$ échantillons est illustré dans la Figure 7.6. Son makespan est de 46 unités de temps.

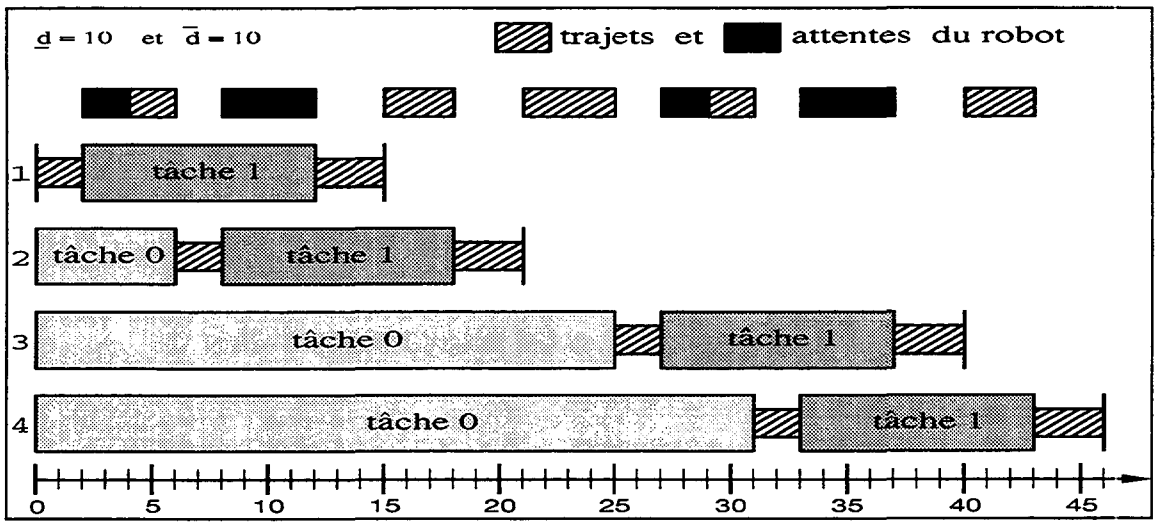


Figure 7.6 : Un ordonnancement en 46 unités de temps.

Notons que la stratégie consistant à introduire les échantillons aussitôt que possible dans r(1) n'est souvent pas la bonne façon de procéder. En effet, cela vaut parfois la peine d'attendre un petit peu avant d'introduire un échantillon supplémentaire dans r(1). Par exemple pour l'exemple du problème où $\underline{d} = \bar{d} = 10$, il n'est pas difficile de montrer que l'ordonnancement optimal est obtenu en introduisant le second échantillon dans r(1) au temps 11 au lieu de 8. Bien que trois unités de temps soient perdues à ce moment, le makespan de l'ordonnancement complet, représenté dans la Figure 7.7, est de 43 unités de temps. En effet, comme dans ce cas au moins 9 unités de temps séparent deux introductions d'un échantillon dans r(1), le robot a suffisamment de temps pour

introduire un nouvel échantillon dans r(1) entre deux déplacements consécutifs d'un échantillon de r(1) vers la ressource de stockage de sortie r(2) car $w_{r(1),r(2)} + w_{r(0),r(2)} + w_{r(0),r(1)} = 9$.

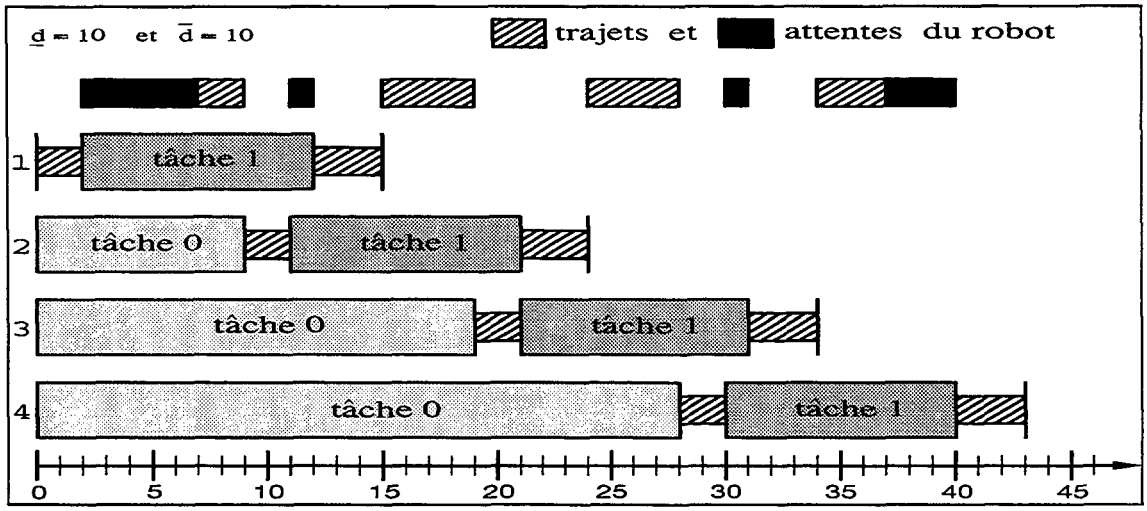


Figure 7.7 : Un ordonnancement en 43 unités de temps.

Finalement, supposons que $\underline{d} \leq \overline{d}$ et que la capacité de r(1) est toujours supérieure ou égale à n. Dans l'exemple du problème que l'on considère (voir Figure 7.4), la durée de la tâche t_1 n'est pas fixée précisément mais doit être comprise entre $\underline{d} = 10$ et $\overline{d} = 15$. Les ordonnancements représentés dans la Figure 7.6 et la Figure 7.7 sont toujours admissibles. Cependant, il est possible de construire un meilleur ordonnancement que ceux-ci car le fait que la tâche t_1 puisse durer plus que $\underline{d} = 10$ unités de temps apporte une certaine flexibilité au problème. En effet, l'ordonnancement illustré dans la Figure 7.8 a un makespan égal à 36 unités de temps et nous pouvons démontrer facilement qu'il est optimal.

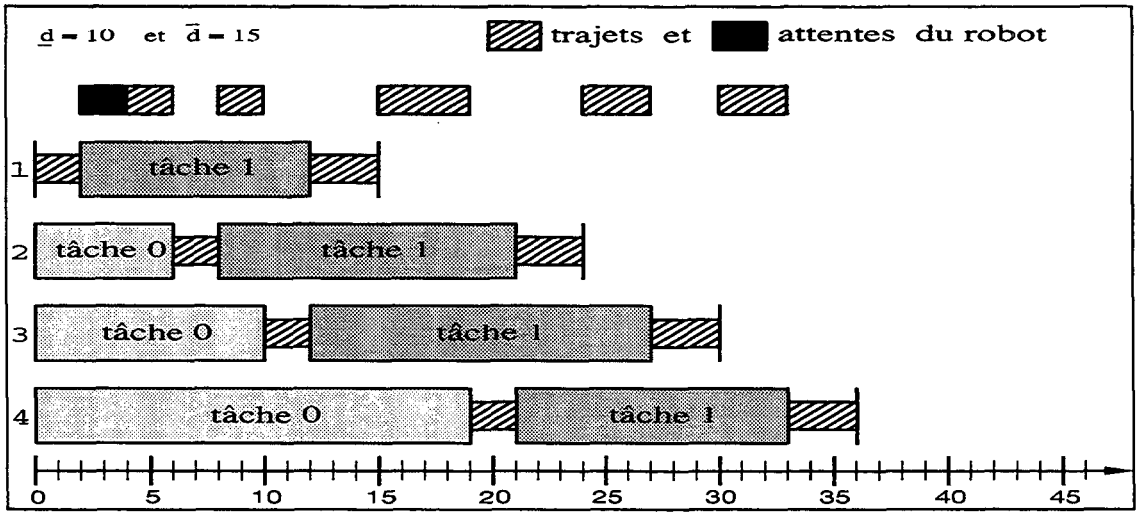


Figure 7.8 : Un ordonnancement en 36 unités de temps.

Comme on peut l'observer dans la Figure 7.8, les temps de traitement des échantillons 1, 2, 3 et 4 dans r(1) sont respectivement de 10, 13, 15 et 12 unités de temps. Par conséquent, trois échantillons ont passé plus de $\underline{d} = 10$ unités de temps dans r(1) mais l'ordonnancement obtenu est de meilleur qualité que les précédents. Son makespan est d'ailleurs 2 fois plus petit que celui de l'ordonnancement correspondant à la meilleure solution du problème classique étudié dans la littérature où les capacités des ressources sont égales à 1 et $\underline{d} = \overline{d}$.

A ce stade de notre étude, nous espérons avoir convaincu le lecteur que le POTELR est un problème d'ordonnancement moins trivial qu'il n'y paraît à première vue. De plus, notons que l'ordonnancement qui consiste à traiter complètement chaque échantillon l'un après l'autre n'est pas forcément admissible car la durée de la tâche t_0 (temps passé dans la ressource de stockage d'entrée) peut, éventuellement, être limitée dans le temps. Sur la base de ces remarques et d'une étude approfondie de la littérature, nous formulons la conjecture suivante :

Conjecture

La détermination d'une solution admissible pour le problème d'ordonnancement des traitements analytiques d'échantillons dans un laboratoire robotisé est un problème NP-difficile.

Nous avons donc développé une méthode heuristique pour résoudre le POTELR. Cette heuristique, présentée en détail dans la prochaine section, sera désignée dans la suite de ce texte par CASSIRAS (de l'anglais Constructive Algorithm for Scheduling Samples In a Robotized Analytical System).

7.2 Un algorithme heuristique itératif pour la résolution du POTELR

7.2.1 Généralités et notations

L'étape fondamentale d'une procédure itérative consiste à construire une nouvelle solution s' du problème étudié à partir de la solution courante s et à déterminer si le processus doit s'arrêter à ce stade ou s'il faut exécuter une nouvelle itération. Les méthodes de recherche d'une solution par examen d'un voisinage (par exemple le recuit simulé et la méta-heuristique Tabou présentés dans la section 2.3.2.2) sont des

procédures où un voisinage N(s) est défini pour toute solution s et la nouvelle solution s' est choisie parmi les solutions contenues dans N(s). De telles méthodes ont été appliquées avec succès à des problèmes d'ordonnancement très variés. Ces applications sont décrites, par exemple, dans [Wid89], [Laa92], [Bar93], [Del93], [Her96a] et [Now93]).

Pour adapter une technique de recherche par examen de voisinage à un problème spécifique d'optimisation combinatoire, deux notions fondamentales doivent être définies : l'ensemble S des solutions qui seront explorées au cours de la recherche et le voisinage N(s) de chaque solution s de S. Dans le cadre de notre problème, nous ne pouvons pas définir S comme l'ensemble des ordonnancements admissibles (c'est-à-dire des ordonnancements qui vérifient l'ensemble des contraintes du problème) puisque, suite à la conjecture que nous avons énoncée, la recherche d'un ordonnancement admissible est peut-être un problème NP-difficile. Par conséquent, l'ensemble S des solutions devrait contenir des ordonnancements qui violent certaines contraintes. Nous pourrions, par exemple, décider de relâcher la contrainte sur la durée des périodes actives des tâches chimiques et de pénaliser chaque violation enregistrée. Mais, étant donné une nouvelle séquence des opérations du robot, il serait bien difficile dorénavant de décider combien de temps devrait durer chacune de ces tâches pour minimiser une mesure quelconque des violations des contraintes. Il apparaît donc que les techniques de recherche par examen de voisinage sont difficilement applicables au POTELR. C'est la raison pour laquelle nous avons développé un algorithme heuristique d'ordonnancement qui est basé sur une méthode constructive itérative.

Au début de l'algorithme, les n échantillons à traiter se trouvent dans la ressource de stockage d'entrée r_0 = r(0) et le robot est positionné devant r(0), prêt à déplacer un échantillon vers la ressource r(1) du laboratoire. À un pas donné du processus d'élaboration de l'ordonnancement des mouvements du robot, un ordonnancement partiel S, planifié jusqu'à une heure H, existe. Celui-ci, appelé *ordre courant des opérations*, est constitué des instructions données au robot pour déplacer certains échantillons d'une ressource vers une autre. Cet ordonnancement partiel se développera lors des étapes suivantes du processus. Ce dernier s'arrête lorsqu'une configuration finale est atteinte, c'est-à-dire quand les n échantillons se trouvent dans la ressource finale r(m+1). Une telle configuration finale ne peut être obtenue que si le POTELR comporte des solutions admissibles.

À chaque pas intermédiaire de l'algorithme, l'ordre courant des opérations du robot induit un certain stade d'avancement du traitement chimique de chaque échantillon.

Les échantillons dont le traitement chimique est terminé sont dit *passifs*, les autres sont considérés comme *actifs*. De plus, un échantillon qui ne se trouve ni dans r(0) ni dans r(m+1) est dit *en traitement*. Pour chaque échantillon actif du système on calcule, dès son entrée dans une ressource, les heures de sortie *au plus tôt* et *au plus tard* de cette ressource qu'il faut absolument respecter sous peine de violer les contraintes chimiques du processus. Ces deux valeurs sont déterminées sur la base des temps de traitements ainsi que des délais maximaux qui peuvent séparer l'exécution des différentes tâches chimiques d'un même échantillon.

À chaque pas de l'algorithme, il faut décider quelle sera la prochaine tâche à être exécutée, c'est-à-dire qu'il faut choisir quel échantillon actif va être déplacé lors du prochain déplacement du robot. Pour cela, les mouvements suivants du robot doivent être planifiés. Il faut décider quand le robot va quitter sa position actuelle pour atteindre la ressource où se trouve l'échantillon que l'on veut déplacer, à quel moment il va retirer l'échantillon de cette ressource et quand il va le déposer dans la ressource où sa prochaine tâche sera effectuée. Cette suite de déplacements du robot est appelée une *opération.*

Tous les ordonnancements partiels que nous considérerons tiennent compte des capacités des ressources et des particularités de certaines d'entre elles comme le fait d'être bloquantes. Par exemple, si le robot se trouve devant une ressource bloquante dans laquelle il vient d'introduire un échantillon, il devra obligatoirement attendre la fin du traitement de cet échantillon. Ensuite, sa prochaine opération consistera nécessairement à déplacer ce même échantillon dans la ressource qui exécutera la prochaine tâche chimique de cet échantillon. Notons de plus, que nous n'envisagerons jamais de déplacer un échantillon dans une ressource où toutes les places de traitement sont occupées. Toutes les opérations du robot considérées pour construire un ordonnancement sont donc des opérations *admissibles*, c'est-à-dire que la ressource qui doit recevoir l'échantillon transporté par le robot a encore au moins une place de traitement de libre.

Soit S un ordonnancement partiel. Dans la suite de ce texte, nous utiliserons les notations suivantes :

- r(i) : la ressource dans laquelle la tâche t_i doit être exécutée,
- R(S) : la ressource devant laquelle se trouve positionné le robot,
- $w_{r(i),r(j)}$: le temps nécessaire au robot pour se déplacer de la ressource r(i) vers la ressource r(j),

- $x_{\alpha,i}$ (respectivement $y_{\alpha,i}$) : l'heure à laquelle l'échantillon α entre (respectivement sort) de la ressource r(i),
- $b_{\alpha,i}$ (respectivement $e_{\alpha,i}$) : l'heure à laquelle la période active de la tâche t_i commence (respectivement se termine) pour l'échantillon α,
- $\underline{d}_i$ (respectivement $\overline{d}_i$) : la durée minimale (respectivement maximale) de la période active de la tâche t_i. Nous supposons que $\underline{d}_0 = 0$ et $\overline{d}_T = \infty$,
- $t_\alpha(S)$: indice ($0 \leq t_\alpha(S) \leq T$) de la tâche chimique en cours d'exécution sur l'échantillon α ,
- s(S) : l'échantillon que le robot a déplacé lors de sa dernière opération,
- H(S) : l'heure à laquelle le robot est prêt pour exécuter sa prochaine opération. Cette heure est égale à $x_{\alpha,t_\alpha(S)}$ où $\alpha = s(S)$,
- $\underline{h}_\alpha$ (respectivement $\overline{h}_\alpha$) : l'heure au plus tôt (respectivement au plus tard) à laquelle peut se terminer la période active de la tâche $t_{t_\alpha(S)}$,
- $L_{i,j}$: la durée maximale autorisée pouvant séparer la fin de la période active de la tâche t_i du début de la période active de la tâche t_j ($0 \leq i < j \leq T$),
- S_α : l'ordonnancement obtenu à partir de S en déplaçant l'échantillon α vers la ressource $r(t_\alpha(S) + 1)$.

7.2.2 Les contraintes du problème

Considérons un ordonnancement partiel S et supposons que la prochaine opération du robot consiste à déplacer l'échantillon actif α de la ressource $r(t_\alpha(S))$ vers la ressource $r(t_\alpha(S)+1)$. Afin de construire l'ordonnancement partiel S_α, deux mouvements du robot doivent être planifiés. Nous devons décider quand il quittera R(S) pour se déplacer vers $r(t_\alpha(S))$, quand il ôtera α de $r(t_\alpha(S))$ et à quel moment il introduira α dans $r(t_\alpha(S)+1)$.

Comme le robot doit se déplacer de R(S) vers $r(t_\alpha(S))$ avant de pouvoir retirer l'échantillon α, l'heure $y_{\alpha,\, t_\alpha(S)}$ doit être plus grande ou égale à $H(S) + w_{R(S),\, r(t_\alpha(S))}$. Posons $\gamma = s(S)$, $i = t_\gamma(S)$ et $j = t_\alpha(S)$. La dernière opération réalisée par le robot a donc été le placement de l'échantillon γ dans r(i) et elle s'est déroulée à l'heure $H(S) = x_{\gamma,i}$. Comme maintenant l'échantillon α doit être retiré de r(j), la contrainte suivante doit être respectée :

$$\boxed{y_{\alpha,j} \geq x_{\gamma,i} + w_{r(i),r(j)}}$$

Le temps qui sépare le retrait de α de r(j) et son placement dans r(j+1) doit être supérieur ou égal au temps de déplacement du robot de r(j) à r(j+1), c'est-à-dire :

$$x_{\alpha,j+1} - y_{\alpha,j} \geq w_{r(j),r(j+1)}$$

Si la première partie de cette inégalité est plus grande que la seconde de Δt unités de temps, cela signifie que le robot est utilisé comme une ressource de stockage temporaire (l'échantillon α reste dans le bras du robot) pendant ce laps de temps. Ce type de situation peut se produire lorsque α doit être retiré immédiatement de r(j) et que son introduction dans r(j+1) ne doit pas avoir lieu trop rapidement.

En résumé, lorsque nous considérons deux opérations consécutives du robot (la première étant le déplacement de l'échantillon γ dans la ressource r(i) où sa tâche t_i sera exécutée et la seconde consistant à transporter l'échantillon α de la ressource r(j) vers la ressource r(j+1) où sa tâche t_{j+1} se déroulera), la contrainte suivante doit être respectée :

$$(1) \quad x_{\gamma,i} + w_{r(i),r(j)} \leq y_{\alpha,j} \leq x_{\alpha,j+1} - w_{r(j),r(j+1)}$$

Comme nous l'avons déjà signalé, la période active d'une tâche t_i ($1 \leq i \leq T-1$) d'un échantillon α ne correspond pas nécessairement à l'intervalle $[x_{\alpha,i}, y_{\alpha,i}]$ en entier. Les contraintes suivantes doivent être prises en considération :

$$(2) \quad x_{\alpha,i} \leq b_{\alpha,i}$$
$$(3) \quad e_{\alpha,i} \leq y_{\alpha,i}$$
$$(4) \quad b_{\alpha,i} + \underline{d}_i \leq e_{\alpha,i} \leq b_{\alpha,i} + \bar{d}_i$$

Dans le cas où la ressource r(i) serait à activation implicite, les inégalités (2) et (3) se transforment en égalités. Le respect de la durée maximale autorisée entre la fin de la période active d'une tâche t_i pour un échantillon α et le début de la période active d'une tâche t_j ($j > i$) pour ce même échantillon s'exprime par la contrainte suivante :

$$(5) \quad b_{\alpha,j} \leq e_{\alpha,i} + L_{i,j}$$

Définition

Un ordonnancement partiel est admissible si et seulement si les contraintes (1) à (5) sont vérifiées.

Condition nécessaire d'admissibilité pour un POTELR

Une condition nécessaire pour qu'un exemple d'un POTELR ait une solution admissible est que la condition suivante soit vérifiée :

$$\sum_{k=i}^{j-1} w_{r(k),r(k+1)} + \sum_{k=i+1}^{j-1} \underline{d}_k \leq L_{i,j}, \quad 0 \leq i < j \leq T$$

En effet, entre la fin de la période active de sa tâche t_i et le début de la période active de sa tâche t_j ($i < j$), un échantillon α doit être introduit successivement dans les ressources r(i+1), ..., r(j). Le temps de transport nécessaire à ces déplacements du robot est donc au moins égal à la somme des temps de transport entre chaque ressource utilisée. De plus, chaque tâche t_k qu'il subit dans la ressource r(k), $i+1 \leq k < j$, doit durer au moins $\underline{d}_k$ unités de temps. La condition précitée est donc bien nécessaire. ❐

Un ordonnancement partiel S est complètement déterminé par ce qui suit :

- l'ensemble $\{t_\alpha(S) \mid \alpha = 1, \ldots, n\}$ des indices des tâches en cours d'exécution pour les n échantillons à traiter,
- les heures $x_{\alpha,i}$, $b_{\alpha,i}$, $e_{\alpha,i}$ et $y_{\alpha,i}$ pour $\alpha = 1, \ldots, n$ et $i = 1, \ldots, t_\alpha(S) - 1$,
- les heures $y_{\alpha,0}$ pour les α tels que $t_\alpha(S) > 0$,
- les heures $x_{\alpha,0}$ pour $\alpha = 1, \ldots, n$.

Initialement, nous avons $t_\alpha(S) = 0$ et $x_{\alpha,0} = 0$ ($\alpha = 1, \ldots, n$) car nous supposons que tous les échantillons ont été placés dans r(0) à l'heure 0.

Proposition

Soient S un ordonnancement partiel admissible, α un échantillon actif quelconque et $k = t_\alpha(S)$ l'indice de sa tâche en cours d'exécution. Les heures au plus tôt, $\underline{h}_\alpha$, et au plus tard, $\overline{h}_\alpha$, auxquelles l'échantillon α peut quitter la ressource r(k) sont :

$$\underline{h}_\alpha = \max\{x_{\alpha,k} + \underline{d}_k, H(S) + w_{R(S),r(k)}\}$$
$$\overline{h}_\alpha = \min\{H_1, H_2\}$$
avec $$H_1 = \min_{i<k}(e_{\alpha,i} + L_{i,k+1}) - w_{r(k),r(k+1)}$$
$$H_2 = \begin{cases} x_{\alpha,k} + \overline{d}_k & \textbf{si r(k) a une activation implicite} \\ \min_{i<k}(e_{\alpha,i} + L_{i,k}) + \overline{d}_k + L_{k,k+1} - w_{r(k),r(k+1)} & \textbf{sinon} \end{cases}$$

Démonstration

Sur la base des contraintes (2) à (4), nous avons :

$$x_{\alpha,k} + \underline{d}_k \leq b_{\alpha,k} + \underline{d}_k \leq e_{\alpha,k} \leq y_{\alpha,k}$$

Pour pouvoir retirer α de r(k) le robot doit d'abord se déplacer de l'endroit où il se trouve, R(S), vers r(k). Il s'en suit que $y_{\alpha,k} \geq H(S) + w_{R(S),r(k)}$. Par conséquent, l'heure au plus tôt $\underline{h}_\alpha$ à laquelle α peut être retiré de r(k) est :

$$\underline{h}_\alpha = \max\{x_{\alpha,k} + \underline{d}_k, H(S) + w_{R(S),r(k)}\}$$

Soit t_i une tâche telle que $i < k$. En vertu des contraintes (1), (2) et (5) nous avons les inégalités suivantes : $y_{\alpha,k} + w_{r(k),r(k+1)} \leq x_{\alpha,k+1} \leq b_{\alpha,k+1} \leq e_{\alpha,i} + L_{i,k+1}$. Par conséquent, nous devons avoir :

$$y_{\alpha,k} \leq H_1 = \min_{i<k}(e_{\alpha,i} + L_{i,k+1}) - w_{r(k),r(k+1)}$$

Sur la base des contraintes (2), (3) et (4), nous savons que si r(k) a une activation implicite alors $y_{\alpha,k} \leq x_{\alpha,k} + \bar{d}_k$. Si par contre l'activation de r(k) est explicite, la situation est plus compliquée. A partir des contraintes (4) et (5) nous obtenons les inégalités suivantes :

$$e_{\alpha,k} - \bar{d}_k \leq b_{\alpha,k} \leq e_{\alpha,i} + L_{i,k}, \quad \forall\, i < k$$
$$b_{\alpha,k+1} \leq e_{\alpha,k} + L_{k,k+1}$$

Finalement, nous devons avoir (voir Figure 7.9) :

$$\begin{aligned} y_{\alpha,k} &\leq x_{\alpha,k+1} - w_{r(k),r(k+1)} \\ &\leq b_{\alpha,k+1} - w_{r(k),r(k+1)} \\ &\leq \min_{i<k}(e_{\alpha,i} + L_{i,k}) + \bar{d}_k + L_{k,k+1} - w_{r(k),r(k+1)} \end{aligned}$$

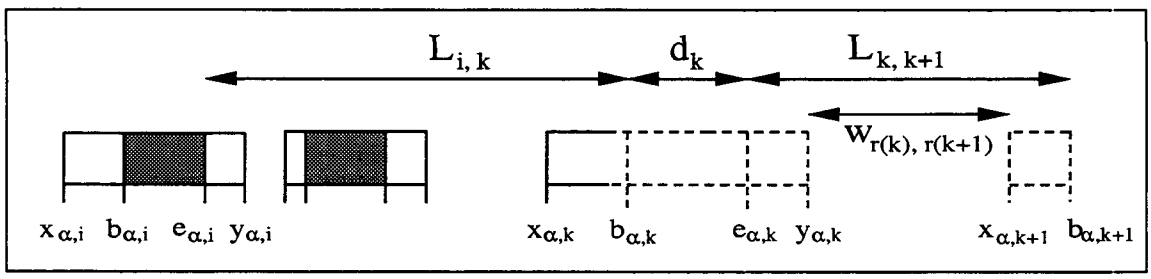


Figure 7.9 : Illustration des contraintes sur les tâches d'un échantillon.

Par conséquent, l'heure à laquelle α quitte r(k) doit satisfaire l'inégalité suivante :

$$y_{\alpha,k} \leq H_2 = \begin{cases} x_{\alpha,k} + \bar{d}_k & \text{si r(k) a une activation implicite} \\ \min\limits_{i<k}(e_{\alpha,i} + L_{i,k}) + \bar{d}_k + L_{k,k+1} - w_{r(k),r(k+1)} & \text{sinon} \end{cases}$$

Relevons que le calcul de $\bar{h}_\alpha$ peut être simplifié si la contrainte (5) ne concerne que des paires de tâches consécutives. Dans ce cas nous avons :

$$H_2 = \begin{cases} x_{\alpha,k} + \bar{d}_k & \text{si r(k) a une activation implicite} \\ e_{\alpha,k-1} + L_{k-1,k} + \bar{d}_k + L_{k,k+1} - w_{r(k),r(k+1)} & \text{sinon} \end{cases}$$

Comme $y_{\alpha,k}$ doit être à la fois inférieure à H_1 et à H_2, l'heure au plus tard à laquelle α peut être retiré de r(k) est donnée par $\bar{h}_\alpha = \min\{H_1, H_2\}$. ❐

En résumé, une nouvelle contrainte du POTELR est :

$$\boxed{(6) \quad \underline{h}_\alpha \leq y_{\alpha, t_\alpha(S)} \leq \bar{h}_\alpha}$$

A chaque étape de notre algorithme constructif, nous ne considérerons que les ordonnancements partiels qui satisfont la contrainte suivante :

$$\boxed{(7) \quad \underline{h}_\alpha \leq \bar{h}_\alpha \,, \quad \forall \alpha = 1,\ldots,n}$$

Soit S un ordonnancement partiel qui satisfait la contrainte (7). Pour obtenir l'ordonnancement partiel S_α à partir de S, la procédure DÉPLACEMENT(α) (où $k = t_\alpha(S)$) doit être exécutée :

$y_{\alpha,k}$	$\leftarrow$	$\underline{h}_\alpha$
$e_{\alpha,k}$	$\leftarrow$	$\min\{\underline{h}_\alpha, \min\limits_{i<k}(e_{\alpha,i} + L_{i,k}) + \bar{d}_k\}$
$b_{\alpha,k}$	$\leftarrow$	$\min\{e_{\alpha,k} - \underline{d}_k, \min\limits_{i<k}(e_{\alpha,i} + L_{i,k})\}$
$x_{\alpha,k+1}$	$\leftarrow$	$y_{\alpha,k} + w_{r(k),r(k+1)}$

Tableau 7-1 : Procédure DÉPLACEMENT(α).

En d'autres termes, cette procédure réalise les 4 étapes suivantes :

1. L'échantillon α quitte la ressource r(k) le plus tôt possible, c'est-à-dire en $\underline{h}_\alpha$.
2. Afin de conserver le plus de flexibilité possible, l'heure $e_{\alpha,k}$ de fin de la période active de la tâche t_k est fixée aussi tard que possible une fois l'heure $y_{\alpha,k}$ déterminée. Il sera ainsi plus aisé d'étendre notre ordonnancement partiel tout en satisfaisant la contrainte (5).

3. La durée de la période active de la tâche t_k ($e_{\alpha,k}$ - $b_{\alpha,k}$) est choisie aussi courte que possible une fois l'heure $e_{\alpha,k}$ connue.
4. Finalement, l'échantillon α est introduit dans la ressource r(k+1) aussitôt que possible.

Le départ du robot de R(S) est planifié au temps $y_{\alpha,k}$ - $w_{R(S),r(k)}$. Cela signifie donc qu'il attend devant la ressource R(S) entre les heures H(S) et $y_{\alpha,k}$ - $w_{R(S),r(k)}$.

Il est possible qu'à un certain stade de notre heuristique constructive, aucune opération admissible du robot ne permette d'étendre l'ordre courant S des opérations à un ordre S_α qui respecte les contraintes (1) à (5) et (7) du POTELR. Dans ce cas on parlera d'un ordonnancement partiel *non admissible*. Cependant, cela ne signifie pas forcément que l'ordre courant S des opérations doive être changé car il est possible que de mauvais choix aient été faits lorsque nous avons fixé les heures $x_{\beta,i}$, $b_{\beta,i}$, $e_{\beta,i}$ et $y_{\beta,i}$ ($\beta = 1, \ldots, n$ et $i = 0, \ldots, t_\beta(S)$) des opérations.

Pour illustrer nos propos, considérons l'exemple suivant du POTELR représenté dans la Figure 7.3 : n = 2, $\underline{d}_1 = 5$, $\overline{d}_1 = 6$, $w_{r(0),r(1)} = 2$, $w_{r(1),r(2)} = 3$ et $w_{r(0),r(2)} = 5$. De plus, nous supposons qu'il n'y a pas de contrainte du type (5) et que $\overline{d}_0 = \infty$. Comme solution initiale S de ce problème, nous avons $t_1(S) = t_2(S) = 0$, $x_{1,0} = x_{2,0} = 0$, $\underline{h}_1 = \underline{h}_2 = 0$ et $\overline{h}_1 = \overline{h}_2 = \infty$. Sans perte de généralité, nous pouvons supposer que l'échantillon 1 est choisi comme étant le premier échantillon à être déplacé par le robot. Ainsi en exécutant DÉPLACEMENT(1), nous obtenons un nouvel ordonnancement partiel avec $y_{1,0} = 0$ et $x_{1,1} = 2$. De plus, $\underline{h}_1 = 7$, $\overline{h}_1 = 8$, $\underline{h}_2 = 4$ et $\overline{h}_2 = \infty$. Comme deuxième opération du robot, nous pouvons décider, par exemple, de déplacer l'échantillon 1 dans la ressource r(2). Dans ce cas, nous exécutons DÉPLACEMENT(1) et par la suite deux fois consécutivement DÉPLACEMENT(2). Nous obtenons ainsi un ordonnancement complet pour les 2 échantillons, illustré dans la Figure 7.10, dont le makespan est de 25.

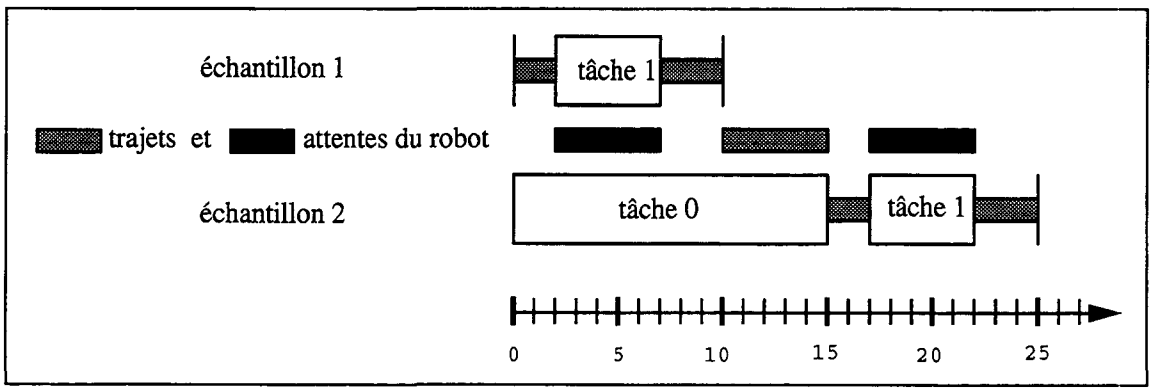


Figure 7.10 : Un ordonnancement en 25 unités de temps.

Si l'on choisit de déplacer l'échantillon 2 en lieu et place de l'échantillon 1 comme deuxième opération du robot (exécuter DÉPLACEMENT(2)), nous obtenons un nouvel ordonnancement partiel avec $y_{2,0} = 4$ et $x_{2,1} = 6$, $\underline{h}_1 = 7$, $\overline{h}_1 = 8$, $\underline{h}_2 = 11$ et $\overline{h}_2 = 12$. A ce stade de construction de l'ordonnancement, nous devons choisir de déplacer l'échantillon 1 car $\overline{h}_1 = 8 < 11 = \underline{h}_2$. Ainsi après avoir effectué DÉPLACEMENT(1), nous avons $y_{1,1} = 7$ et $x_{1,2} = 10$. Cet ordonnancement ne satisfait pas la contrainte (7) car la nouvelle heure de sortie au plus tôt de l'échantillon 2, $\underline{h}_2 = H(S) + w_{R(S),r(1)} = 13$, est supérieure à $\overline{h}_2 = 12$ (voir Figure 7.11).

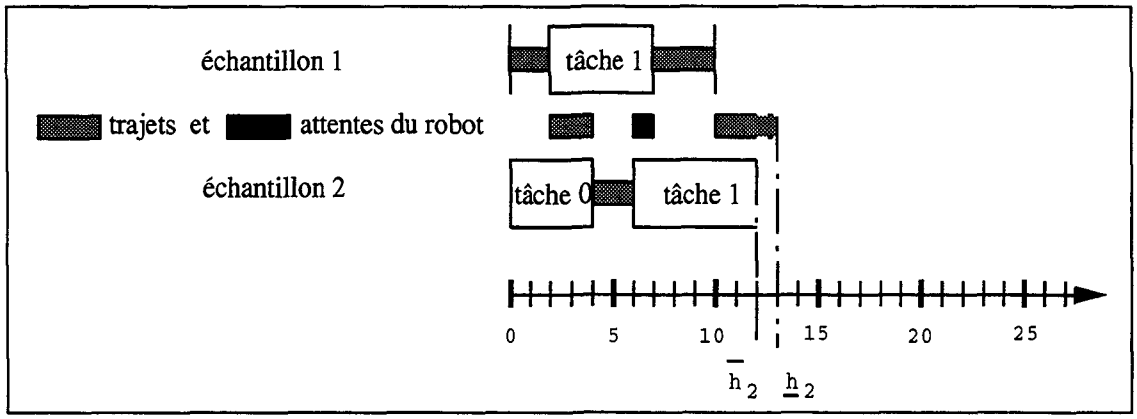


Figure 7.11 : Un ordonnancement partiel qui ne satisfait pas la contrainte (7).

Cela ne signifie pas pour autant que les deux premières opérations du robot ne doivent pas consister en le déplacement des échantillons 1 et 2 de la ressource r(0) vers la ressource r(1). En effet, une telle stratégie nous permet d'obtenir une solution optimale S*, représentée dans la Figure 7.12, dont le makespan est de 16 unités de temps.

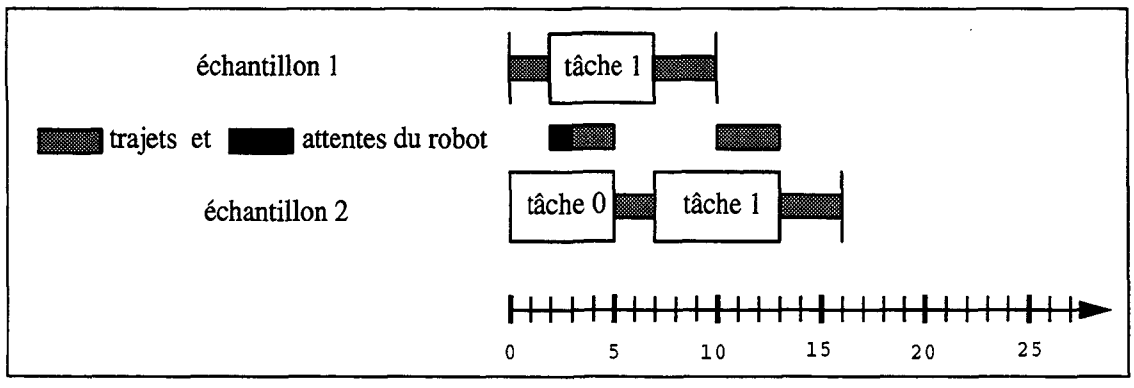


Figure 7.12 : Un ordonnancement optimal en 16 unités de temps.

En comparant attentivement cet ordonnancement optimal avec celui de la Figure 7.11, nous observons que pour obtenir S* il faut fixer la valeur de la variable $x_{2,1}$ à 7 au lieu de 6. Dans la prochaine section, nous montrons comment une telle décision peut être prise en temps polynomial.

7.2.3 Le problème central de l'ordonnancement : la recherche d'un plus long chemin dans un graphe

Au vu de l'exemple précédent, il est clair qu'au cours de notre algorithme de construction d'un ordonnancement pour le POTELR nous allons devoir être capables de répondre à la question suivante : est-ce qu'une suite ordonnée des opérations du robot peut fournir une solution admissible partielle de notre problème ? En d'autres termes, il s'agit de savoir s'il est possible de fixer les heures auxquelles auront lieu chacune des opérations de notre suite ordonnée de façon à obtenir un ordonnancement partiel qui respecte l'ensemble des contraintes du POTELR. Pour répondre rapidement et efficacement à la question précédente, nous avons développé une nouvelle procédure, désignée ci-après par PLUS LONG CHEMIN. Son but est de résoudre un problème de recherche d'un plus long chemin dans un graphe orienté G construit sur la base de la suite ordonnée des mouvements du robot.

Les contraintes (1) à (5) de notre problème sont des contraintes dites *potentielles* c'est-à-dire qu'elle requièrent que la valeur de la différence entre deux variables soit plus grande ou égale à une valeur donnée (i.e. $x_j - x_i \geq b_k$). A chaque système $A \cdot x \geq b$ de contraintes de type potentiel, appelé *système de différences*, on peut associer un graphe *potentiels-tâches* $G = (V, A)$ où :

- l'ensemble V des sommets est composé d'un sommet v_i pour chaque variable x_i du système et d'un sommet additionnel v_0 qui garantit que chaque sommet v_i est accessible depuis v_0,
- l'ensemble A des arcs est défini par :
  - un arc (v_0,v_i) de longueur 0 pour chaque inconnue x_i du système,
  - un arc (v_i,v_j) de longueur b_k pour chaque contrainte potentielle $x_j - x_i \geq b_k$.

Il est bien connu (voir [Kel61] ou le chapitre 25 du livre de Cormen et al. [Cor90]) qu'un système de différences $A \cdot x \geq b$ et son graphe potentiels-tâches associé G satisfont les propriétés suivantes :

- si G contient un circuit de longueur positive alors le système n'a pas de solutions admissibles,
- si G ne contient pas de circuit de longueur positive, une solution du système est obtenue en donnant à chaque x_i la valeur du plus long chemin de v_0 à v_i dans G.

Par conséquent, étant donné une suite ordonnée des opérations du robot, décider si ces opérations peuvent être ordonnancées tout en satisfaisant les contraintes (1), ..., (5) et (7) est équivalent à tester si un système de différences admet une solution ou non. Le graphe potentiels-tâches G = (V,A) associé à une suite ordonnée des opérations du robot est défini dans le paragraphe suivant.

Le graphe potentiels-tâches G = (V, A) d'un POTELR

L'ensemble V des sommets de G consiste en un sommet spécial, a, ainsi que des ensembles X, B, E et Y suivants :

$$X = \{ x_{\alpha,i} \mid \alpha = 1,\ldots,n \quad \text{et} \quad i = 1,\ldots,\min\{T, t_\alpha(S)+1\} \}$$
$$B = \{ b_{\alpha,i} \mid \alpha = 1,\ldots,n \quad \text{et} \quad i = 1,\ldots,\min\{T-1, t_\alpha(S)\} \}$$
$$E = \{ e_{\alpha,i} \mid \alpha = 1,\ldots,n \quad \text{et} \quad i = 1,\ldots,\min\{T-1, t_\alpha(S)\} \}$$
$$Y = \{ y_{\alpha,i} \mid \alpha = 1,\ldots,n \quad \text{et} \quad i = 0,\ldots,\min\{T-1, t_\alpha(S)\} \}$$

L'ensemble A est composé d'arcs de quatre types distincts, décrits ci-dessous :

Type 1

Pour chaque échantillon α, les arcs suivants représentent le temps passé par α dans la ressource de stockage d'entrée r(0) :

arc $(a, y_{\alpha,0})$ de longueur 0
arc $(y_{\alpha,0}, a)$ de longueur $-\bar{d}_0$

Type 2

Pour chaque échantillon α et chaque tâche t_i ($1 \leq i \leq t_\alpha(S)$), les arcs suivants représentent les contraintes (2) à (4) :

arc$(x_{\alpha,i}, b_{\alpha,i})$ de longueur 0
arc$(b_{\alpha,i}, e_{\alpha,i})$ de longueur $\underline{d}_i$
arc$(e_{\alpha,i}, b_{\alpha,i})$ de longueur $-\bar{d}_i$
arc$(e_{\alpha,i}, y_{\alpha,i})$ de longueur 0

De surcroît, si r(i) a une activation implicite, les arcs suivants imposent que la période active de la tâche t_i pour l'échantillon α doit correspondre à l'intervalle $[x_{\alpha,i}, y_{\alpha,i}]$:

arc $(b_{\alpha,i}, x_{\alpha,i})$ de longueur 0
arc $(y_{\alpha,i}, e_{\alpha,i})$ de longueur 0

Type 3

Pour chaque échantillon α et pour chaque paire (t_i, t_j) de tâches $(0 \le i \le j \le \min\{T, t_\alpha(S)+1\})$ telle que $L_{i,j} < \infty$, un des arcs suivants représente la contrainte (5) :

$\text{arc}(b_{\alpha,j}, e_{\alpha,i})$ de longueur $-L_{i,j}$ si $i>0$ et $j<\min\{T, t_\alpha(S)+1\}$
$\text{arc}(x_{\alpha,j}, e_{\alpha,i})$ de longueur $-L_{i,j}$ si $i>0$ et $j=\min\{T, t_\alpha(S)+1\}$
$\text{arc}(b_{\alpha,j}, y_{\alpha,i})$ de longueur $-L_{i,j}$ si $i=0$ et $j<\min\{T, t_\alpha(S)+1\}$
$\text{arc}(x_{\alpha,j}, y_{\alpha,i})$ de longueur $-L_{i,j}$ si $i=0$ et $j=\min\{T, t_\alpha(S)+1\}$

Type 4

Les arcs de ce type représentent les mouvements (réalisés ou potentiels) du robot :

↦ Le transport de l'échantillon α entre les ressources r(i) et r(i+1) , $0 \le i \le t_\alpha(S)$, est représenté par l'arc :

$$\text{arc}\,(y_{\alpha,i}, x_{\alpha,i+1}) \quad \text{de longueur } w_{r(i),r(i+1)}$$

↦ Considérons deux opérations consécutives du robot, la première étant le déplacement de l'échantillon γ dans la ressource r(i) où sa tâche t_i sera exécutée $(1 \le i \le t_\gamma(S))$ et la seconde consistant à transporter l'échantillon α de la ressource r(j) vers la ressource r(j+1) où sa tâche t_{j+1} $(1 \le j \le t_\alpha(S))$ se déroulera. L'arc suivant représente le trajet à vide du robot entre r(i) et r(j) :

$$\text{arc}\,(x_{\gamma,i}, y_{\alpha,j}) \quad \text{de longueur } w_{r(i),r(j)}$$

↦ Soit α un échantillon quelconque tel que $t_\alpha(S) < T$. Si l'on pose $\gamma = s(S)$ et $k = t_\alpha(S)$, les arcs suivants correspondent aux deux prochains mouvements du robot si l'on décide de déplacer l'échantillon α.

$$\text{arc}\,(x_{\gamma,t_\gamma(S)}, y_{\alpha,k}) \quad \text{de longueur } w_{R(S),r(k)}$$
$$\text{arc}\,(y_{\alpha,k}, x_{\alpha,k+1}) \quad \text{de longueur } w_{r(k),r(k+1)}$$

Ceci termine la définition du graphe potentiels-tâches G = (V, A). Comme déjà mentionné précédemment, le graphe G contient un circuit de longueur positive si et seulement si la suite ordonnée des opérations du robot ne peut pas être ordonnancée tout en satisfaisant les contraintes (1) à (5) et (7). Si G ne contient pas de circuit de longueur positive, alors un ordonnancement partiel admissible S_G peut être obtenu en donnant à chaque variable $x_{\alpha,i}$, $b_{\alpha,i}$, $e_{\alpha,i}$ et $y_{\alpha,i}$ la valeur du plus long chemin de a au sommet de G associé à la variable considérée. C'est ce que l'on appelle l'*ordonnancement au plus tôt.* L'utilisation de notre procédure PLUS LONG CHEMIN n'est donc pas seulement utile pour décider si un ordonnancement partiel est admissible ou non, mais permet également de minimiser le makespan d'une suite ordonnée des opérations du robot.

Trouver un plus long chemin entre deux sommets dans un graphe orienté G est un problème qui se résout dans un temps polynomial en la taille du problème considéré. Nous avons utilisé l'algorithme de Bellman [Bel58] qui soit calcule le plus long chemin de a à tous les autres sommets du graphe G soit détecte un circuit de longueur positive dans G en $O(|A| \cdot |V|)$. Dans ce dernier cas, certaines des contraintes du POTELR sont violées et il faudra modifier l'ordre courant des opérations du robot. Pour illustrer notre procédure PLUS LONG CHEMIN, considérons la suite ordonnée des opérations correspondant à l'ordonnancement partiel représenté dans la Figure 7.11. Le graphe potentiels-tâches G associé est décrit dans la Figure 7.13.

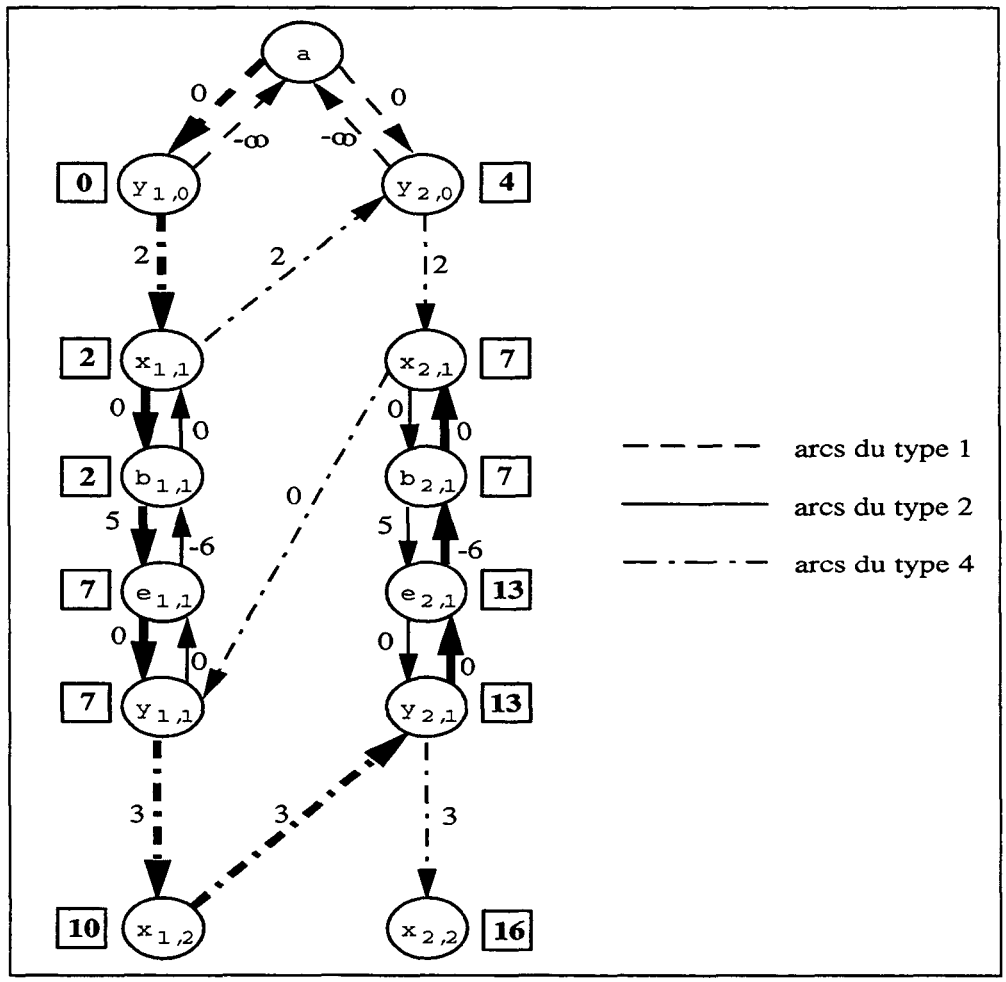


Figure 7.13 : Illustration d'un graphe potentiels-tâches pour le POTELR.

Le plus long chemin de a à $x_{2,1}$ est représenté en gras dans cette figure. Sa longueur est égale à 7, ce qui signifie que l'échantillon 2 ne doit pas être introduit dans r(1) avant l'heure 7 comme nous l'avons déjà mentionné à la fin de la section 7.2.2. Comme le graphe G ne contient pas de circuit de longueur positive, un ordonnancement partiel admissible peut être obtenu en donnant à chaque variable $x_{\alpha,i}$, $b_{\alpha,i}$, $e_{\alpha,i}$ et $y_{\alpha,i}$ la valeur du plus long chemin de a au sommet de G associé à la variable considérée. Ces valeurs sont indiquées à côté de chaque sommet de G dans la Figure 7.13.

7.2.4 Une sorte de "backtracking"

Lorsqu'un circuit de longueur positive est détecté, l'ordre courant des opérations doit être modifié puisqu'il n'est pas admissible. Pour cela, nous avons décidé d'éliminer de l'ordre courant toutes les opérations du robot impliquant un échantillon α, ce dernier étant choisi sur la base d'une règle $\mathcal{R}_2$ qui sera expliquée dans la partie suivante. Les autres opérations ne sont pas modifiées. Le graphe réduit correspondant à ce nouvel ordre ainsi modifié est noté G - $\{\alpha\}$.

Construction du graphe réduit G - $\{\alpha\}$

Mise à jour des mouvements du robot

1. Considérons un chemin P dans G de la forme suivante (où $\beta \neq \alpha$, $\delta \neq \alpha$ et $x \rightarrow y$ représente l'arc (x, y)) :

   $$x_{\beta,i} \rightarrow y_{\alpha,j} \rightarrow x_{\alpha,j+1} \rightarrow y_{\alpha,j+1} \rightarrow x_{\alpha,j+2} \rightarrow \ldots \rightarrow x_{\alpha,j+h} \rightarrow y_{\delta,k}$$

   Le chemin P est remplacé par un arc $x_{\beta,i} \rightarrow y_{\delta,k}$ de longueur $w_{r(i),r(k)}$

2. Enlever tous les arcs du type 4 qui concernent l'échantillon α.

3. Soit β le dernier échantillon, différent de α, qui a été déplacé par le robot dans S. Dans l'ordonnancement partiel correspondant au nouvel ordre des opérations du robot, le robot est positionné devant la ressource $r(t_\beta(S))$. Comme la prochaine instruction qui sera donnée au robot pourrait être de déplacer l'échantillon α de la ressource r(0) vers la ressource r(1), les arcs suivants doivent être ajoutés :

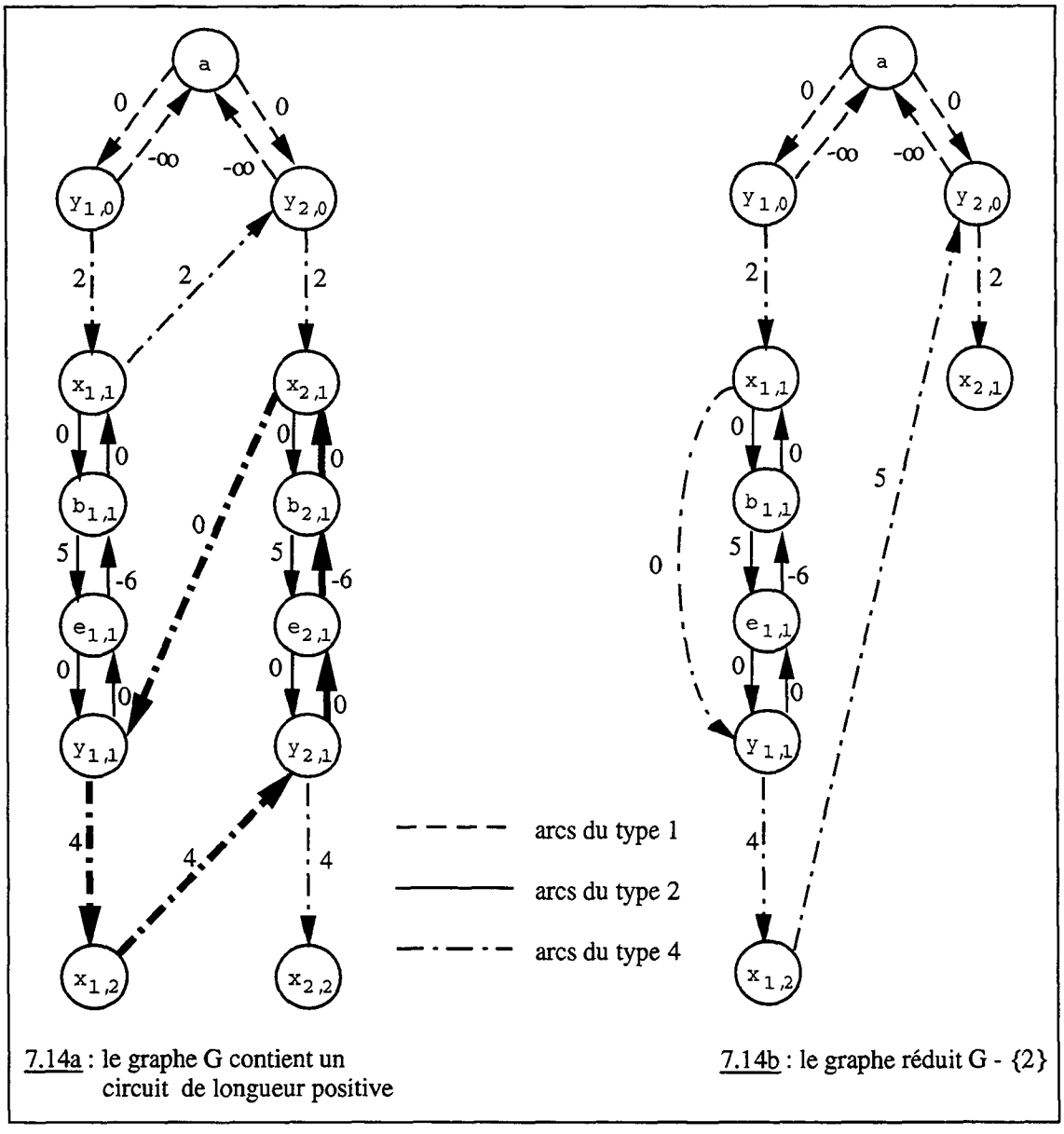
   - arc $(x_{\beta,t_\beta(S)}, y_{\alpha,0})$ de longueur $w_{r(t_\beta(S)),r(0)}$
   - arc $(y_{\alpha,0}, x_{\alpha,1})$ de longueur $w_{r(0),r(1)}$

Effacement des arcs et des sommets restants qui concernent l'échantillon α

4. Enlever tous les arcs du type 2 et 3 qui concernent l'échantillon α.

5. Ôter les sommets $b_{\alpha,i}$, $e_{\alpha,i}$ et $y_{\alpha,i}$, $1 \leq i \leq t_\alpha(S)$, ainsi que les sommets $x_{\alpha,i}$, $2 \leq i \leq \min\{T, t_\alpha(S) + 1\}$.

Considérons le même exemple que celui représenté dans la Figure 7.12, avec comme seule différence le fait que $w_{r(1),r(2)}$ égale 4 au lieu de 3. Les trois premières opérations du robot consistent à déplacer les échantillons 1 et 2 dans la ressource r(1) puis à transporter l'échantillon 1 dans la ressource r(2). Le graphe G qui correspond à cet ordonnancement partiel est représenté dans la Figure 7.14a. Il contient un circuit de longueur 2 qui est dessiné en gras dans cette figure.



7.14a : le graphe G contient un circuit de longueur positive

7.14b : le graphe réduit G - {2}

Figure 7.14 : Le graphe associé à un ordonnancement partiel des mouvements du robot.

En enlevant l'échantillon 2 de cet ordre partiel, nous obtenons le graphe réduit G - {2} de la Figure 7.14b qui ne contient pas de circuit de longueur positive. L'ordonnancement partiel $S_{G-\{2\}}$ peut dorénavant être complété en appelant la procédure DÉPLACEMENT(2) deux fois consécutivement.

Supposons que le graphe G associé à l'ordre partiel des opérations du robot contienne un circuit de longueur positive. En enlevant toutes les opérations concernant l'échantillon α, il est fort possible que le graphe réduit G - $\{\alpha\}$ contienne toujours un circuit de longueur positive. Dans un tel cas, un échantillon additionnel est ôté et ce processus est répété tant que le graphe contient un circuit de longueur positive. La règle pour choisir quel échantillon ôter sera définie plus tard. Nous pouvons désormais décrire notre algorithme constructif d'ordonnancement pour un POTELR.

7.2.5 Description de l'algorithme

Les décisions locales qui sont prisent dans notre algorithme constructif reposent sur deux règles $\mathcal{R}_1$ et $\mathcal{R}_2$, définies après la description de notre heuristique.

(0) Initialiser l'ordonnancement partiel en posant
$x_{\alpha,0} := 0$; $t_\alpha(S) := 0$; $\underline{h}_\alpha := 0$; $\overline{h}_\alpha := \overline{d}_0$; pour $\alpha = 1, \ldots, n$
c := 0; { compteur d'itérations }

(1) Sur la base d'une règle $\mathcal{R}_1$, choisir un échantillon α avec $t_\alpha(S) < T$;

(2) Poser c := c + 1 ;
Appeler DÉPLACEMENT(α) ; poser S := S_α ;
Mettre à jour $\underline{h}_\beta$ et $\overline{h}_\beta$ pour $\beta = 1, \ldots, n$
Si S est un ordonnancement partiel admissible qui satisfait la contrainte (7)
alors Aller à l'étape (7) ;

(3) Soit O l'ordre partiel des opérations du robot induit par S ;
Construire le graphe potentiels-tâches G associé à l'ordre partiel O ;

(4) Utiliser l'algorithme de Bellman pour soit calculer le plus long chemin de a à tous les autres sommets de G, soit déterminer la présence d'un circuit de longueur positive dans G ;
Si G ne contient pas de circuit de longueur positive
alors Poser S := S_G ;
Aller à l'étape (7) ;

(5) Sur la base d'une règle $\mathcal{R}_2$, choisir un échantillon β avec $t_\beta(S) > 0$;

(6) Poser c := c - $t_\beta(S)$ et G := G - $\{\beta\}$;
Aller à l'étape (4) ;

(7) **Si** c = n·T
alors STOP
sinon Aller à l'étape (1) ;

Tableau 7-2 : L'algorithme d'ordonnancement pour le POTELR.

7.2.5.1 La règle $\mathcal{R}_1$

Considérons un ordonnancement partiel S. La première phase de notre heuristique consiste à choisir un échantillon α, à l'aide de la règle $\mathcal{R}_1$, parmi un ensemble A d'échantillons candidats pour la prochaine opération admissible du robot. Cet ensemble A est déterminé en tenant compte des capacités des ressources et du fait que certaines de ces ressources peuvent être bloquantes. Une ressource est dite *libre* si le nombre d'échantillons en cours de traitement dans cette ressource est strictement inférieur à sa capacité (i.e. si elle a encore au moins une place de traitement de libre).

Si la ressource R(S) devant laquelle se trouve le robot est une ressource bloquante, alors A = {s(S)} car le prochain échantillon que doit déplacer le robot est obligatoirement s(S). Dans le cas contraire, toutes les ressources bloquantes sont libres. Soit q_α ($t_\alpha(S) < k < q_\alpha$) l'indice de la tâche telle que chaque ressource r(k) est bloquante et $r(q_\alpha)$ ne l'est pas. En d'autres termes, si le robot retire l'échantillon α de $r(t_\alpha(S))$, il devra le déplacer successivement dans les ressources $r(t_\alpha(S)+1)$, ..., $r(q_\alpha)$ avant de pouvoir s'occuper d'un autre échantillon. Nous définissons alors A comme l'ensemble des échantillons α tel que la ressource $r(q_\alpha)$ est libre. De plus, si la ressource r(1) est libre alors seulement un échantillon α avec $t_\alpha(S) = 0$ (s'il en existe un) est introduit dans A puisque tous ces échantillons sont équivalents.

Avant de décrire la règle $\mathcal{R}_1$, considérons deux ingrédients additionnels que nous utilisons pour déterminer l'ensemble A des échantillons candidats. Le premier est utilisé pour restreindre le nombre d'échantillons en traitement dans le laboratoire. Ainsi, à chaque itération de l'algorithme, nous imposons que la taille de A soit inférieure ou égale à une valeur N qui est un paramètre de notre méthode. La raison de cette restriction tient dans l'observation suivante : si la taille de A est trop importante (elle augmente au fur et à mesure que le nombre d'échantillons en traitement augmente), il devient de plus en plus difficile de déterminer quelle est la prochaine opération du robot à réaliser pour obtenir un ordonnancement partiel qui pourrait être étendu en une solution complète admissible. En effet, nous avons observé que les étapes (5) et (6) de notre algorithme sont fréquemment utilisées dès que le nombre d'échantillons en traitement est trop important. Par conséquent, nous ne considérerons que les ordonnancement partiels S qui satisfont la contrainte supplémentaire suivante :

$$(8) \quad \left|\{\alpha \mid 0 < t_\alpha(S) < T\}\right| \leq N$$

Ainsi donc, la règle $\mathcal{R}_1$ doit choisir un échantillon parmi un ensemble de candidats dont le nombre est au plus N. Dans le cas où $\left|\{\alpha \mid 0 < t_\alpha(S) < T\}\right| = N$ et A contient un échantillon α pour lequel $t_\alpha(S) = 0$, l'échantillon α est ôté de A car si on décidait d'exécuter la tâche t_1 de cet échantillon, la contrainte (8) serait violée.

Le second ingrédient a pour objectif de diminuer le risque de cyclage de notre algorithme. Pour cela nous définissons une matrice M de taille n·T où chaque élément $m_{\alpha,i}$ ($1 \leq \alpha \leq n$; $0 \leq i < T$) stocke la dernière valeur assignée par notre heuristique à la variable $y_{\alpha,i}$. Initialement, chacun de ces éléments reçoit une valeur négative. Soit α un échantillon quelconque de A. S'il existe un échantillon β ($1 \leq \beta \leq n$) tel que $m_{\beta,t_\alpha(S)} = \underline{h}_\alpha$ alors α est retiré de A. La raison de ce choix est que $y_{\alpha,\, t_\alpha(S)}$ serait fixé à la valeur $\underline{h}_\alpha$ dans S_α alors que cette heure a déjà été utilisée, à une étape précédente de notre méthode, comme heure de fin de la période active de la tâche d'indice $t_\alpha(S)$ de β.

La règle $\mathcal{R}_1$ de choix d'un échantillon α parmi A est basé sur deux critères, le second intervenant seulement si le premier ne suffit pas à déterminer de façon unique un échantillon $\alpha \in A$. Si A est un ensemble vide, l'algorithme s'arrête. Dans le cas contraire, soit α un échantillon quelconque de A. Lorsque S_α ne respecte pas la contrainte (7), nous dénotons par V_α l'ensemble des échantillons actifs pour lesquels la contrainte (7) est violée dans S_α.

Critère 1 de $\mathcal{R}_1$

Nous choisissons un sous-ensemble B de A suivant l'une des définitions suivantes :

(a) $B = \{\alpha \in A \mid |V_\alpha| = \min_{\beta \in A} |V_\beta|\}$

(b) $B = \{\alpha \in A \mid \sum_{\delta \in V_\alpha} t_\delta(S) = \min_{\beta \in A} \sum_{\delta \in V_\beta} t_\delta(S)\}$

(c) $B = \{\alpha \in A \mid \sum_{\delta \in V_\alpha} (\underline{h}_\delta - \bar{h}_\delta) = \min_{\beta \in A} \sum_{\delta \in V_\beta} (\underline{h}_\delta - \bar{h}_\delta)\}$

(d) $B = \{\alpha \in A \mid \dfrac{\sum_{\delta \in V_\alpha} (\underline{h}_\delta - \bar{h}_\delta)}{|V_\alpha|} = \min_{\beta \in A} \dfrac{\sum_{\delta \in V_\beta} (\underline{h}_\delta - \bar{h}_\delta)}{|V_\beta|}\}$

(e) $B = \{$un échantillon de A choisi aléatoirement$\}$

Pour chaque échantillon $\delta \in V_\alpha$, une violation d'une contrainte du type (7) dans S_α est observée. Chacune de ces violations se caractérise par :

- son *rang*, $1 < \text{rang} < T$, qui est l'indice de la tâche en cours d'exécution sur l'échantillons actif δ,
- sa *durée*, égale à $\underline{h}_\delta - \bar{h}_\delta$, qui indique en unités de temps l'importance de la violation de la contrainte (7).

Ainsi donc, exprimé en d'autres termes, l'objectif de ce premier critère est de choisir des échantillons $\alpha \in A$ qui minimisent l'une des valeurs suivantes :

(a) le nombre d'échantillons pour lesquels la contrainte (7) est violée dans S_α,

(b) la somme des rangs de chaque violation d'une contrainte de type (7) dans S_α,

(c) la somme totale des durées des violations observées dans S_α.

(d) la durée moyenne des violations observées dans S_α.

Critère 2 de $\mathcal{R}_1$

Le second critère est utilisé uniquement si le cardinal de l'ensemble B est supérieur à un. Dans ce cas, nous choisissons un sous-ensemble C de B sur la base de l'une des définitions suivantes :

(f) $C = \{\alpha \in B \mid H(S_\alpha) = \min_{\beta \in B} H(S_\beta)\}$

(g) $C = \{\alpha \in B \mid (y_{\alpha,t_\alpha(S)} - w_{R(S),r(t_\alpha(S))} - H(S)) = \min_{\beta \in B} (y_{\alpha,t_\beta(S)} - w_{R(S),r(t_\beta(S))} - H(S))\}$

(h) $C = \{$un échantillon de B choisi aléatoirement$\}$

L'objectif de ce second critère est donc de choisir un échantillon candidat si et seulement si plusieurs d'entre eux sont équivalents en regard du premier critère. Parmi tous les échantillons équivalents par rapport au premier critère, nous en choisissons un soit aléatoirement (variante (h)) ou de sorte que l'opération admissible du robot le concernant soit :

(f) celle qui libère le robot le plus tôt (on cherche donc à minimiser l'heure à laquelle le robot se retrouvera prêt à effectuer un nouveau mouvement),

(g) celle qui prend l'échantillon à déplacer dans les meilleurs délais (on cherche donc à minimiser l'attente du robot avant la prise de l'échantillon qu'il doit déplacer).

Finalement, si par hasard il subsistait encore une ambiguïté quant au choix de l'échantillon à déplacer, on en choisirait un aléatoirement parmi ceux qui sont équivalents en regard du second critère.

Les variantes de la règle $\mathcal{R}_1$ que nous avons testées sont la définition (e) ainsi que toutes les combinaisons possibles obtenues en choisissant une définition dans {(a), (b), (c), (d)} et une autre dans {(f), (g), (h)}. En se basant sur les résultats obtenus pour un POTELR réel décrit dans la section 7.3, il s'avère que les meilleures variantes sont la combinaison de (a) ou (b) avec (g).

7.2.5.2 La règle $\mathcal{R}_2$

Soient α l'échantillon choisi par la règle $\mathcal{R}_1$ et S_α le nouvel ordonnancement obtenu après que le robot ait déplacé α. Supposons que S_α ne soit pas admissible et que l'on doive donc éliminer de l'ordre courant des opérations du robot toutes celles impliquant un échantillon β. Le choix de cet échantillon β est réalisé avec l'une des trois variantes suivantes de la règle $\mathcal{R}_2$ où D représente l'ensemble des échantillons γ en traitement (i.e. $0 < t_\gamma(S_\alpha) < T$) :

(i)	choisir l'échantillon $\beta \in D$ qui maximise $y_{\beta,0}$
(ii)	choisir aléatoirement un échantillon $\beta \in D$
(iii)	choisir aléatoirement un échantillon $\beta \in V_\alpha$

Avec la variante (i), β est le dernier échantillon sorti de r(0). Notons que si l'on combine (iii) avec la variante (b) de $\mathcal{R}_1$, nous avons plus de chance d'ôter de S_α un échantillon β qui ne soit pas trop avancé dans son processus analytique (i.e. tel que $t_\beta(S_\alpha)$ soit petit). La variante (iii) a fourni les meilleurs résultats pour le problème réel que nous avons étudié.

7.3 Les résultats numériques

7.3.1 Un problème réel

Nous avons testé notre algorithme heuristique d'ordonnancement sur un problème réel : la préparation d'échantillons pour l'identification des bactéries par leurs acides gras membranaires. Le laboratoire robotisé dans lequel cette application chimique est réalisée est constitué d'un unique robot ainsi que d'une ressource de stockage d'entrée, de 10 ressources de traitement et d'une ressource de stockage de sortie. Les caractéristiques de ces différentes ressources sont répertoriées dans le Tableau 7-3.

Ressource	Type d'activation	Capacité	Bloquante
Stockage d'entrée	Implicite	∞	Non
Distributeur 1	Implicite	1	Oui
Distributeur 2	Implicite	1	Oui
Distributeur 3	Implicite	1	Oui
Distributeur 4	Implicite	1	Oui
Distributeur 5	Implicite	1	Oui
Bain 100°C	Implicite	15	Non
Bain 80°C	Implicite	15	Non
Bain 20°C	Implicite	15	Non
Rotateur	Implicite	20	Non
Agitateur orbital	Explicite	1	Non
Stockage de sortie	Implicite	∞	Non

Tableau 7-3 : Description des ressources du laboratoire.

L'ensemble des tâches qui doivent être exécutées pour chaque échantillon de l'application sont représentées dans le Tableau 7-4 et illustrées dans la Figure 7.2. Nous indiquons l'indice de la tâche, la ressource du laboratoire qui l'exécute, les durées de traitement minimale et maximale ainsi que les contraintes du type (5). Dans notre cas, les seules contraintes du type (5) qui doivent être prise en compte concernent des tâches consécutives.

Indice i de la tâche	Ressource r(i)	Durée minimale $\underline{d}_i$	Durée maximale $\overline{d}_i$	$L_{i,\,i+1}$
0	Stockage d'entrée	0	∞	∞
1	Distributeur 1	90	90	∞
2	Agitateur orbital	5	10	∞
3	Bain 100°C	285	315	120
4	Agitateur orbital	5	10	∞
5	Bain 100°C	1425	1575	120
6	Bain 20°C	120	600	∞
7	Distributeur 2	90	90	∞
8	Agitateur orbital	5	10	∞
9	Bain 80°C	600	660	120
10	Bain 20°C	120	600	∞
11	Distributeur 3	90	90	∞
12	Rotateur	600	900	∞
13	Distributeur 4	135	135	∞
14	Rotateur	300	600	∞
15	Distributeur 5	165	165	∞
16	Stockage de sortie	0	∞	∞

Tableau 7-4 : L'ensemble ordonné des tâches qui doivent être exécutées pour chaque échantillon (les temps sont donnés en secondes).

Pour se calquer d'avantage à la réalité, nous avons modifié légèrement notre modèle. Supposons que $T_{in}(i)$ et $T_{out}(i)$ représentent les temps nécessaires au robot pour respectivement déposer un échantillon dans r(i) et retirer un échantillon de r(i). Lorsqu'une opération du robot est effectuée, celle-ci est toujours composée des deux mouvements suivants du robot : le déplacement de R(S) vers une ressource r(i) pour y retirer un échantillon α et le trajet de r(i) vers la ressource r(i+1) pour y déposer α. Les durées de ces deux mouvements sont donc respectivement égales à $w_{R(S),r(i)} + T_{out}(i)$ et $w_{r(i),r(i+1)} + T_{in}(i+1)$. Nous considérons donc deux matrices pour indiquer les temps de déplacement du robot : l'une représente les temps nécessaires au robot pour aller retirer un échantillon d'une ressource et l'autre ceux requis pour déposer un échantillon dans une ressource. Ainsi, chaque occurrence de la variable $w_{r(i),r(j)}$ dans les sections précédentes fait référence à l'une ou l'autre de ces deux matrices selon le type de mouvement effectué par le robot. Par exemple, reprenons la contrainte (1) de la section 7.2.2 :

$$(1) \quad x_{\gamma,i} + w_{r(i),r(j)} \leq y_{\alpha,j} \leq x_{\alpha,j+1} - w_{r(j),r(j+1)}$$

La valeur de $w_{r(i),r(j)}$ doit être lue dans la première matrice puisque le robot doit se rendre de r(i) (où il vient de déposer γ) vers r(j) pour y retirer α. Par contre, la valeur $w_{r(j),r(j+1)}$ est tirée de la seconde matrice car cette fois le robot doit ôter α de r(j) pour aller le déposer dans r(j+1).

La première matrice des temps de déplacement $w_{r(i),r(j)}$ du robot est donnée dans le Tableau 7-5.

r(i) / r(j)	1	2	3	4	5	6	7	8	9	10	11	12
1	3	3	3	3	3	3	9	10	10	10	8	7
2	3	0	0	0	0	0	5	5	5	6	5	3
3	3	0	0	0	0	0	5	5	5	6	5	3
4	3	0	0	0	0	0	5	5	5	6	5	3
5	3	0	0	0	0	0	5	5	5	6	5	3
6	3	0	0	0	0	0	5	5	5	6	5	3
7	6	5	5	5	5	5	6	8	8	9	8	6
8	7	5	5	5	5	5	8	6	8	9	8	6
9	7	5	5	5	5	5	8	8	6	8	9	5
10	8	7	7	7	7	7	10	10	9	7	10	6
11	7	7	7	7	7	7	10	10	11	11	8	5
12	4	3	3	3	3	3	6	6	5	5	3	0

Tableau 7-5 : Temps de déplacement du robot (en secondes) pour aller retirer un échantillon d'une ressource.

Nous observons que les éléments diagonaux de cette matrice ne sont pas nécessairement nuls. En effet, si l'on décide de déplacer un échantillon de r(i) vers r(i+1) et que le robot se trouve déjà en r(i) (i.e. R(S) = r(i)), nous aurons un temps de déplacement de R(S) à r(i) qui peut être plus grand que zéro puisqu'il est égal à $T_{out}(i)$.

Pour la seconde matrice, seules les valeurs $w_{r(i),r(i+1)}$ $(0 \leq i < T)$ sont pertinentes. Celles-ci sont donc représentées sous la forme d'un vecteur dans le Tableau 7-6 ci-dessous.

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$w_{r(i),r(i+1)}$	3	5	10	8	10	8	5	5	10	8	5	6	7	6	7	3

Tableau 7-6 : Temps de déplacement du robot (en secondes) pour aller déposer un échantillon dans une ressource.

Soit S un ordonnancement complet admissible pour le traitement de n échantillons. Si l'on désire effectuer le même ordonnancement pour n échantillons additionnels, le robot doit préalablement se déplacer de r(m+1) vers r(0). Dans ce cas, la combinaison de S et du mouvement du robot de r(m+1) vers r(0) définit un cycle C qui peut être répété indéfiniment. Si M(S) représente le makespan de S, nous définissons la valeur f(S) de S comme :

$$f(S) = \frac{M(S) + w_{r(m+1),r(0)}}{n}$$

Ainsi, f(S) est égal au temps moyen nécessaire pour traiter un échantillon dans une séquence répétée du cycle C des mouvements du robot. Notons que minimiser cette valeur revient à maximiser le taux de production moyen ou, en d'autres termes, le nombre d'échantillons déplacés dans la ressource r(m+1) par unité de temps. La séquence des opérations du robot ne doit pas être différente à chaque fois qu'un nouveau groupe d'échantillons entre dans le laboratoire car le nombre de ces échantillons peut varier de jour en jour et de plus, il n'est généralement pas connu à l'avance. Pour ces raisons, le nombre d'échantillons traités dans un cycle C ne doit pas être trop grand. Pour déterminer un cycle C d'une taille raisonnable, nous proposons la stratégie suivante.

Soient $\bar{n}$ une borne supérieure sur le nombre d'échantillons qui peuvent être traités dans un cycle C, $S^*(\bar{n}, N)$ l'ordonnancement obtenu avec notre algorithme pour $\bar{n}$ échantillons et une borne supérieure N pour la taille de l'ensemble A des échantillons candidats et finalement $O^*(\bar{n}, N)$ l'ordre des opérations du robot induit par l'ordonnancement $S^*(\bar{n}, N)$. Alors, étant donné n'importe quel nombre $n \leq \bar{n}$ d'échantillons, un ordonnancement admissible complet S(n, N) pour n échantillons peut être obtenu en appliquant la procédure suivante :

- considérer l'ordre des opérations O(n, N) obtenu à partir de $O^*(\bar{n}, N)$ en ôtant toutes les opérations du robot qui impliquent les échantillons n+1, n+2, ..., $\bar{n}$,
- construire le graphe potentiels-tâches G(n, N) associé à l'ordre O(n, N),
- donner à chaque variable $x_{\alpha,i}$, $b_{\alpha,i}$, $e_{\alpha,i}$ et $y_{\alpha,i}$ la valeur du plus long chemin de a au sommet de G(n, N) associé à la variable considérée. Ainsi, $S(n, N) = S_{G(n, N)}$. Cette dernière étape est toujours possible car $S^*(\bar{n}, N)$ est admissible et par conséquent G(n, N) ne contient pas de circuit de longueur positive. Notons de plus que par construction, le makespan de S(n, N) est inférieur ou égal au temps nécessaire pour traiter les n premiers échantillons dans $S^*(\bar{n}, N)$.

Comme le nombre d'échantillons qui doivent être traités chaque jour dans l'application réelle considérée varie entre 100 et 1000, nous avons décidé de fixer la valeur de $\bar{n}$ à 50. Dans la Figure 7.15, la valeur de S(n, N) (n = 1, ..., 50) est comparée avec la valeur de l'ordonnancement obtenu à partir de S*(50, N) en ôtant toutes les opérations du robot impliquant les échantillons n+1, n+2, ..., $\bar{n}$ = 50. Toutes ces valeurs ont été obtenues en posant N = 9 et en utilisant la combinaison (a)-(g)-(iii) des variantes des règles $\mathcal{R}_1$ et $\mathcal{R}_2$. Le meilleur ordonnancement, de valeur 948, est obtenu pour n = 31 alors que f(S*(50, 9) = f(S(50, 9)) = 1015.

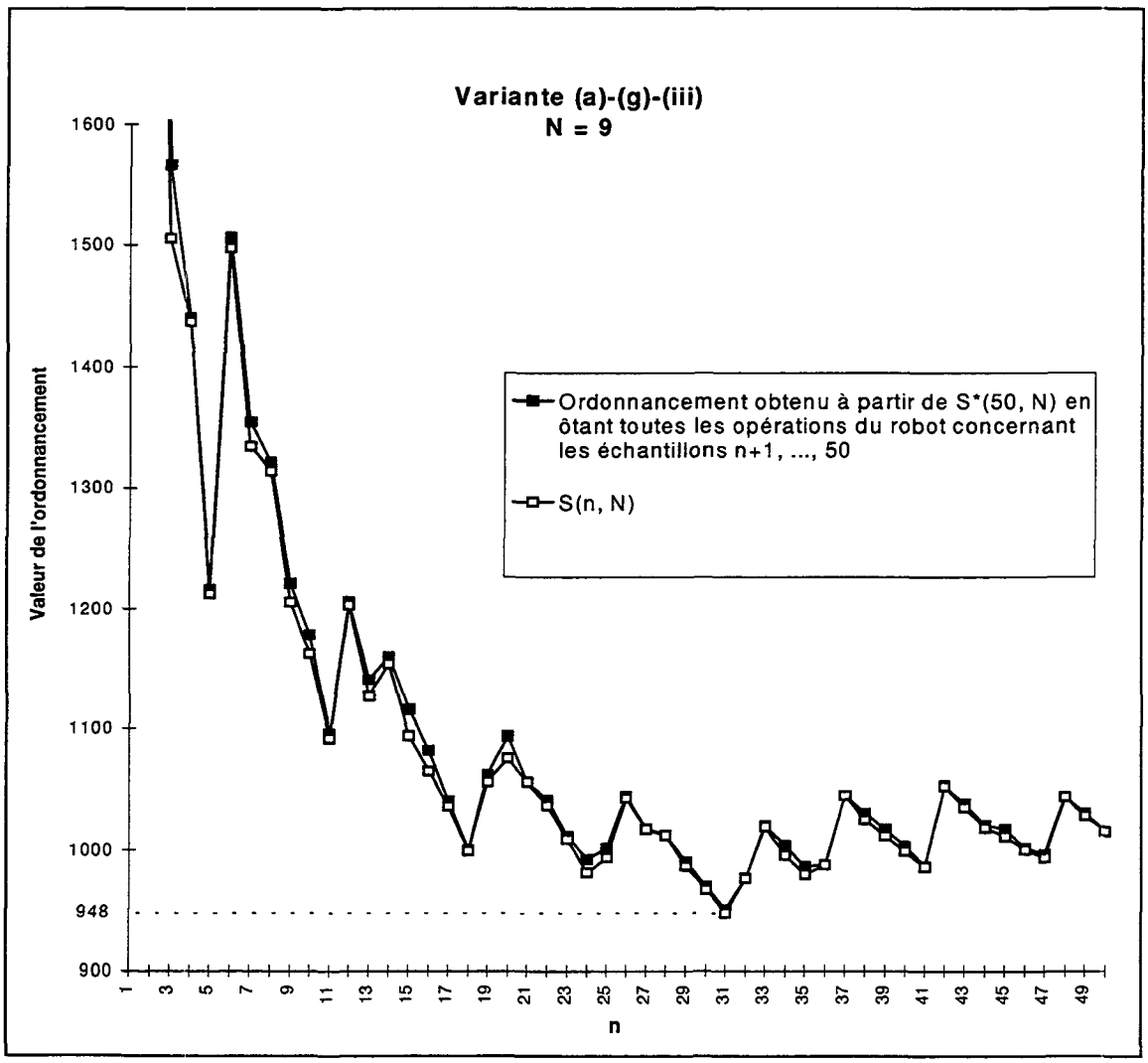


Figure 7.15 : Comparaison entre S(n, N) et les ordonnancements obtenus à partir de S(50, N).*

Nous définissons F($\bar{n}$, N) comme la valeur du meilleur ordonnancement parmi S(1, N), ..., S($\bar{n}$, N), i.e. $F(\bar{n},N) = \min_{n=1,...,n} f(S(n,N))$. Dans ce cas, F(50,9)=f(S(31,9)).

Nous avons testé 13 variantes pour la règle $\mathcal{R}_1$: (a)-(f), (a)-(g), (a)-(h), (b)-(f), (b)-(g), (b)-(h), (c)-(f), (c)-(g), (c)-(h), (d)-(f), (d)-(g), (d)-(h) et (e). Ces variantes ont été ensuite combinées avec les définitions de la règle $\mathcal{R}_2$ si bien que le nombre total des variantes testées est de 39. Dans le but de comparer les 39 variantes des règles $\mathcal{R}_1$ et $\mathcal{R}_2$ de notre algorithme, nous avons calculé $F(\bar{n}, N)$ pour $N = 1, \ldots, n$. De ces calculs, il apparaît clairement que la définition (iii) doit être utilisée pour la règle $\mathcal{R}_2$. Comme exemple, nous avons représenté dans la Figure 7.16 les résultats obtenus en combinant la variante (a)-(g) de la règle $\mathcal{R}_1$ avec les trois définitions possibles de la règle $\mathcal{R}_2$. Les trois courbes dessinées dans la Figure 7.16 sont représentatives de tous les autres choix possibles pour la règle $\mathcal{R}_1$.

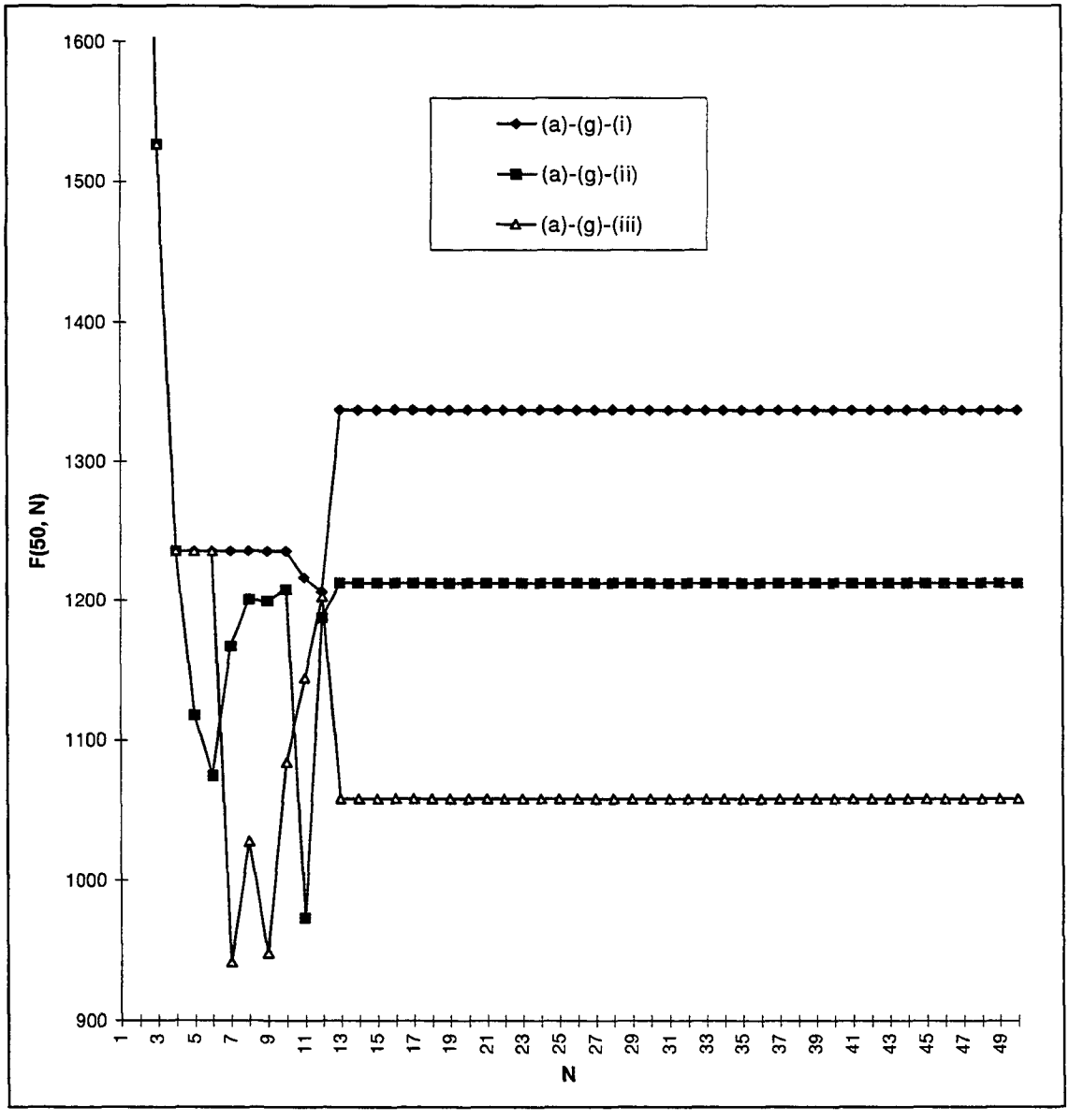


Figure 7.16 : Comparaison des trois définitions possibles de la règle $\mathcal{R}_2$.

De plus, nous avons remarqué que la définition (g) de la règle $\mathcal{R}_1$ domine clairement les définitions (f) et (h). Pour illustrer notre propos, nous avons représenté les résultats obtenus avec les combinaisons (d)-(f)-(iii), (d)-(g)-(iii), (d)-(h)-(iii) dans la Figure 7.17.

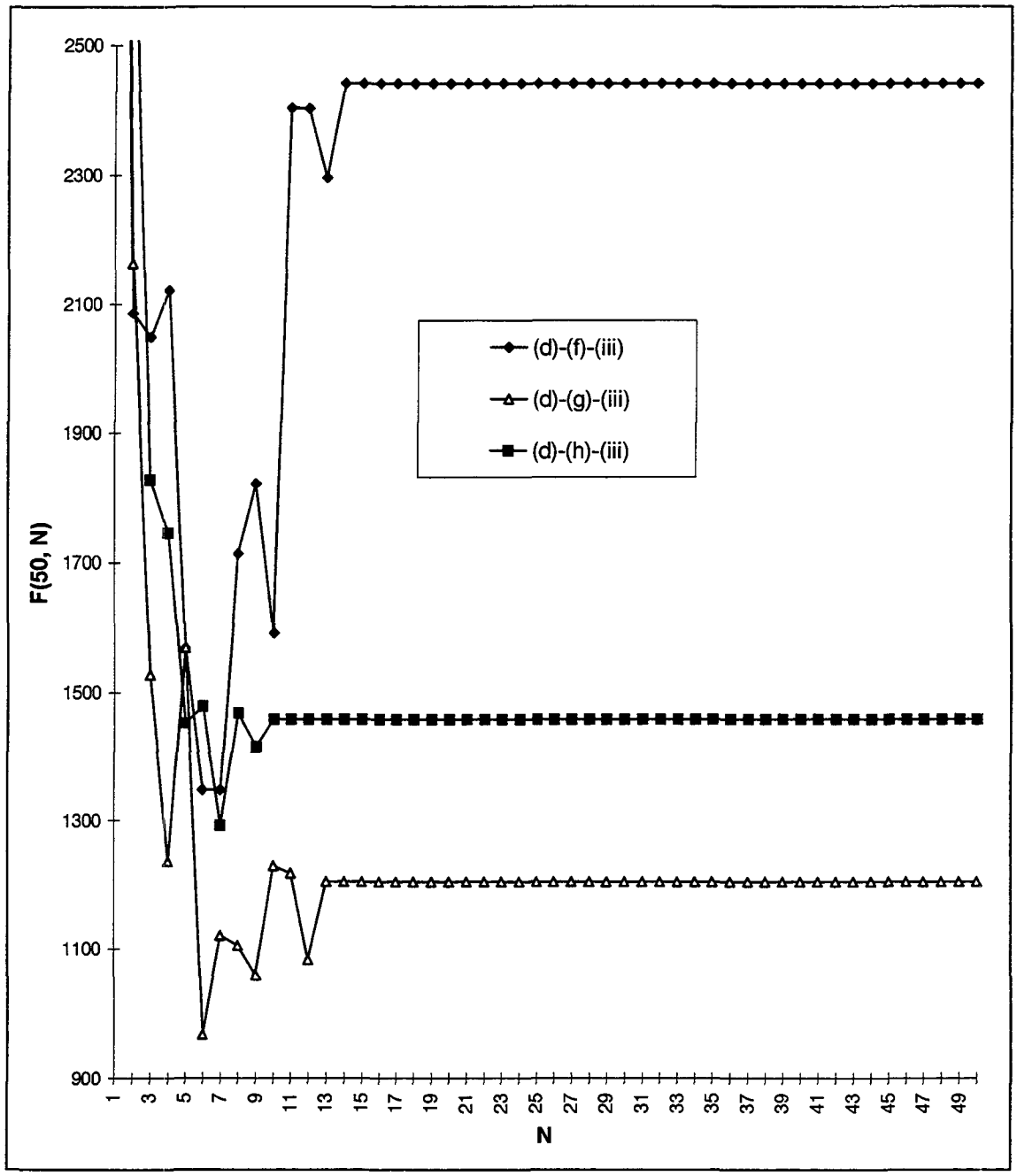


Figure 7.17 : Comparaison des trois définitions possibles du second critère de la règle $\mathcal{R}_1$.

Il est important de noter que nous avons obtenu des résultats similaires lorsque les variantes (a), (b) ou (c) étaient utilisées en lieu et place de (d).

Finalement, nous avons combiné (g)-(iii) avec les définitions (a), (b), (c) et (d). Les résultats sont illustrés dans la Figure 7.18. Le meilleur ordonnancement S* que nous avons obtenu, en fixant N = 7 et en utilisant les combinaisons (a)-(g)-(iii) ou (b)-(g)-(iii), a une valeur de 942.

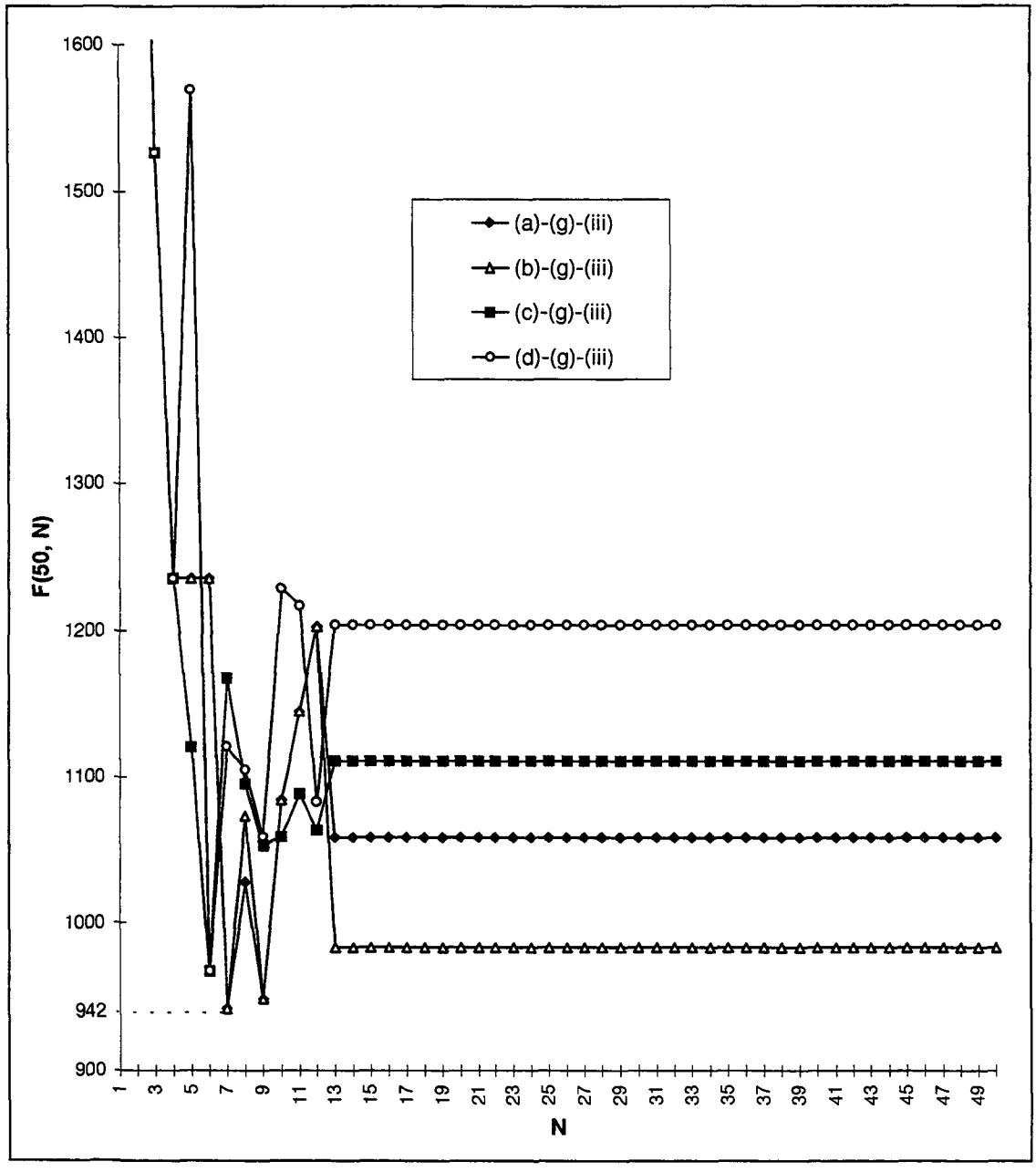


Figure 7.18 : Comparaison des cinq définitions possibles du premier critère de la règle $\mathcal{R}_1$.

Signalons que la définition (e) (choix aléatoire du prochain échantillon à déplacer) n'est pas représentée dans cette figure car le meilleur ordonnancement obtenu avec cette variante de $\mathcal{R}_1$ a une valeur de 3688. Il est donc préférable de guider le choix du prochain échantillon à déplacer plutôt que de se référer au hasard.

Dans la Figure 7.19, nous avons représenté pour S* l'évolution du nombre d'échantillons en traitement au fur et à mesure que le temps avance. Nous observons que la partie de cet ordonnancement située entre les heures 03:04:56 et 06:28:49 définit un cycle C durant lequel 14 échantillons sont retirés de r(0) et 14 autres sont introduits dans r(m+1). Ainsi donc, si l'on exécutait notre algorithme avec une valeur de $\bar{n}$ supérieure à 50, nous obtiendrions un ordonnancement dans lequel le cycle C serait répété aussi souvent que possible. Il s'en suit que si le nombre d'échantillons à traiter pour cette application est très grand, la valeur de l'ordonnancement admissible complet pour traiter ces échantillons tendrait vers le temps moyen nécessaire pour traiter 14 échantillons dans le cycle C, c'est-à-dire (06:28:49- 03:04:56)/14 = 874 secondes.

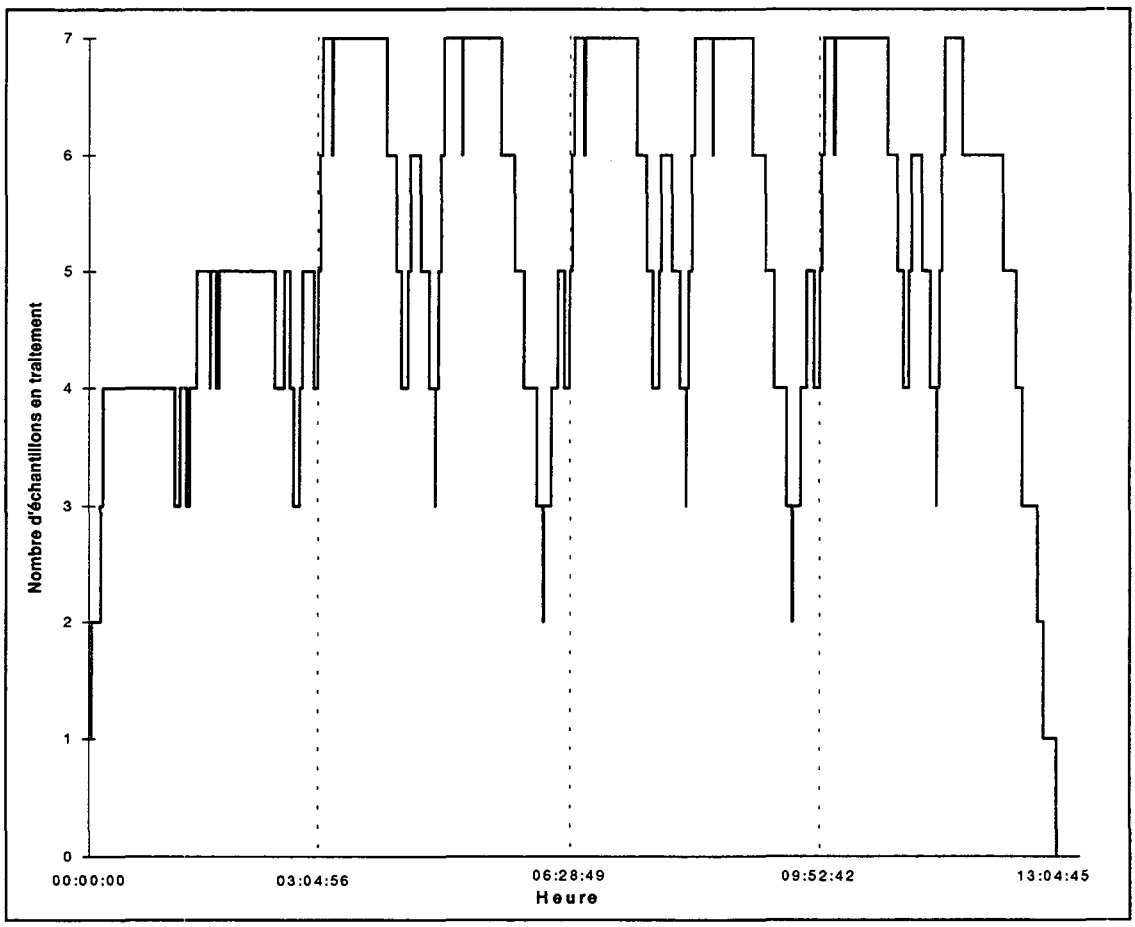


Figure 7.19 : Évolution du nombre d'échantillons en traitement en fonction du temps.

Il est important de relever qu'une borne inférieure de bonne qualité pour la valeur optimale d'un ordonnancement ne peut pas être déterminée facilement. Une façon de procéder pourrait consister à ignorer le robot (i.e. considérer tous les temps de déplacement du robot comme nuls). Mais en raison des contraintes du type (5) et du fait que les ressources ont une capacité limitée qui peut être plus grande que 1, ce problème simplifié n'est pas plus facile à résoudre que le problème original.

Dans le POTELR réel que nous avons étudié, la durée de la tâche t_0 n'est pas limitée dans le temps (i.e. $\bar{d}_0 = \infty$). Par conséquent, un ordonnancement admissible pour le traitement de n échantillons peut être obtenu en calculant S*(n, 1), c'est-à-dire en traitant complètement chaque échantillon l'un après l'autre. La valeur de cet ordonnancement est F(1, 1) = 4145. La valeur de S* est donc 4.4 fois plus petite que F(1, 1) qui est obtenue en n'autorisant qu'un seul échantillon dans le système. Cette amélioration importante s'explique principalement par le fait qu'il y a en moyenne 5.36 échantillons qui sont en traitement pour la solution S*. La Figure 7.20 représente le pourcentage du makespan de S* pendant lequel un nombre donné d'échantillons sont en traitement.

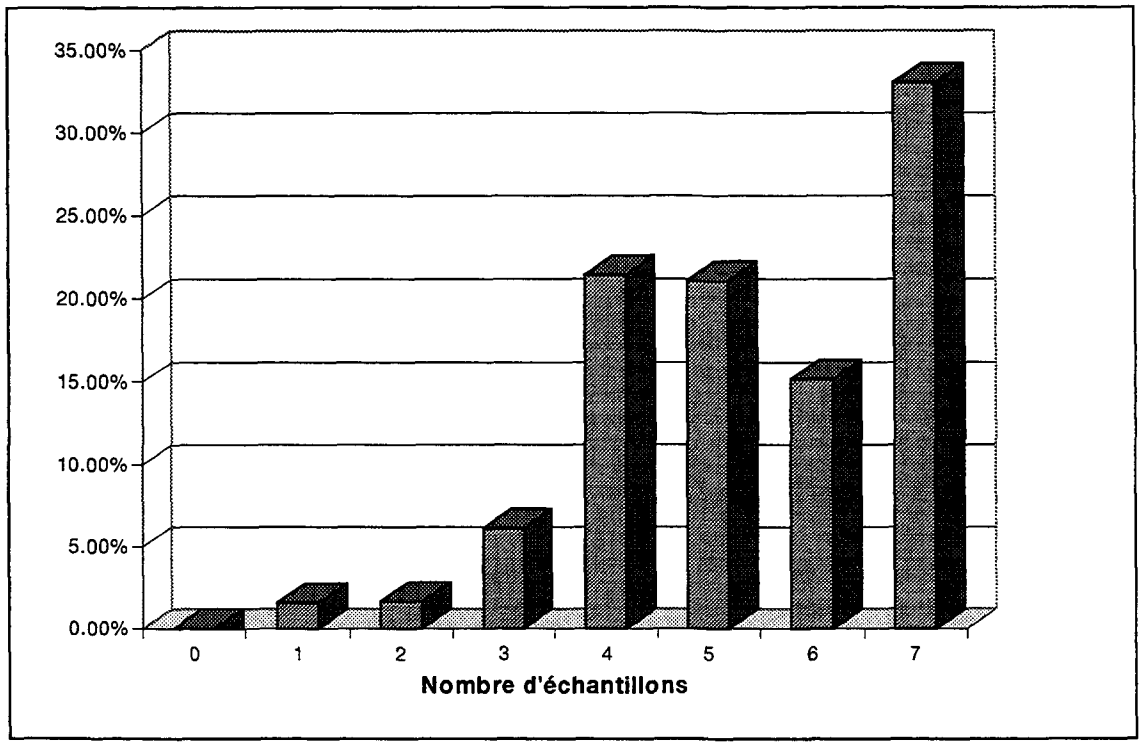


Figure 7.20 : Pourcentage du temps pendant lequel un nombre donné d'échantillons sont en traitement.

La qualité d'un ordonnancement pour le POTELR peut aussi être mesurée en calculant le pourcentage du makespan pendant lequel le robot est soit en attente soit en train de réaliser des déplacements à vide. Dans S*, ces pourcentages sont respectivement de 19.26 et 4.92 ce qui signifie que le taux d'occupation du robot est de 75.82 %. Pour comparaison, ces mêmes pourcentages pour S(1, 1) sont respectivement de 81.88 et 0.10 et donc le taux d'occupation du robot dans ce cas est de seulement 19.02 %. C'est dû au fait que chaque fois qu'un échantillon est introduit dans une ressource, le robot doit attendre devant cette ressource la fin du traitement de cet échantillon avant de pouvoir le déplacer dans une autre ressource.

7.3.2 Quelques expériences additionnelles

La partie de notre algorithme qui est la plus gourmande en temps de calcul est l'étape 4 où un problème de plus long chemin dans un graphe doit être résolu. Nous avons implanté une version simplifiée de notre heuristique dans laquelle les étapes 3 et 4 sont omises et où les étapes 5 et 6 sont remplacées par les points suivants :

(5) Ôter tous les échantillons β qui se trouvent dans V_α ;

(6) Poser $c := c - \sum_{\beta \in V_\alpha} t_\beta(S)$

Nous appellerons A_1 l'algorithme original et A_2 sa version simplifiée. Le temps de calcul est réduit de façon significative si l'on utilise A_2 au lieu de A_1. Par contre, la qualité des ordonnancements que l'on obtient avec A_2 diminue. En effet, en éliminant de l'ordre courant des opérations toutes celles qui concernent des échantillons créant des violations, nous créons des "trous" dans l'ordonnancement courant ce qui inévitablement nuit à la minimisation du makespan de l'application. Cependant, les solutions obtenues avec A_2 peuvent être améliorées si l'on résout un problème de plus long chemin à la fin de cet algorithme. Ainsi donc, la solution finale obtenue avec A_2 sera plus compacte et tous les temps d'attente inutiles du robot auront été éliminés. Nous avons donc implanté un algorithme additionnel A_3 qui est en quelque sorte un compromis entre A_1 et A_2. Cette nouvelle heuristique A_3 est identique à A_2 sauf pour les points suivants :

(7) **Si** $c = n \cdot T$
alors Aller à l'étape (8)
sinon Aller à l'étape (1) ;

(8) Soit O l'ordre des opérations du robot induit par S ;
Construire le graphe potentiels-tâches G associé à l'ordre O ;
Résoudre le problème de plus long chemin dans G et poser $S := S_G$;
STOP.

Comme nous l'avons déjà mentionné, nous utilisons l'algorithme de Bellman [Bel58] pour résoudre le problème de plus long chemin dans un graphe G. Le temps d'exécution de cet algorithme appartient à $O(|A| \cdot |V|)$ pour un graphe $G = (V, A)$. Dans notre cas, le nombre de sommets dans le graphe est proportionnel au nombre ν d'échantillons qui ne sont pas dans r(0) et le nombre d'arcs est proportionnel à ν. Ainsi la complexité de l'algorithme de Bellman est en $O(\nu^2)$ avec ν qui augmente de 0 à n. Selon nos expériences numériques, le nombre d'appels à la procédure PLUS LONG CHEMIN est proportionnel à n. Cela explique la complexité en $O(n^3)$ que nous avons observée pour A_1.

L'exécution des étapes (1) à (6) des algorithmes A_2 et A_3 requiert un temps de calcul en O(n) et ainsi ces deux méthodes ont une complexité $O(n^2)$, même si une exécution de A_3 demande un temps de calcul légèrement plus long qu'une exécution de A_2. Comme exemple, nous avons représenté dans la Figure 7.21 les temps de calcul nécessaires aux algorithmes A_1, A_2 et A_3 pour résoudre un POTELR où N = n avec n variant entre 1 et 50 et en utilisant la variante (a)-(g)-(iii) des règles $\mathcal{R}_1$ et $\mathcal{R}_2$. Ces tests numériques ont été réalisés sur une station de travail Silicon Graphics 200[Mhz].

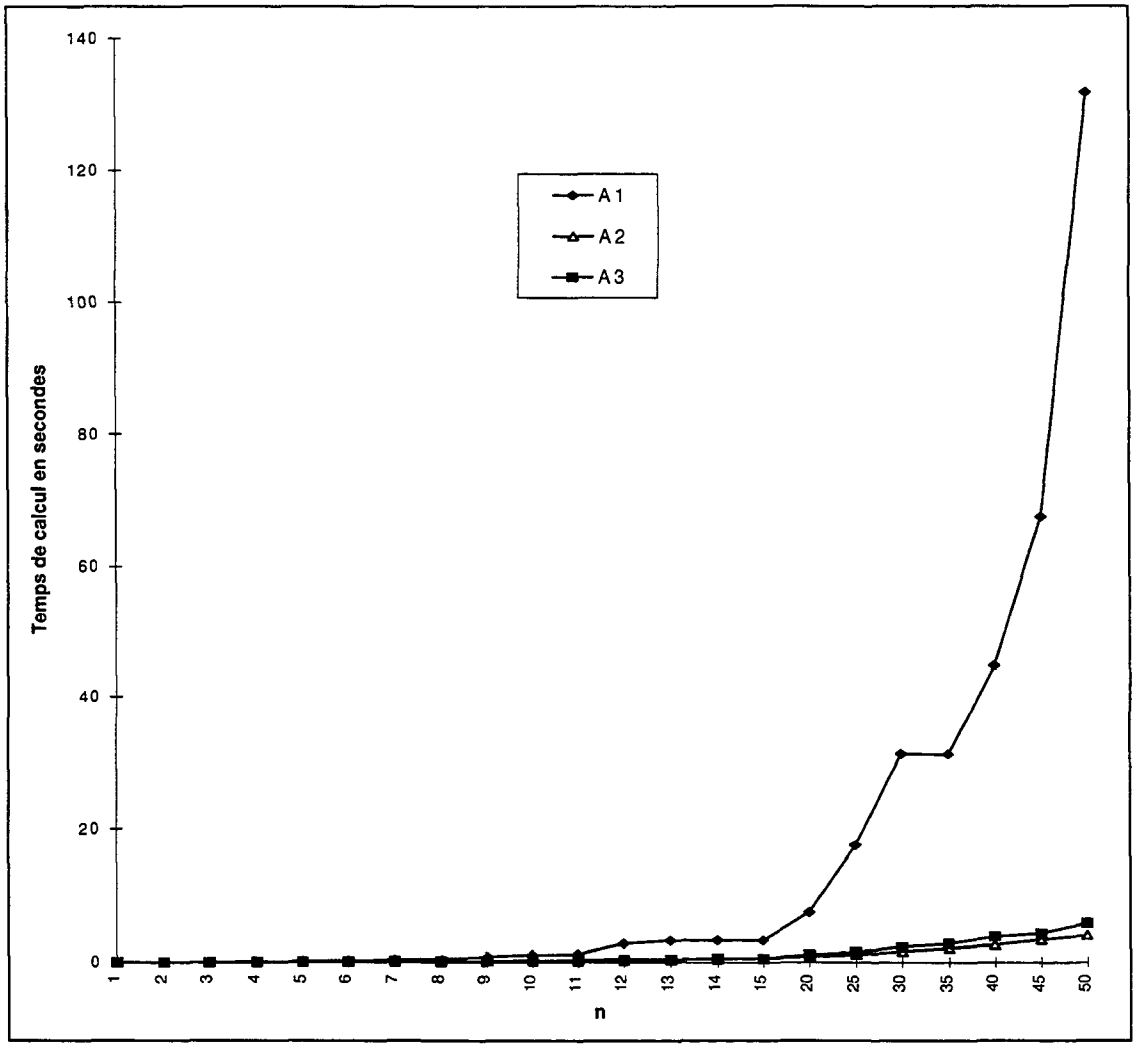


Figure 7.21 : Comparaison entre les trois algorithmes A_1, A_2 et A_3.

Le meilleur ordonnancement fourni par A_2 est obtenu en posant n = 50, en choisissant N supérieur à 10 et en utilisant la combinaison (b)-(g)-(iii). Cet ordonnancement, dont la valeur est de 965, peut être ensuite amélioré en résolvant un problème de plus long chemin. Un ordonnancement valant 945 est obtenu avec cette post-optimisation. Il correspond au meilleur ordonnancement qui peut être trouvé en utilisant A_3.

Ces nombreuses expériences numériques réalisées sur un seul exemple d'un POTELR permettent de se faire une bonne idée sur la ou les variantes des règles $\mathcal{R}_1$ et $\mathcal{R}_2$ qu'il faudrait utiliser pour résoudre une autre application. De plus, le POTELR qui a été utilisé pour nos expériences numériques peut être considéré comme un problème de référence pour les chercheurs qui seraient intéressés à développer d'autres méthodes heuristiques (ou exactes) pour ce problème particulier d'ordonnancement.

Afin de vérifier que les meilleures variantes des règles $\mathcal{R}_1$ et $\mathcal{R}_2$ que nous avons mentionnées ne sont pas dépendantes d'un problème particulier, nous avons réalisé des tests numériques supplémentaires sur un POTELR généré aléatoirement tout en conservant la plupart des caractéristiques du POTELR réel. Nous n'avons cependant pas considéré les variables $\mathcal{T}_{in}(i)$ et $\mathcal{T}_{out}(i)$ si bien que les temps de déplacement du robot peuvent être représentés par une unique matrice symétrique à diagonale nulle. Ce nouveau problème est décrit dans les tableaux suivants :

Ressource	Type d'activation	Capacité	Bloquante
1	Implicite	∞	Non
2	Implicite	3	Non
3	Explicite	10	Non
4	Implicite	1	Oui
5	Implicite	6	Non
6	Implicite	10	Non
7	Implicite	1	Oui
8	Implicite	6	Non
9	Implicite	6	Non
10	Implicite	10	Non
11	Explicite	3	Non
12	Implicite	∞	Non

Tableau 7-7 : Description des ressources du POTELR aléatoire.

Indice i de la tâche	Ressource r(i)	Durée minimale $\underline{d}_i$	Durée maximale $\overline{d}_i$	$L_{i,\,i+1}$
0	1	0	∞	∞
1	2	148	291	∞
2	7	29	41	∞
3	5	118	295	130
4	11	80	156	∞
5	9	26	141	154
6	4	34	184	∞
7	5	165	580	391
8	9	137	790	236
9	8	112	264	∞
10	4	47	130	∞
11	7	35	177	∞
12	10	35	63	∞
13	2	142	190	311
14	3	102	179	∞
15	6	19	29	∞
16	12	0	∞	∞

Tableau 7-8 : L'ensemble ordonné des tâches du POTELR aléatoire.

r(i) / r(j)	1	2	3	4	5	6	7	8	9	10	11	12
1	0	2	5	3	2	3	3	2	5	4	5	5
2	2	0	2	4	3	1	3	1	5	2	1	1
3	5	2	0	3	3	3	3	5	2	3	4	4
4	3	4	3	0	1	4	1	4	5	2	5	1
5	2	3	3	1	0	3	5	3	3	4	1	1
6	3	1	3	4	3	0	2	3	5	3	3	5
7	3	3	3	1	5	2	0	1	5	1	5	3
8	2	1	5	4	3	3	1	0	1	2	2	4
9	5	5	2	5	3	5	5	1	0	4	2	3
10	4	2	3	2	4	3	1	2	4	0	5	3
11	5	1	4	5	1	3	5	2	2	5	0	5
12	5	1	4	1	1	5	3	4	3	3	5	0

Tableau 7-9 : Temps de déplacement du robot (en secondes).

Pour ce problème aussi, les meilleurs ordonnancements ont été obtenus en combinant (g)-(iii) avec (a), (b), (c) ou (d). Les résultats de ces quatre combinaisons sont représentés dans la Figure 7.22. Le meilleur ordonnancement pour ce problème a une valeur de 285 et a été trouvé en posant N = 6 et en utilisant la variante (d)-(g)-(iii). Notons de plus que la valeur de ce meilleur ordonnancement est 4.5 fois plus petite que celle de S*(n, 1).

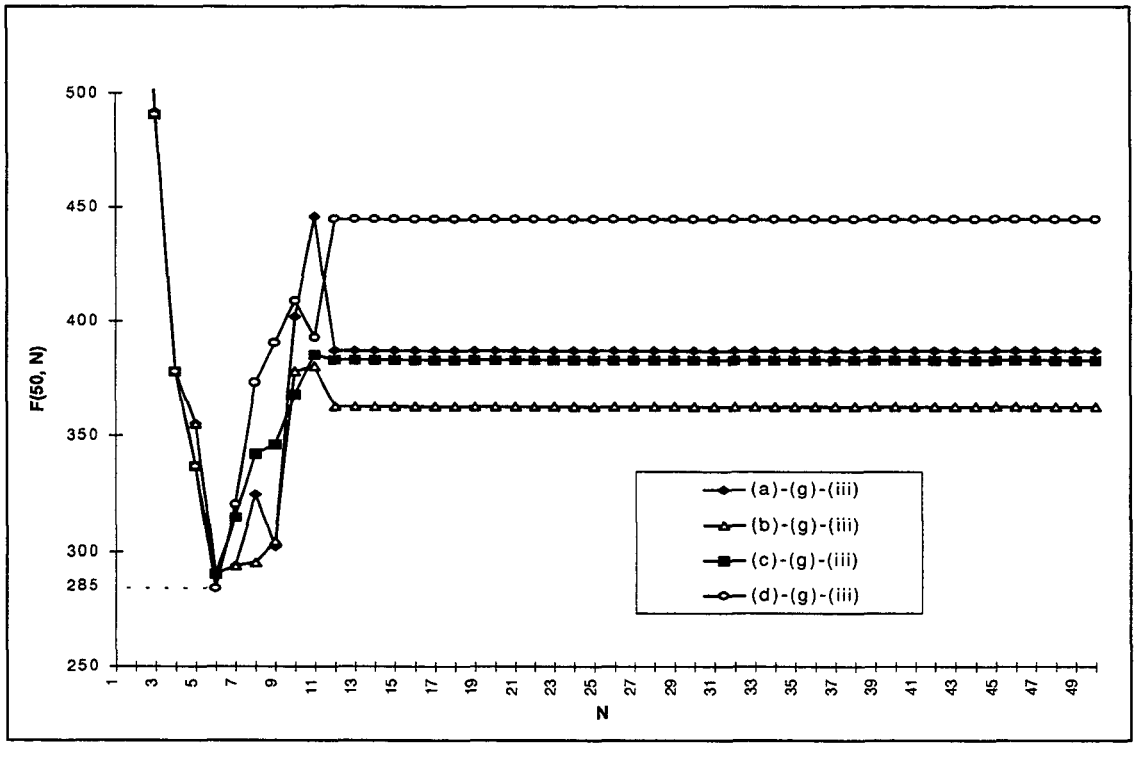


Figure 7.22 : Les résultats obtenus pour POTELR généré aléatoirement.

Nous aurions pu générer bien plus d'exemples aléatoires, mais ils auraient sûrement été très différents des POTELR réels et de plus, les résultats que nous aurions obtenus n'auraient pu être comparés à aucun autre.

7.4 Extensions de notre recherche et remarques finales

Le modèle et les algorithmes présentés dans ce chapitre sont flexibles. Ils peuvent en effet être étendus relativement aisément afin de tenir compte de contraintes additionnelles.

Par exemple, nous avons rencontré des configurations de laboratoire dans lesquelles une tâche pouvait être exécutée par plusieurs ressources. Dans ce type de situation, l'ensemble R des ressources du laboratoire se décompose en *classes* de ressources. A chacune de ces classes est associée une fonction chimique unique correspondant à l'un des rôles caractéristiques joué par une ressource dans le laboratoire chimique (incubation, agitation, etc.). Ainsi donc, les ressources qui appartiennent à une même classe sont identiques dans le sens qu'elles peuvent effectuer la même fonction chimique mais par contre, elle peuvent très bien avoir des caractéristiques physiques (par exemple leur capacité) différentes. Si R_i représente la classe des ressources dans lesquelles la tâche t_i peut être exécutée, nous définissons par $r(\alpha, i)$ (au lieu de r(i)) la ressource de R_i dans laquelle la tâche t_i de l'échantillon α se déroule. De plus, supposons que S soit un ordonnancement partiel admissible, α un échantillon quelconque et k l'indice de la tâche en cours d'exécution pour cet échantillon. Lorsque l'échantillon α est ôté de $r(\alpha, k)$, il doit être introduit dans une ressource libre de R_{k+1}, par exemple la plus proche de $r(\alpha,k)$. Il faudrait alors modifier les définitions des variables H_1 et H_2 (voir section 7.2.2) en remplaçant $w_{r(k),r(k+1)}$ par :

$$\min_{r_u \in R_{k+1}} w_{r(\alpha,k),r_u}$$

Les autres changements à apporter à notre méthode pour tenir compte des classes de ressources sont évidentes. Par exemple, si R(S) n'est pas une ressource bloquante, l'ensemble A des échantillons candidats pour la prochaine opération du robot est défini comme l'ensemble des échantillons α pour lesquels au moins une ressource de R_{q_α} est libre.

Nous pouvons aussi facilement prendre en compte une particularité supplémentaire des ressources : la *réservation* d'une de ses places de traitement pendant un certain intervalle de temps. C'est une caractéristique spécifique d'une ressource qui peut être soit vraie (ressource avec réservation) soit fausse (ressource sans réservation). Considérons par exemple une ressource du type "visseur-dévisseur" où cette information

est vraie. Alors, la place de traitement de cette ressource dans laquelle on aurait ôté le bouchon d'un échantillon α est réservée pour cet échantillon (aucune autre tâche d'un échantillon $\beta \neq \alpha$ ne peut s'y dérouler) tant que ce même bouchon n'a pas été restitué à α. La seule modification à apporter à notre modélisation pour gérer cette nouvelle contrainte est de redéfinir le concept d'une ressource libre : nous dirons qu'une ressource r_u est libre pour un échantillon α s'il existe au moins une place dans r_u qui ne soit pas occupée ni réservée pour un échantillon $\beta \neq \alpha$.

7.4.1 Améliorations possibles de CASSIRAS

Dans notre méthode nous avons considéré que tout échantillon actif qui induit une opération admissible du robot est un candidat potentiel qui est introduit dans A, pour autant que le cardinal de A ne soit pas supérieur à N. Afin d'optimiser CASSIRAS, il serait préférable de ne considérer qu'un seul échantillon candidat par tâche et par ressource, comme nous l'avons fait pour r(0). C'est à dire que si plusieurs échantillons dont la prochaine tâche est la même se trouvent dans la même ressource, seul l'échantillon entré en premier dans cette ressource est ajouté à l'ensemble A des échantillons candidats. Cette modification permettrait un gain de temps dû au plus petit nombre de tests effectués lors de l'évaluation des opérations admissibles. Nous avons en effet observé que beaucoup d'entre elles étaient jugées équivalentes (par la règle $\mathcal{R}_1$) dans la première version de notre heuristique.

Lors des tests des trois variantes de la règle $\mathcal{R}_2$ qui choisit l'échantillon β à supprimer d'un ordonnancement partiel S_α non admissible, c'est toujours la définition (iii) basée sur l'ensemble V_α qui a donné les meilleurs résultats. Nous pensons donc qu'il pourrait être intéressant d'affiner cette règle. La modification principale que l'on pourrait apporter à $\mathcal{R}_2$ serait de choisir β uniquement parmi l'ensemble V_α des échantillons qui se trouvent en violation et non plus aussi parmi l'ensemble D des échantillons actifs du système (voir section 7.2.5.2). De cette façon, seuls les échantillons qui violent au moins l'une des contraintes chimiques de leur processus analytique dans S_α seraient considérés. La nouvelle règle $\mathcal{R}_2$ pourrait donc être :

(i) choisir un échantillon $\beta \in V_\alpha$ dont la durée de la violation est la plus grande
(ii) choisir un échantillon $\beta \in V_\alpha$ dont le rang de la violation est la plus petite
(iii) choisir aléatoirement un échantillon $\beta \in V_\alpha$

En cas d'égalité entre plusieurs échantillons candidats, un choix aléatoire serait effectué.

7.4.2 Conclusion

Afin d'accroître la productivité ainsi que la flexibilité de nos méthodes, il serait intéressant d'élargir le cadre des recherches que nous avons menées pour être en mesure de gérer :

- des systèmes matériels plus complexes contenant, par exemple, des ressources de stockage intermédiaire entre l'entrée et la sortie du laboratoire,
- plus qu'un seul robot,
- des tâches préemptives,
- des échantillons prioritaires,
- plusieurs applications simultanément.

Notons cependant que la version actuelle de notre heuristique a été implantée dans la pratique pour le compte d'une société de la région lausannoise et qu'elle a prouvé son efficacité.

8. Une approche évolutive pour le POTELR

8.1 Introduction

Ce dernier chapitre a pour objectif de présenter une méthode évolutive pour résoudre le POTELR présenté dans le chapitre précédent. L'heuristique CASSIRAS a deux atouts majeurs qui sont d'une part, sa rapidité d'exécution et d'autre part, son efficacité qui a été démontrée dans la pratique. Malheureusement, la qualité des solutions produites par cet algorithme constructif dépend fortement des variantes retenues parmi les définitions des règles $\mathcal{R}_1$ et $\mathcal{R}_2$ (voir section 7.2.5) ainsi que de la valeur N de la taille maximale autorisée pour l'ensemble A des échantillons candidats (voir 7.2.5.1). La détermination de bonnes valeurs pour ces paramètres de contrôle de l'algorithme CASSIRAS est une tâche qui peut s'avérer fastidieuse et demander une bonne connaissance du POTELR que l'on désire résoudre. De plus, notons que même si de bonnes valeurs pour ces paramètres de contrôle ont pu être déterminées pour une certaine application chimique, il n'est pas évident que celles-ci soient toujours adéquates pour d'autres applications chimiques.

Il est aussi important de noter qu'une fois que les valeurs des paramètres de contrôle ont été choisies, elles restent fixes durant toute l'exécution de la méthode CASSIRAS. C'est un désavantage de notre première approche car on peut facilement envisager que l'utilisation d'une combinaison différente des définitions des règles $\mathcal{R}_1$ et $\mathcal{R}_2$, à un certain stade de l'évolution du processus de construction d'une solution, pourrait permettre à notre méthode de trouver une solution finale de meilleure qualité.

C'est donc dans l'optique de remédier aux désavantages précités de CASSIRAS que nous avons développé une nouvelle méthode robuste, basée sur CASSIRAS et gouvernée par un algorithme génétique, pour la résolution du POTELR. Dans notre cas, l'algorithme génétique (désigné ci-après par AG) joue le rôle d'un système expert dont l'objectif est de choisir, à chaque étape de l'heuristique constructive de base, les valeurs des paramètres de contrôle de cette heuristique qui sont les plus appropriées pour la

situation courante. Une telle méthode est transparente aux yeux de son utilisateur. En effet, il n'a plus à choisir les valeurs des paramètres de contrôle mais simplement à indiquer le nombre n d'échantillons à ordonnancer. De surcroît, cet algorithme est évolutif et s'adapte de manière dynamique aux nouvelles applications chimiques que l'on pourrait lui donner à résoudre.

Ce chapitre est organisé de la manière suivante. La section 8.2 rappelle succinctement les fondements des algorithmes génétiques puis, dans la section 8.3, nous présentons dans le détail notre AG d'Ordonnancement pour un laboratoire Robotisé d'Analyses chimiques (ci-après désigné par AGORA). La section 8.4 indique les résultats numériques obtenus avec cette nouvelle technique. De plus, nous montrons que cette approche évolutive domine très nettement l'heuristique CASSIRAS au niveau de la qualité des résultats fournis. Finalement, nous indiquons dans la section 8.5 comment cette nouvelle approche peut être utilisée avec n'importe quelle méthode constructive dont le nombre d'étapes est fini et où les décisions prises à chacune de ces étapes sont basées sur des ensembles déterminés de règles.

8.2 Les fondements des algorithmes génétiques

Les algorithmes génétiques sont des méthodes évolutives (voir section 2.3.3) d'exploration d'un espace S, par exemple celui des solutions d'un problème d'optimisation combinatoire, qui reposent sur les principes de la sélection naturelle des êtres vivants et de la génétique. Ils utilisent simultanément les principes de la survie des structures les mieux adaptées à leur environnement et les échanges pseudo-aléatoires d'information pour constituer un algorithme de recherche.

Les êtres artificiels ou individus (solutions d'un problème d'optimisation combinatoire) qui constituent la population $\mathcal{P}$ d'un algorithme génétique sont chacun décrits à l'aide d'une chaîne de caractères (à l'origine uniquement des 0 et des 1) de longueur finie appelée *génome*. Ce codage d'une solution à l'aide d'un génome est le pendant du stockage des facteurs héréditaires d'un être vivant dans un chromosome.

A l'origine, ces méthodes ont été développées par Holland [Hol75] pour des applications dans le domaine de la biologie. Elles furent ensuite adaptées à des contextes des plus divers. Dans cette section, nous nous contentons de présenter les mécanismes d'un algorithme génétique simple. Le lecteur intéressé par les raffinements et les détails de ce type de méthodes se référera à [Gol89, Raf95].

Le processus itératif qui est à la base d'un AG simple est constitué de deux phases qui se succèdent à tour de rôle. Chaque itération de ces deux phases est appelée une *génération.* La première phase est une *phase de coopération* des individus de la population $\mathcal{P}$, gouvernée par un *opérateur de reproduction* et un *opérateur de combinaison.* Dans cette phase, les génomes sont d'abord évalués puis les mieux adaptés au problème étudié sont combinés deux à deux dans le but de créer des nouvelles solutions inédites dont la qualité moyenne s'améliore génération après génération. Pour ce faire, les caractéristiques prédominantes des meilleures solutions de la population sont retenues (lors de la combinaison de deux génomes) à l'aide de règles probabilistes générales.

La seconde phase du processus itératif est une phase d'*adaptation individuelle* des génomes à leur environnement qui fait appel à un *opérateur de mutation.* Les mutations sont des variations ou modifications aléatoires apportées aux génomes de manière indépendante. Il n'y a aucune interaction entre les individus de la population artificielle au cours de cette phase. Au terme de chaque phase d'adaptation individuelle, une nouvelle population de solutions est créée. La taille de cette nouvelle population est généralement identique à celle de la population initiale et elle est obtenue en utilisant les caractéristiques prédominantes des meilleurs individus de la génération précédente ainsi que des caractéristiques nouvelles. Même si les AG se servent du hasard, ils ne sont pas purement aléatoires. Ils exploitent efficacement l'information récoltée au cours de la recherche pour se diriger vers des régions prometteuses de l'espace S, avec l'espoir d'améliorer la qualité des solutions de $\mathcal{P}$.

Dans le cadre de notre étude, nous allons considérer un ensemble S qui sera l'espace des solutions admissibles S du POTELR. Ainsi, chaque $S \in S$ est un ordonnancement complet admissible dont la valeur, définie en section 7.3.1, est f(S). Le problème d'optimisation combinatoire que nous devons résoudre est la détermination d'un ordonnancement $S^* \in S$ dont la valeur f(S*) est minimum. Comme nous l'avons déjà évoqué dans la section 2.3.3, les AG sont différents des méthodes de recherche usuelles rencontrées en optimisation combinatoire pour les raisons suivantes :

- ils utilisent un codage des solutions $S \in S$ et non les solutions elles-mêmes,
- ils se basent sur des règles de décision probabilistes pour guider la recherche,
- l'évolution de la recherche progresse sur la base d'une population de solutions plutôt qu'à partir d'une solution unique,
- seules les valeurs f(S) des solutions $S \in S$ sont utilisées au cours de la recherche.

En résumé, un AG simple est composé de trois opérateurs qui sont appliqués séquentiellement à la population courante et itérativement de génération en génération. Sa structure peut se résumer comme suit :

Opérateurs génétiques	Type d'opération	Type de phase
Reproduction } **Combinaison** }	Opérations principales	Coopération
Mutation	Opération secondaire	Adaptation individuelle

Tableau 8-1: La structure d'un algorithme génétique.

Ce type d'algorithme s'avère d'une grande utilité dans la résolution de problèmes difficiles d'optimisation combinatoire et en particulier pour l'ordonnancement des traitements analytiques d'échantillons dans un laboratoire robotisé.

8.3 Description de l'algorithme évolutif

Notre nouvelle approche est basée sur les études de Fisher et Thompson [Fis63], O'Grady et Harrison [Ogr85] ainsi que celle de Dorndorf et Pesch [Dor95] qui sont toutes trois relatives au problème du jobshop.

Les notations utilisées pour décrire le POTELR sont celles qui ont été définies dans le chapitre précédent. Celles que nous utiliserons pour décrire notre AG sont les suivantes :

- $\mathcal{P}$: la population des solutions $S \in \mathcal{S}$,
- $|\mathcal{P}|$: la taille de la population $\mathcal{P}$,
- G_i : le génome représentant la $i^{ième}$ solution S_i ($1 \leq i \leq |\mathcal{P}|$) de la population $\mathcal{P}$,
- $\mathcal{A}$: un alphabet fini d'où proviennent les caractères qui composent les génomes,
- I : le nombre de générations de l'algorithme génétique,
- E : l'entropie de la population,
- p_m : la probabilité de mutation.

Caractérisation d'un génome pour le POTELR

Une décision locale pour le POTELR est le choix de la prochaine opération du robot à effectuer. Une solution de notre problème d'ordonnancement peut donc être décrite par une séquence finie de règles de choix de ces décisions locales. Ainsi, chaque solution $S_i \in \mathcal{P}$ peut être représentée par un génome $G_i = (g_{i1}, g_{i2}, \ldots, g_{ij}, \ldots, g_{i\eta})$ à η composantes. La valeur η est le nombre total, n·T, d'opérations dans l'application chimique à ordonnancer. Une composante $g_{ij} \in [1, 2, \ldots, \mu]$ est le numéro d'une règle de choix d'une décision locale pour le POTELR. Chacun de ces μ numéros représente un *caractère* de $\mathcal{A}$ correspondant à une combinaison possible des définitions des règles $\mathcal{R}_1$ et $\mathcal{R}_2$. En d'autres termes, g_{ij} indique quelle variante des règles $\mathcal{R}_1$ et $\mathcal{R}_2$ il faut utiliser dans l'algorithme CASSIRAS pour réaliser la $j^{ième}$ opération de l'application chimique. Chaque génome G_i est donc un codage, en une chaîne de caractères de longueur finie η, des paramètres de contrôle de CASSIRAS, qui permet d'obtenir l'ordonnancement admissible complet $S_i \in \mathcal{P}$.

Le Tableau 8-2 décrit les $\mu = 16$ règles de décision locale que nous avons retenues parmi les 39 variantes possibles des règles $\mathcal{R}_1$ et $\mathcal{R}_2$.

Numéro de règle de décision locale	Règle $\mathcal{R}_1$		Règle $\mathcal{R}_2$
	Critère 1	Critère 2	
1	a	f	i
2	b	f	i
3	c	f	i
4	d	f	i
5	a	g	i
6	b	g	i
7	c	g	i
8	d	g	i
9	a	f	iii
10	b	f	iii
11	c	f	iii
12	d	f	iii
13	a	g	iii
14	b	g	iii
15	c	g	iii
16	d	g	iii

Tableau 8-2 : Les règles de décision locale.

De manière à tenir compte d'un des ingrédients additionnels de CASSIRAS (voir section 7.2.5.1) nous modifions légèrement les caractéristiques d'un génome G_i en ajoutant une composante g_{i0}. Celle-ci représente le paramètre N de notre algorithme constructif. Ainsi, si l'on désire ordonnancer une application chimique pour un ensemble de n échantillons, g_{i0} prendra ses valeurs dans l'ensemble {1, 2, ..., n}.

Un génome G_i peut donc être défini par le vecteur à $\eta+1$ composantes illustré dans le Tableau 8-3 suivant :

	0	1	...	j	...	η
G_i :	$g_{i0} \in [1, ..., n]$	$g_{i1} \in [1, ..., \mu]$	...	$g_{ij} \in [1, ..., \mu]$	...	$g_{i\eta} \in [1, ..., \mu]$

Tableau 8-3 : Représentation d'un génome.

Opérateur de reproduction

La reproduction est un processus au cours duquel chaque génome G_i est copié un nombre de fois proportionnel à la valeur $f(S_i)$ de la solution S_i de la population courante qu'il représente. Meilleure est la valeur de cette solution, plus G_i aura une probabilité élevée de contribuer à la génération suivante de solutions. Ce procédé, appelé *survie des génomes les mieux adaptés*, est une version artificielle de la sélection naturelle dans laquelle les plus mauvaises solutions disparaissent et les meilleures sont reproduites.

Lorsque l'on traite un problème de maximisation, la façon usuelle d'implanter un opérateur de reproduction est de définir une probabilité de sélection $p(G_i)$ de chaque génome G_i comme le ratio entre la valeur $f(S_i)$ et la somme des valeurs des solutions de la population $\mathcal{P}$. Cependant, comme notre POTELR est un problème de minimisation, nous définissons $p(G_i)$ différemment afin que les génomes les mieux adaptés soient ceux dont les ordonnancements associés ont des petites valeurs. Nous avons besoin d'une fonction $p(G_i)$ qui soit monotone décroissante avec $f(S_i)$. Si l'on dénote par f_{max} la valeur de la plus mauvaise solution appartenant à $\mathcal{P}$, la probabilité de sélection que nous utilisons est la suivante :

$$p(G_i) = \frac{f_{max} - f(S_i)}{|\mathcal{P}| \cdot f_{max} - \sum_{S \in \mathcal{P}} f(S)} \geq 0$$

Nous calculons ensuite le nombre espéré e_i de copies de chaque génome comme étant $p(G_i) \cdot |\mathcal{P}|$. Avec ce procédé, chaque génome est reproduit un nombre de fois égal à la partie entière de e_i. De plus, une copie additionnelle est créée avec une probabilité égale à la partie fractionnaire de e_i. Ce processus, connu dans la littérature sous le nom *d'échantillonnage stochastique de la partie restante sans remplacement* [Gol89], est répété jusqu'à ce que le nombre total de génomes reproduits soit égal à $|\mathcal{P}|$.

Opérateur de combinaison

L'opérateur de combinaison produit des nouveaux génomes en combinant entre eux les génomes créés à l'aide de l'opérateur de reproduction. C'est une phase de coopération, réalisée en deux étapes, dans laquelle les génomes échangent de l'information. Une façon simple de mettre en œuvre cette phase de combinaison consiste tout d'abord à former des paires de *génomes-parents* (on suppose $|\mathcal{P}|$ pair) puis dans un second temps, à créer deux nouveaux *génomes-enfants* pour chaque couple de génomes-parents. L'opérateur de combinaison le plus simple rencontré dans la littérature est illustré dans la Figure 8.1.

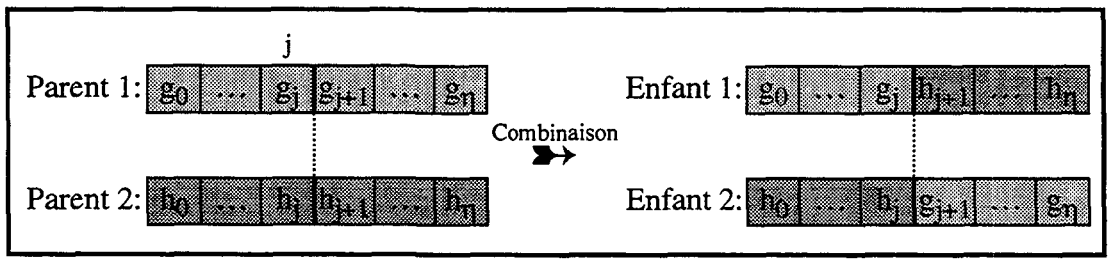


Figure 8.1 : Opérateur de combinaison le plus simple.

Cet opérateur de combinaison choisit aléatoirement l'indice $j \in \{0, \ldots, \eta-1\}$ d'une composante d'un génome et échange tous les caractères stockés dans les composantes des génomes-parents dont les indices sont compris entre $j+1$ et η inclus. Cet opérateur est appelé *opérateur de combinaison à un point.*

Dans le cadre de notre étude, nous nous sommes intéressés à un opérateur de combinaison uniforme introduit par Syswerda [Sys89]. Celui-ci nécessite la définition d'un vecteur binaire auxiliaire $\mathcal{M}$, appelé *masque d'échange* et de taille identique à celle des génomes, pour créer deux génomes-enfants à partir de deux génomes-parents. Les caractères contenus dans les composantes $j \in \{0, \ldots, \eta\}$ des génomes-parents sont recopiés sans échange dans les composantes j respectives des génomes-enfants si $\mathcal{M}_j = 0$ ou avec échange si $\mathcal{M}_j = 1$ (voir Figure 8.2).

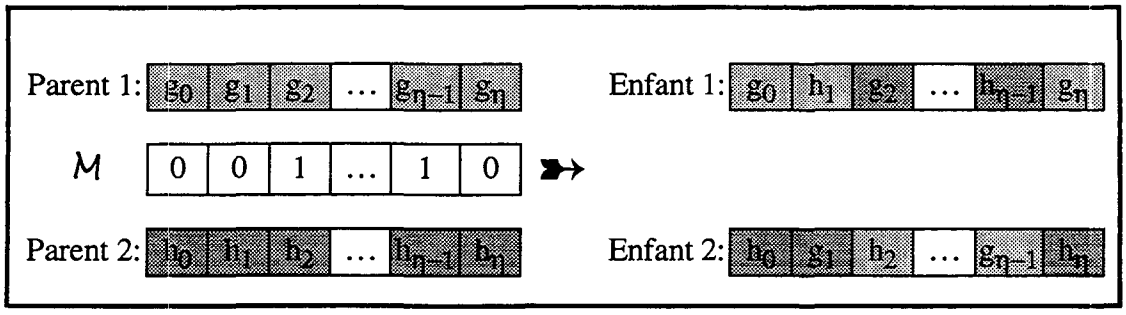


Figure 8.2 : Opérateur de combinaison uniforme.

Généralement, les valeurs binaires des composantes de M sont générées aléatoirement avec une probabilité ½ . Dans son étude, Syswerda donne des résultats théoriques et empiriques qui montrent la supériorité de cet opérateur de combinaison par rapport à l'opérateur de combinaison à un point.

Pour notre part, il nous a paru intéressant de généraliser l'opérateur uniforme de Syswerda en utilisant un nouvel opérateur de combinaison : l'*opérateur de combinaison uniforme généralisé*. Cette opérateur crée deux génomes-enfants dont les composantes proviennent du meilleur des deux génomes-parents utilisés pour la combinaison avec une probabilité qui peut être supérieure à ½ . Plus précisément, supposons que nous avons deux génomes $\mathcal{G} = (g_0, g_1, \ldots, g_j, \ldots, g_\eta)$ et $\mathcal{H} = (h_0, h_1, \ldots, h_j, \ldots, h_\eta)$ représentant deux solutions $S_\mathcal{G}$ et $S_\mathcal{H} \in P$, $S_\mathcal{G}$ étant la meilleure des deux (i.e. $f(S_\mathcal{G}) \leq f(S_\mathcal{H})$). Notre opérateur de combinaison uniforme généralisé va créer deux nouveaux génomes $\mathcal{G}' = (g'_0, g'_2, \ldots, g'_j, \ldots, g'_\eta)$ et $\mathcal{H}' = (h'_0, h'_1, \ldots, h'_j, \ldots, h'_\eta)$ où les composantes g'_j et h'_j ($j \in [0, \ldots, \eta]$) sont déterminées avec une probabilité biaisée p_b (probabilité $\geq$ ½ qui dépend des valeurs relatives $f(S_\mathcal{G})$ et $f(S_\mathcal{H})$) comme suit :

$$g'_j = \begin{cases} g_j \text{ avec probabilité } & p_b \\ h_j \text{ avec probabilité } & 1-p_b \end{cases} \quad \text{et} \quad h'_j = \begin{cases} g_j \text{ avec probabilité } & p_b \\ h_j \text{ avec probabilité } & 1-p_b \end{cases}$$

Si nous calculons la différence en pourcentage $\mathcal{D}$ entre $f(S_\mathcal{G})$ et $f(S_\mathcal{H})$ comme $100 \cdot ((f(S_\mathcal{H}) - f(S_\mathcal{G})) / f(S_\mathcal{H})$, nous définissons p_b de la manière suivante :

$$p_b = \begin{cases} 0.5 & \text{si } 0 \leq \mathcal{D} < 2 \\ 0.6 & \text{si } 2 \leq \mathcal{D} < 7 \\ 0.7 & \text{si } 7 \leq \mathcal{D} < 12 \\ 0.8 & \text{si } 12 \leq \mathcal{D} < 15 \\ 0.9 & \text{sinon} \end{cases}$$

Opérateur de mutation et entropie de la population

La phase de mutation est une étape d'adaptation individuelle qui introduit, avec une très faible probabilité p_m, des transformations locales aléatoires des génomes obtenus après la phase de combinaison. Généralement, p_m reste constante tout au long de l'algorithme. Même si c'est un opérateur secondaire, la mutation ne doit pas pour autant être négligée. La principale raison est que cet opérateur permet à l'algorithme génétique d'explorer des nouvelles régions prometteuses de l'espace des solutions et parfois évite que le processus ne se bloque dans un optimum local de la fonction objectif. De plus, l'opérateur de mutation peut être bénéfique en réintroduisant de la diversité dans une population qui *converge prématurément*, c'est-à-dire qui contient un pourcentage élevé de solutions identiques.

Dans le double but d'éviter cet inconvénient et de contrôler la convergence de la population lors de l'exécution de notre AG, nous utilisons une probabilité de mutation p_m dynamique qui peut varier entre p_m^{min} et p_m^{max} en fonction de l'*entropie* E de la population $\mathcal{P}$. Dans une étude récente, Fogarty [Fog89] a montré que la qualité des résultats fournis par un AG peut être sensiblement améliorée si l'on fait varier p_m de façon stratégique. L'entropie $E \in [0, 1]$ est une mesure de la diversité de la population $\mathcal{P}$ qui indique à quel stade de convergence se trouve $\mathcal{P}$. Une valeur de 0 signifie que toutes les solutions de $\mathcal{P}$ sont identiques alors qu'une valeur de 1 indique une grande diversité dans $\mathcal{P}$. Cette idée a été utilisée par Fleurent [Fle94] pour le problème de l'affectation quadratique et par Costa et al. [Cos95] pour des problèmes de coloration de graphes.

Dans notre cas et avec la convention que $0 \cdot \log 0 = 0$, nous commençons par définir l'entropie $E_j \in [0, 1]$ d'une opération j ($j = 0, 1, \ldots, \eta$) comme suit :

$$E_j = \begin{cases} \dfrac{-\sum_{k=1}^{\mu} \frac{c_{jk}}{|\mathcal{P}|} \cdot \log(\frac{c_{jk}}{|\mathcal{P}|})}{\log(\mu)} & j = 1, \ldots, \eta \\ \dfrac{-\sum_{N=1}^{n} \frac{c_{jN}}{|\mathcal{P}|} \cdot \log(\frac{c_{jN}}{|\mathcal{P}|})}{\log(n)} & j = 0 \end{cases}$$

où c_{jk} ($j > 0$) représente le nombre de fois que le numéro k d'une règle de décision locale est présent dans la composante j des génomes de $\mathcal{P}$. Lorsque $j = 0$, c_{0N} indique le nombre de fois que le nombre maximum autorisé d'échantillons en traitement est égal à N dans les génomes de $\mathcal{P}$.

L'entropie de la population peut alors être définie comme :

$$E = \frac{\sum_{j=0}^{\eta} E_j}{\eta + 1}$$

Nous pouvons utiliser une telle mesure pour contrôler la valeur de la probabilité de mutation. Dans notre étude, nous avons fixé la valeur de p_m^{min} à 0.001 (comme cela se fait usuellement dans la littérature) et testé différentes valeurs pour p_m^{max} : 0.030, 0.025, ..., 0.005. De cette façon, p_m varie dynamiquement au cours de l'exécution de notre AG comme indiqué ci-dessous :

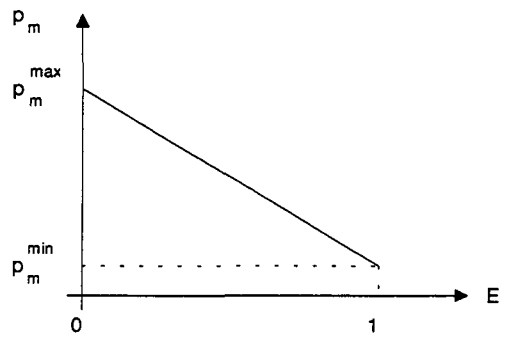
$p_m = (p_m^{min} - p_m^{max}) \cdot E + p_m^{max}$	

Tableau 8-4 : Variation de la probabilité de mutation.

8.4 Résultats numériques

Dans cette section, nous analysons le comportement de notre approche évolutive et nous la comparons avec l'heuristique CASSIRAS. Ces deux méthodes ont été testées sur le POTELR réel et le POTELR aléatoire décrits dans le chapitre précédent. Nous nous référerons respectivement à P_REAL et P_RANDOM lorsque nous évoquerons ces deux problèmes par la suite.

En ce qui concerne CASSIRAS, les 39 variantes possibles des règles $\mathcal{R}_1$ et $\mathcal{R}_2$ ont été testées. Pour chacune d'entre elles, nous avons résolu P_REAL et P_RANDOM avec n = 1, 2, ..., 15, 20, ..., 50 et N = 1, 2, ..., n, ce qui représente 365 tests par variante. Les 14'235 ordonnancements obtenus pour chaque POTELR ont été triés par ordre croissant en fonction de leur valeur, puis les 100 meilleurs ont été retenus. Nous rappelons au lecteur que le meilleur ordonnancement trouvé pour P_REAL a une valeur

de 942 (en utilisant n = 50, N = 7 et les variantes (a)-(g)-(iii) ou (b)-(g)-(iii)) et celui obtenu pour P_RANDOM a une valeur de 285 (en employant n = 50, N = 6 et la variante (d)-(g)-(iii)).

Pour AGORA, les paramètres suivants ont été utilisés :

- n = 5, 10, ..., 25
- $|\mathcal{P}|$ = 10, 20, ..., 100
- p_m^{max} = 0.005, 0.010, ..., 0.030
- μ = 16 (toutes les règles décrites dans le Tableau 8-2 sont utilisées), 8 (seules les règles numéro 9 à 16 du Tableau 8-2 sont employées) et 4 (on se sert uniquement des règles numéro 13 à 16 du Tableau 8-2 où le second critère de $\mathcal{R}_1$ est (h) en lieu est place de (g)).

Ces paramètres ont été choisis de manière relativement arbitraire, sans procéder à une détermination spécifique de leurs valeurs. Par conséquent, il est fort probable que les résultats obtenus par AGORA puissent être encore améliorés, par exemple en choisissant différemment la probabilité biaisée p_b de l'opérateur de combinaison uniforme généralisé ou en utilisant un opérateur de reproduction plus spécifique que l'échantillonnage stochastique de la partie restante sans remplacement.

De façon similaire aux expériences numériques réalisées pour CASSIRAS, les 900 ordonnancements obtenus en combinant les différents paramètres d'AGORA ont été triés par ordre croissant en fonction de leur valeur, puis les 100 meilleurs ont été retenus. Les meilleurs ordonnancements pour P_REAL et P_RANDOM ont respectivement des valeurs de 876 (en utilisant n = 20, p = 90, $\mathcal{I}$ = 200, p_m^{max} = 0.025, μ = 8) et 266 (avec n = 15, p = 20, $\mathcal{I}$ = 200, p_m^{max} = 0.020, μ = 8). Cela représente des améliorations de 7% et 6.7% par rapport aux meilleurs valeurs connues des ordonnancements fournis par CASSIRAS pour P_REAL et P_RANDOM respectivement.

Les Figures 8.3 et 8.4 illustrent les valeurs des 100 meilleurs ordonnancements obtenus respectivement pour P_REAL et P_RANDOM à l'aide de CASSIRAS et d'AGORA. En observant ces deux figures, nous remarquons clairement que notre nouvelle méthode domine nettement CASSIRAS et fournit des résultats plus robustes. Relevons aussi que la qualité du 100^e meilleur ordonnancement obtenu avec AGORA est supérieure à celle du meilleur ordonnancement trouvé avec CASSIRAS.

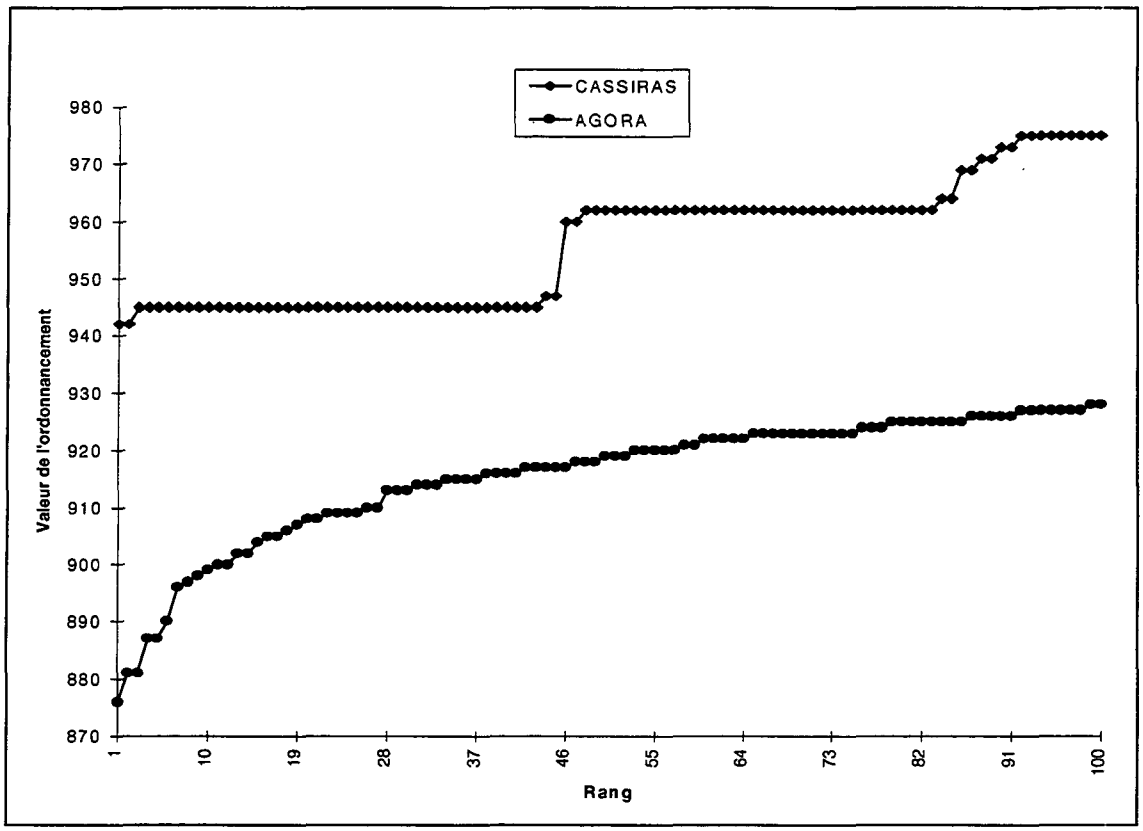


Figure 8.3 : Comparaison des algorithmes CASSIRAS et AGORA pour P_REAL.

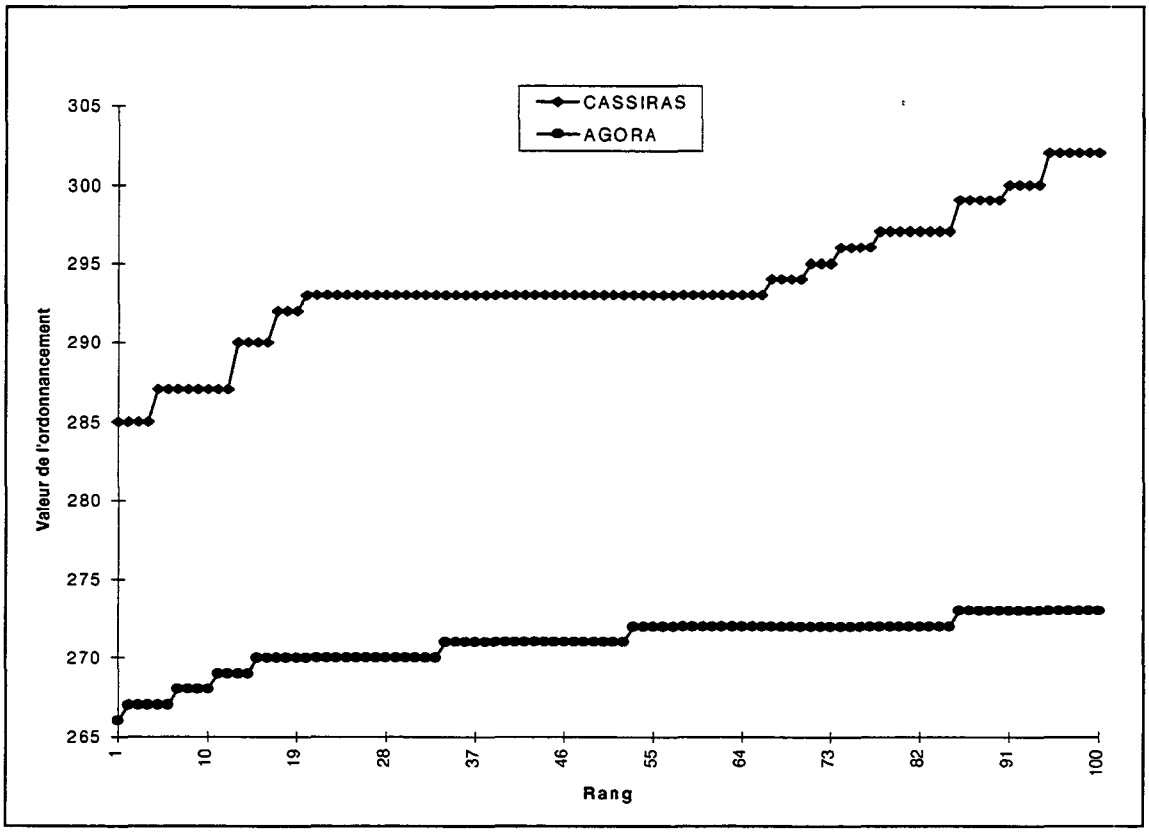


Figure 8.4 : Comparaison des algorithmes CASSIRAS et AGORA pour P_RANDOM.

Dans la Figure 8.5 nous indiquons les valeurs des différents ordonnancements fournis par AGORA pour P_REAL avec n = 20 et p_m^{max} = 0.025 lorsque le nombre μ de règles utilisées varie. Cette figure est représentative pour les différentes valeurs de n et p_m^{max} testées.

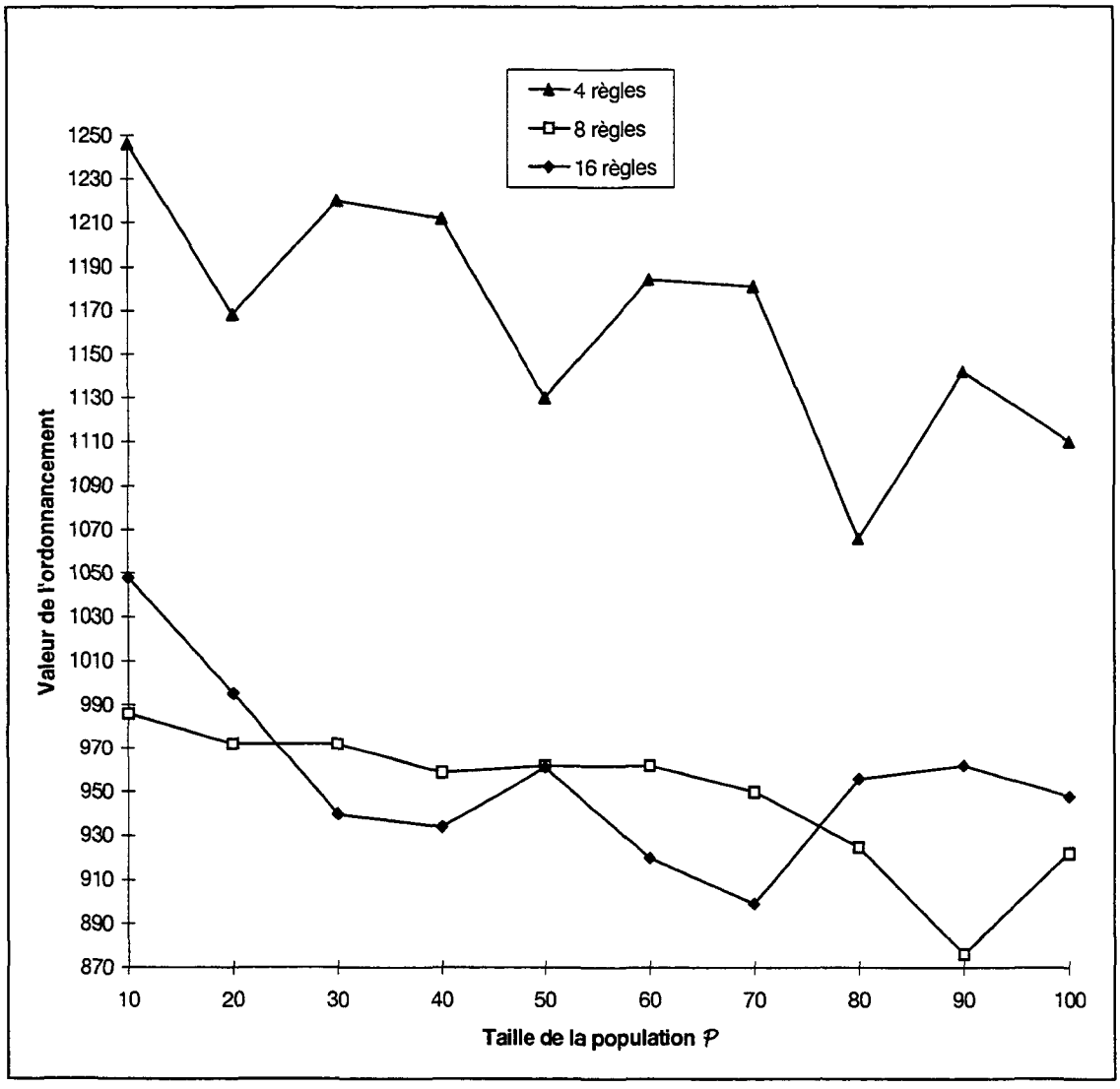


Figure 8.5 : Comparaison des valeurs des ordonnancements fournis par AGORA lorsque le nombre μ de règles utilisées varie.

Nous observons que lorsque μ est trop petit (μ = 4, i.e. que seulement le premier critère de $\mathcal{R}_1$ est employé pour déterminer la prochaine opération du robot) la qualité des résultats obtenus se détériore.

Remarquons aussi qu'une condition pour obtenir des résultats satisfaisants avec AGORA est d'utiliser une population $\mathcal{P}$ dont la taille soit suffisamment grande (supérieure à 20).

La Figure 8.6 compare les opérateurs de combinaison à un point et de combinaison uniforme avec notre opérateur de combinaison uniforme généralisé. Cette comparaison se base sur les 300 meilleurs ordonnancements fournis par AGORA avec chacun de ces opérateurs de combinaison lors de nos tests sur P_REAL. Une comparaison similaire est obtenue si l'on analyse les résultats enregistrés pour P_RANDOM.

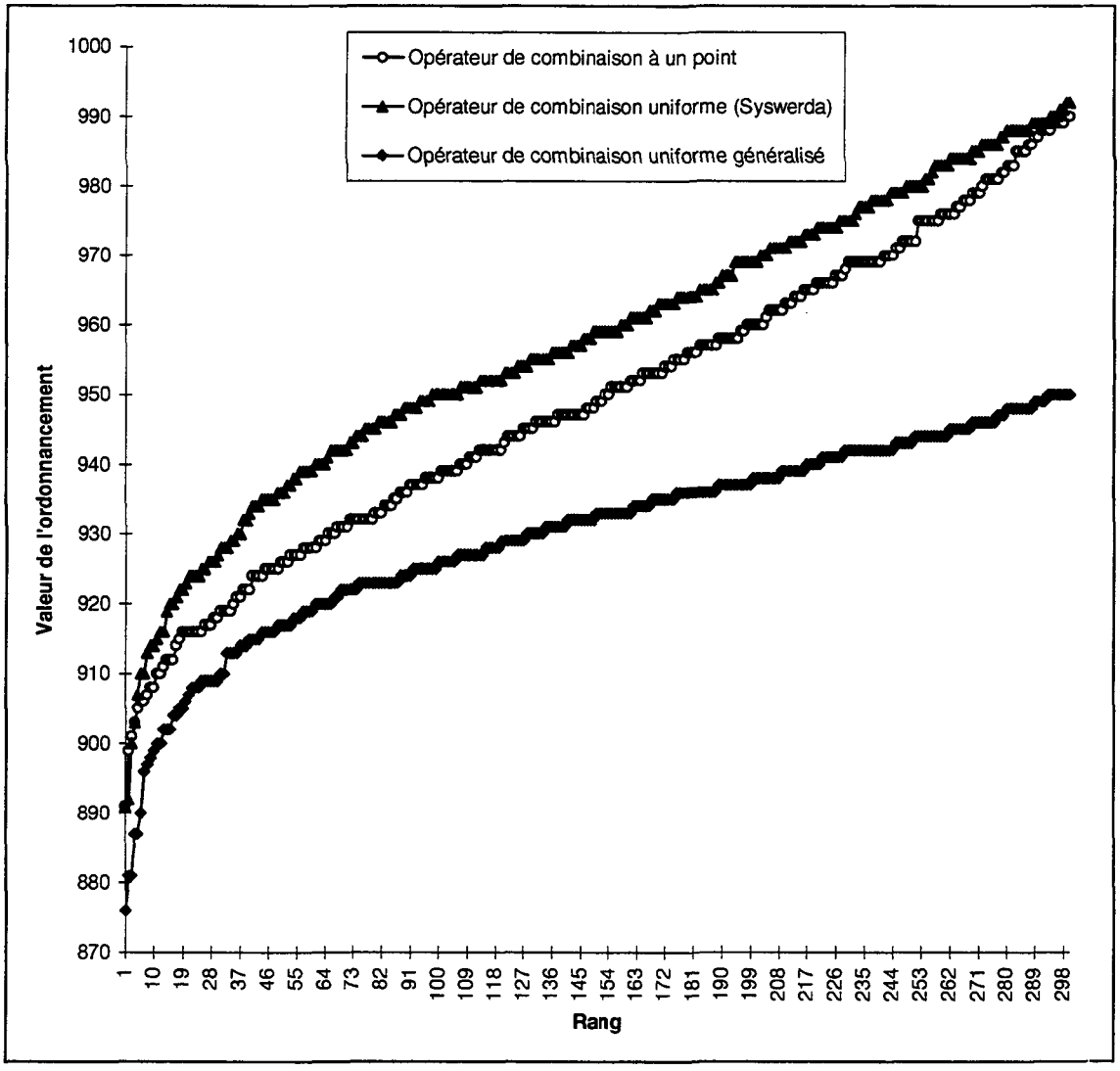


Figure 8.6 : Comparaison des opérateurs de combinaison.

Nous remarquons que l'utilisation de l'opérateur de combinaison uniforme généralisé permet d'améliorer significativement la qualité des ordonnancements obtenus à l'aide des deux autres opérateurs de combinaison. Cette amélioration s'explique principalement par le fait que nous utilisons des informations relatives aux solutions de $\mathcal{P}$ (par exemple leur valeur) lors de l'opération de combinaison des génomes alors que cette même opération est réalisée de manière purement aléatoire avec l'opérateur de combinaison à un point et l'opérateur de combinaison uniforme.

La Figure 8.7 illustre l'évolution de certaines valeurs représentatives des solutions présentes dans la population $\mathcal{P}$ au cours de l'exécution d'AGORA qui nous a permis d'obtenir le meilleur ordonnancement pour P_REAL. À chaque génération de notre algorithme évolutif, nous indiquons les valeurs maximale, moyenne et minimale des ordonnancements de $\mathcal{P}$.

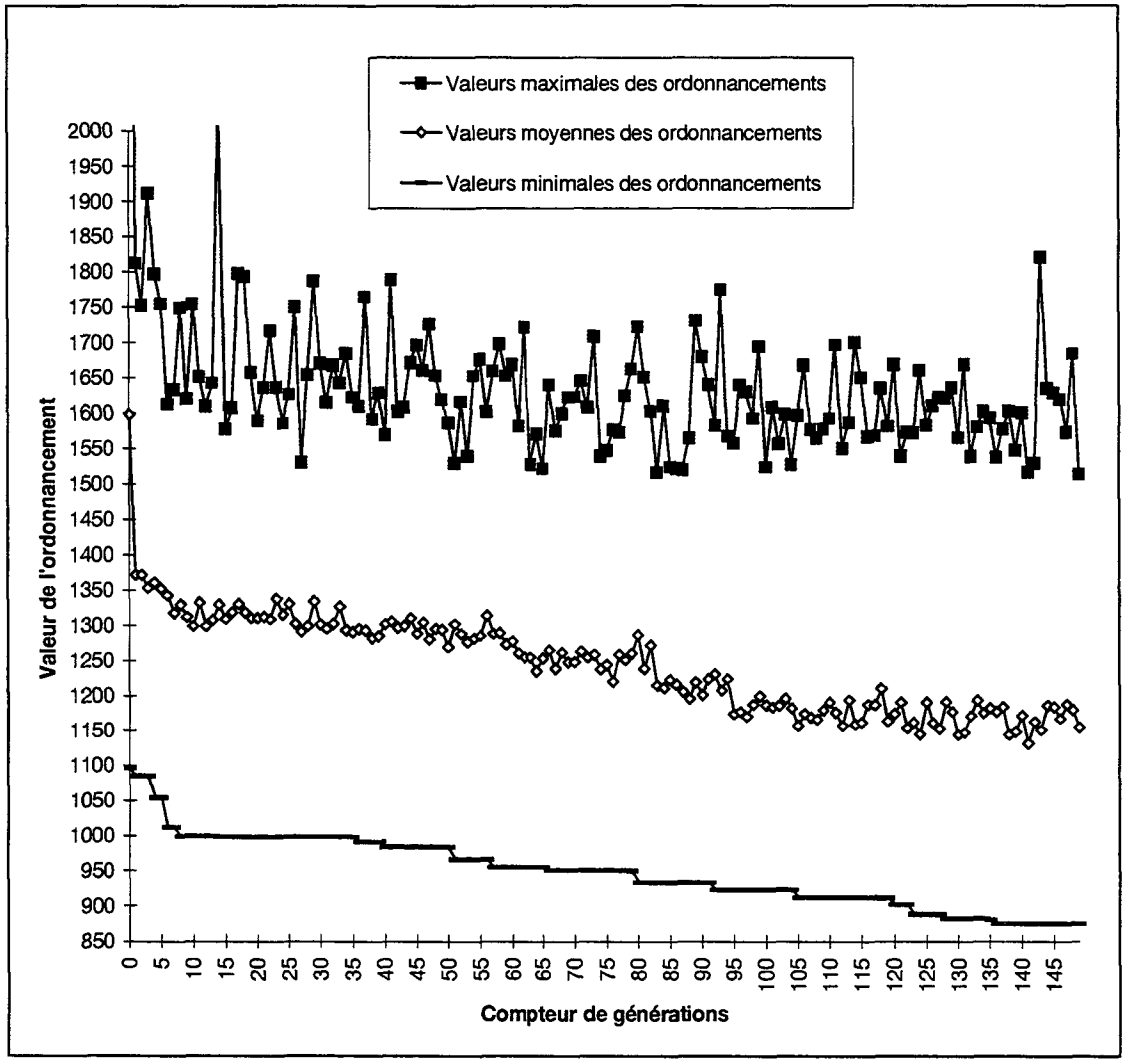


Figure 8.7 : Évolution des valeurs représentatives des solutions de $\mathcal{P}$ lors d'une exécution d'AGORA.

Nous observons que la valeur moyenne des solutions de $\mathcal{P}$ diminue au fur et à mesure que le nombre de générations augmente, ce qui est conforme à l'objectif principal d'une méthode évolutive.

Dans la Figure 8.8 nous indiquons les variations de l'entropie E de $\mathcal{P}$ et de la probabilité de mutation p_m enregistrées lors de l'exécution d'AGORA illustrée dans la Figure 8.7.

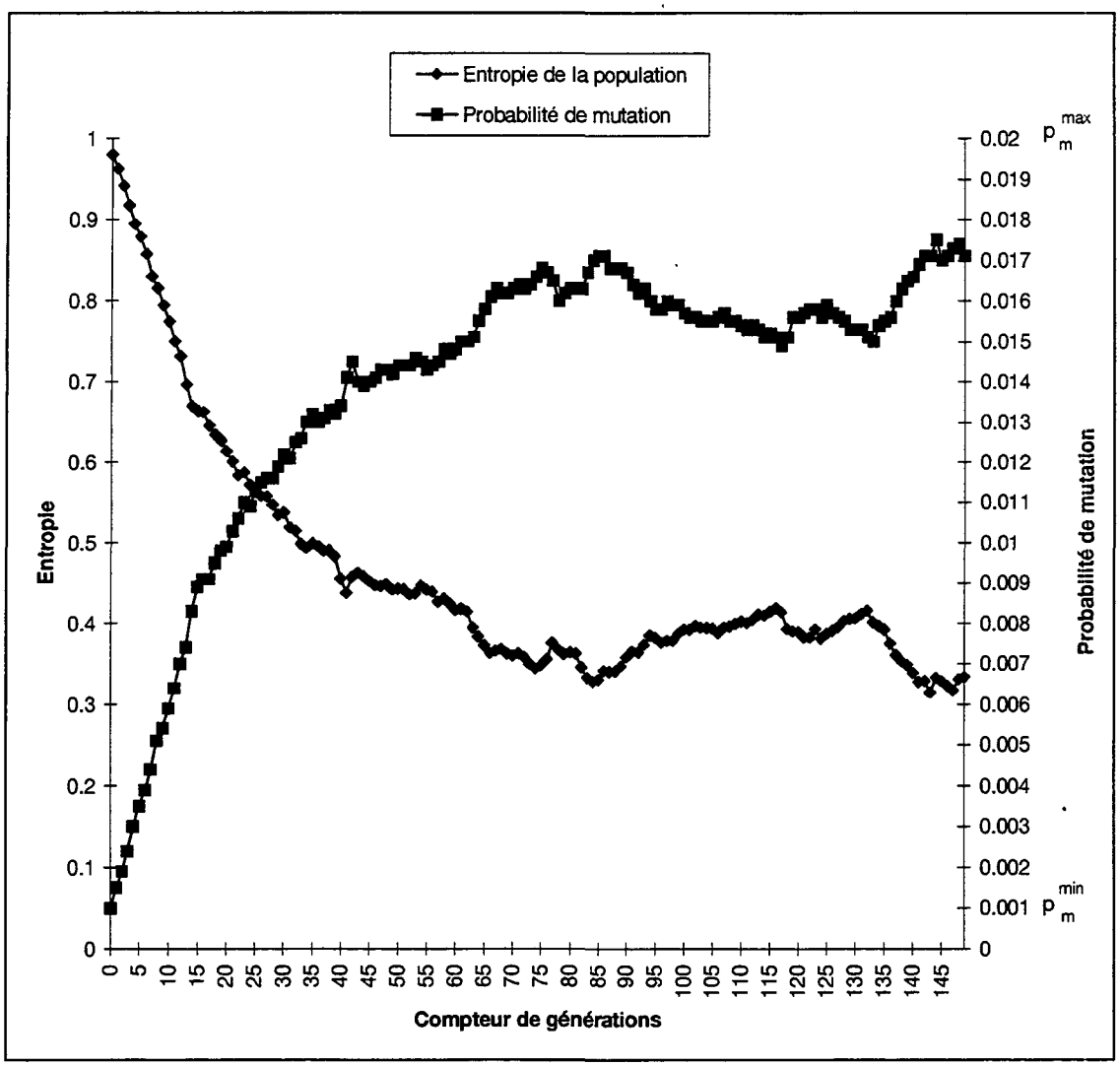


Figure 8.8 : Évolution de l'entropie et de la probabilité de mutation lors d'une exécution d'AGORA.

Nous observons que p_m varie en phase avec E. Lorsque la population $\mathcal{P}$ est très diversifiée (i.e. E est proche de 1), p_m est proche de p_m^{min}. Dès que $\mathcal{P}$ commence à converger (i.e. E se rapproche de 0), p_m augmente, jusqu'à une valeur maximale p_m^{max}, afin de maintenir une certaine diversité dans $\mathcal{P}$.

Comme notre AG joue le rôle d'un système expert dont l'objectif est de choisir les valeurs des paramètres de contrôle de CASSIRAS qui sont les plus appropriées, il est important de relever que chaque évaluation de la qualité d'une solution de $\mathcal{P}$ requiert une exécution de notre heuristique constructive. Par conséquent, le temps d'exécution de notre AG est proportionnel au produit $|\mathcal{P}| \cdot I$. Les temps de calcul nécessaires pour obtenir les meilleurs ordonnancements pour P_REAL et P_RANDOM sont respectivement de 1'739 et 258 secondes sur une station de travail Silicon Graphics (200 Mhz).

8.5 Remarques finales

Nous avons présenté une nouvelle méthode robuste dirigée par un AG pour résoudre un POTELR. Cette technique rend l'algorithme constructif CASSIRAS plus robuste car il converge plus souvent vers des ordonnancements dont la qualité est très proche des meilleures solutions que nous ayons obtenues pour deux POTELR différents. De plus, cette nouvelle approche possède les avantages suivants :

- l'utilisateur d'une telle méthode n'a plus besoin de choisir les valeurs des paramètres de contrôle de l'algorithme CASSIRAS,
- elle s'adapte dynamiquement à la structure du POTELR qu'on lui donne à résoudre.

De plus, notons que notre méthode évolutive est flexible dans le sens qu'elle peut être généralisée afin d'être utilisable avec presque n'importe quel algorithme constructif. Pour illustrer notre propos, considérons un algorithme constructif qui fournit la solution d'un problème P d'optimisation combinatoire en un nombre fini η d'étapes et où les décisions prises à chacune de ces η étapes sont basées sur un ensemble de μ règles. Dans cette situation, n'importe quelle solution de P peut être vue comme une séquence $Z = (z_1, z_2, \ldots, z_k, \ldots, z_\eta)$ de η règles de décisions locales, chacune d'entre elles étant choisie parmi μ règles distinctes. En d'autre termes, z_k indique quelle règle parmi les μ règles à disposition doit être utilisée à l'étape k de la construction d'une solution de P. Par conséquent, la recherche d'une bonne solution au problème P peut être ramenée à la détermination d'une bonne séquence de η règles de décisions locales. Pour atteindre ce but, l'algorithme génétique présenté dans ce chapitre peut être utilisé.

Bibliographie relative au POTELR

[Agn94] Agnetis A., Pacciarelli D. et Rossi P. (1994), "Batch scheduling in a two-machine cell with swapping devices", Rapport 28.94, Università di Roma (La Sapienza), Italie.

[Arm94] Armstrong R., Lei L. et Gu S. (1994), "A bounding scheme for deriving the minimal cycle time of a single-transporter N-stage process with time-window constraints", *European Journal of Operational Research* 78, 130-140.

[Asf85] Asfahl C.B. (1985), *Robots and Manufacturing Automation*, John Wiley, New York.

[Bak74] Baker K.R. (1974), *Introduction to Sequencing and Scheduling*, Addison Wesley, Reading, Mass.

[Bap92] Baptiste P., Legeard B. et Varnier C. (1992), "Hoist scheduling problem : an approach based on constraints logic programming", Proceedings of IEEE international conference on robotics and automation 2, 1139-1144.

[Bap93] Baptiste P., Manier M.A. et Varnier C. (1993), "Ordonnancement de lignes de galvanoplastie : approche programmation en logique avec contraintes", Journées thématiques de Besançon, France.

[Bar93] Barnes J.W. et Laguna M. (1993), "A tabu search experience in production scheduling", *Annals of Operations Research* 41, 141-156.

[Bel58] Bellman R.E. (1958), "On a routing problem", *Quarterly of Applied Mathematics* 16, 87-90.

[Bla87] Blazewicz J. (1987), "Selected topics in scheduling theory", *Annals of Discrete Mathematics* 31, 1-60.

[Bla91] Blazewicz J., Eiselt H.A, Finke G., Laporte G. et Weglarz J. (1991), "Scheduling tasks and vehicles in a flexible manufacturing system", *The International Journal of Flexible Manufacturing Systems* 4, 5-16.

[Bla94] Blazewicz J. et Finke G. (1994), "Scheduling with resource management in manufacturing systems", *European Journal of Operational Research* 76, 1-14.

[Car82] Carlier J. et Chrétienne P. (1982), "Les problèmes d'ordonnancement : un domaine très ouvert", *RAIRO*, 175-217.

[Car88] Carlier J. et Chrétienne P. (1988), *Les Problèmes d'Ordonnancement : Modélisation, Complexité, Algorithmes*, Masson, Paris.

[Cof76] Coffman E.G. (1976), *Computer & Jobshop Scheduling Theory*, John Wiley, New York.

[Con67] Conway R.W., Maxwell W.L. et Miller L.W. (1967), *Theory of Scheduling*, Addison Wesley, Reading, Mass.

[Cor90] Cormen T.H., Leiserson C.E. et Rivest R.L. (1990), *Introduction to Algorithms*, The MIT Press.

[Cos95] Costa D., Hertz A. et Dubuis O. (1995), "Embedding of a sequential procedure within an evolutionary algorithm for coloring problems in graphs", *Journal of Heuristics* 1, 1, 105-128.

[Cra94a] Crama Y. et van de Klundert J. (1994), "Cyclic scheduling of identical parts in a robotic cell", à paraître dans *Operations Research.*

[Cra94b] Crama Y., Oerlemans A.G. et Spieksma F.C.R. (1994), "Production planning in automated manufacturing", Lecture Notes in Economics and Mathematical Systems 414, Springer-Verlag, Berlin Heidelberg.

[Cra95] Crama Y. (1995), "Combinatorial optimization models for production scheduling in automated manufacturing systems", Proceedings of the 14th EURO conference, 237-259.

[Dav87] Davis L. (1987). *Genetic Algorithms and Simulated Annealing*, Pitman, Londres.

[Del93] Dell'Amico M. et Trubian M. (1993), "Applying tabu search to the job shop scheduling problem", *Annals of Operations Research* 41, 231-252.

[Dor95] Dorndorf U. et Pesch E. (1995), "Evolution based learning in job shop scheduling environment", *Computers and Operations Research* 22, 1, 25-40.

[Fis63] Fisher H. et Thompson G. L. (1963), "Probabilistic learning combinations of local job shop scheduling rules" dans *Industrial Scheduling*, Muth J. F. et Thompson G. L. (éditeurs), Prentice Hall, Englewood Cliffs.

[Fle94] Fleurent C. (1994), "Algorithmes génétiques hybrides pour l'optimisation combinatoire", Thèse présentée au Département d'Informatique et de Recherche Opérationnelle, Université de Montréal, Canada.

[Fog89] Fogarty T. C. (1989), "Varying the probability of mutation in the genetic algorithm", Proceedings of the third international conference on genetic algorithms, Schaffer J. D. (éditeur), Morgan Kaufman Publishers, 104-109.

[Fre82] French S. (1982), *Sequencing and Scheduling*, John Wiley, New York.

[Gar79] Garey M.R. et Johnson D.S. (1979), *Computers and Intractability : a Guide to the Theory of NP-Completeness*, Freeman, New York.

[Gil64] Gilmore P. et Gomory R. (1964), "Sequencing a one-state variable machine : a solvable case of the traveling salesman problem", *Operations Research* 12, 655-679.

[Gol89] Goldberg D.E. (1989), *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley.

[Got93] GOTHA (1993), "Les problèmes d'ordonnancement", *Recherche Opérationnelle* 27, 1, 77-150.

[Goy75] Goyal S. (1975), "Jobshop sequencing problem with no-wait in process", *Inter. J. Prod. Res.* 13, 197-206.

[Goy88] Goyal S. et Sriskandarajah C. (1988), "No-wait shop scheduling : computational complexity and approximate algorithms", *Opsearch* 25, 220-244.

[Gra79] Graham R.L., Lawler E.L., Lenstra J.K. et Rinnooy Kan A.H.G. (1979), "Optimization and approximation in deterministic sequencing and scheduling : a survey", *Annals of Discrete Mathematics* 5, 287-326.

[Hal93a] Hall N.G. et Sriskandarajah C. (1993), "Machine scheduling problems with blocking and no-wait in process", Rapport 91-05, Department of Industrial Engineering, University of Toronto, Canada.

[Hal93b] Hall N.G., Kamoun H. et Sriskandarajah C. (1993), "Scheduling in robotic cells : two machine cells and identical parts", Rapport 93-06, Department of Industrial Engineering, University of Toronto, Canada. À paraître dans *Operations Research.*

[Hal93c] Hall N.G., Kamoun H. et Sriskandarajah C. (1993), "Scheduling in robotic cells : large cells", Rapport 93-07, Department of Industrial Engineering, University of Toronto, Canada.

[Hal94] Hall N.G., Potts C.N et Sriskandarajah C. (1994), "Parallel machine scheduling with a common server", Rapport 94-21, College of Business, The Ohio State University, États-Unis.

[Han93] Hanen C. et Munier A. (1993), "Ordonnancement cyclique d'un robot sur une ligne de galvanoplastie : modèles et algorithmes", Rapport LITP 93-30, Paris.

[Han94] Hanen C. et Munier A. (1994), "Periodic scheduling of several hoists", Proceedings of fourth international conference on project management and scheduling, 108-110.

[Her96a] Hertz A. et Widmer M. (1996), "An improved tabu search approach for solving the job shop scheduling problem with tooling constraints", *Discrete Applied Mathematics* 65, 319-345.

[Her96b] Hertz A., Mottet Y. et Rochat Y. (1996), "On a scheduling problem in a robotized analytical system", *Discrete Applied Mathematics* 65, 285-318.

[Hol75] Holland J.H. (1975), *Adaptation in Natural and Artificial Systems*, Ann Arbor, University of Michigan Press.

[Ioa93] Ioachim I. et Soumis F. (1993), "Schedule efficiency in a robotic production cell, Cahier du GERAD G-93-05, École des HEC, Montréal, Canada.

[Jen93] Jeng W.D., Lin J.T. et Wen U.P. (1993), "Algorithms for sequencing robot activities in a robot-centred parallel-processor workcell", *Computers and Operations Research* 20, 185-197.

[Kam93] Kamoun H., Hall N.G. et Sriskandarajah C. (1993), "Scheduling in robotic cells : heuristics and cell design", à paraître dans *Management Science*.

[Kel61] Kelley J.E. (1961), "Critical-path planning and scheduling : mathematical basis", *Operations Research* 9, 296-320.

[Kin93] King R.E., Hodgson T.J et Chafee F.W. (1993), "Robot task scheduling in a flexible manufacturing cell", *IIE Transactions* 25, 80-87.

[Kis91a] Kise H. (1991), "On an automated two-machine flowshop scheduling problem with infinite buffer", *Journal of the Operations Research Society of Japan* 34, 354-361.

[Kis91b] Kise H., Shioyama T. et Ibaraki T. (1991), "Automated two-machine flowshop scheduling : a solvable case", *IIE Transactions* 23, 10-16.

[Klu96] van de Klundert J.J. (1996), "Scheduling problems in automated manufacturing", Thèse présentée à l'Université de Limburg, Maastricht, Pays-Bas.

[Kub89] Kubiak W. (1989), "A pseudo-polynomial algorithm for a two-machine no-wait jobshop scheduling problem", *European Journal of Operational Research* 43, 267-270.

[Laa92] Laarhoven P.J.M., Aarts E.H. et Lenstra J.K. (1992), "Job shop scheduling by simulated annealing", *Operations Research* 40, 113-125.

[Law82a] Lawler E.L. (1982), "Recent results in the theory of machine scheduling" dans *Math. Programming : the State of the Art*, Bachem A., Groetschel M. et Korte B., (éditeurs), Springer.

[Law82b] Lawler E.L., Lenstra J.K. et Rinnooy Kan A.H.G. (1982), "Recent developments in deterministic sequencing and scheduling : a survey", dans *Deterministic and Stochastic Scheduling*, Dempster M.A.H. et al., (éditeurs), Dordrect.

[Lei89] Lei L. et Wang T.J. (1989), "A proof : the cyclic hoist-scheduling problem is NP-complete", Rapport 89-16, Graduate School of Management, Rutgers University, États-Unis.

[Lei91] Lei L. et Wang T.J. (1991), "The minimum common-cycle algorithm for cyclic scheduling of two material handling hoists with time window constraints", *Management Science* 37, 1629-1639.

[Lei94] Lei L. et Wang T.J. (1994), "On the optimal cyclic schedules of single hoist electroplating processes", *IIE Transactions* 26, 25-33.

[Len77] Lenstra J.K., Rinnooy Kan A.H.G. et Brucker P. (1977), "Complexity of machine scheduling problems", *Annals of Discrete Mathematics* 1, 343-362.

[Len85] Lenstra J.K. et Rinnooy Kan A.H.G. (1985), "Scheduling theory since 1981 : an annotated bibliography", dans *Combinatorial Optimization : Annotated Bibliographies*, O'Leigeartaigh M., Lenstra JK. et Rinnooy Kan A.H.G., (éditeurs), John Wiley, Chichester.

[Lev95] Levner E., Kats V. et Levit V.E. (1995), "An improved algorithm for a cyclic robotic scheduling problem", Proceedings of the international workshop on intelligent scheduling of robots and FMS, Levner E. (éditeur), 129-141.

[Liu93] Liu S.-C. et Lin L. (1993), "Dynamic sequencing of robot moves in a manufacturing cell", *European Journal of Operational Research* 69, 482-497.

[Log92] Logendran R. et Sriskandarajah C. (1992), "Sequencing of robot activities and parts in two machine robotic cells", Rapport 91-06, Department of Industrial Engineering, University of Toronto, Canada.

[Mag80] Maggu P.L. et Dass G. (1980), "On 2xn sequencing problem with transportation time of jobs", *Pure and Applied Mathematika Science* 12, 1-6.

[Man94] Manier M.-A, Varnier C. et Baptiste P. (1994), "A multi-hoists scheduling problem approach", Proceedings of fourth international conference on project management and scheduling, 110-115.

[Mat87] Matsuo H., Shang S. et Sullivan R.S. (1987), "Crane scheduling and machine layout problem in a computer integrated manufacturing environment", Rapport technique, Graduate School of Business, University of Texas, Austin, États-Unis.

[Mat88] Matsuo H. (1988), "Cyclic sequencing problems in the two-machine permutation flow shop : complexity, worst case and average case analysis", Rapport technique, Graduate School of Business, University of Texas, Austin, États-Unis.

[Now93] Nowicki E. et Smutnicki C. (1993), "A fast taboo search algorithm for the job shop scheduling problem", Rapport 93/8, Instytut Cybernetyki Technicznej, Politechniki Wroclawskiej, Wroclaw, Pologne.

[Ogr85] O'Grady P. J. et Harrison C. (1985), "A general search sequencing rule for job shop sequencing", *Int. J. Prod. Res.* 23, 951-973.

[Pan79] Panwalkar S.S. et Woollam C.R. (1979), "Flowshop problems with no in-process waiting : a special case", *Journal of the Operational Research Society* 30, 661-664.

[Pan80] Panwalkar S.S. et Woollam C.R. (1980), "Ordered flowshop problems with no in-process waiting : further results", *Journal of the Operational Research Society* 31, 1039-1043.

[Phi76] Phillips L.W. et Unger P.S. (1976), "Mathematical programming solution of a hoist scheduling program", *AIIE Transactions* 8, 219-225.

[Raf95] Raff S. J. (éditeur) (1995), *Computers and Operations Research* 22, Special Issue on Genetic Algorithms, No. 1.

[Red72] Reddi S.S. et Ramamoorthy C.V. (1972), "On the flowshop sequencing problem with no-wait in process", *Operational Research Quarterly* 23, 323-331.

[Roc84] Rock H. (1984), "Some new results in no-wait flowshop scheduling", *Zeitschrift fuer Operations Research* 28, 1-16.

[Roc95] Rochat Y. (1995), "A genetic approach for solving a scheduling problem in a robotized analytical system", Proceedings of the international workshop on intelligent scheduling of robots and FMS, Levner E. (éditeur), 191-203.

[Sah79] Sahni S. et Cho Y. (1979), "Complexity of scheduling shops with no-wait in process", *Mathematics of Operations Research* 4, 448-457.

[Set92] Sethi S.P., Sriskandarajah C., Sorger G., Blazewicz J. et Kubiak W. (1992), "Sequencing of parts and robot moves in a robotic cell", *The International Journal of Flexible Manufacturing Systems* 4, 331-358.

[Sha88] Shapiro G.W et Nuttle H.L.W. (1988), "Hoist scheduling for a PCB electroplating facility", *IIE Transactions* 20, 157-167.

[Ste85] Stecke K.E (1985), "Design, planning, scheduling and control problems of flexible manufacturing systems", *Annals of Operations Research* 3, 3-12.

[Sys89] Syswerda G. (1989), "Uniform crossover in genetic algorithms", Proceedings of the third international conference on genetic algorithms, Schaffer J.D. (éditeur), Morgan Kaufman Publishers, 2-9.

[Wid89] Widmer M. et Hertz A. (1989), "A new heuristic method for the flow shop sequencing problem", *European Journal of Operational Research* 41, 186-193.

CONCLUSION

Les méthodes développées dans le cadre de ce travail pour résoudre des problèmes d'élaboration de tournées de véhicules sous contraintes de fenêtres de temps sont performantes. L'originalité de notre méthode probabiliste de diversification et d'intensification pour les heuristiques de recherche locale du PTV a notamment permis d'améliorer significativement un grand nombre des meilleurs résultats connus à ce jour pour des problèmes-tests représentatifs de ce type de problèmes. De plus, grâce à une adaptation judicieuse de la méta-heuristique Tabou (où un processus d'intensification spécifique est employé et certaines contraintes du problème sont relâchées) nous avons pu résoudre un problème réel de distribution auquel l'un des principaux producteurs suisses de farine et de nourriture pour bétail doit faire face.

Dans le domaine de l'ordonnancement, nous proposons un modèle mathématique et des algorithmes pour un nouveau type problèmes qui peut être considéré comme une généralisation des problèmes de flowshop rencontrés dans les ateliers automatisés de traitement et /ou de production. Bien que basées sur une heuristique constructive simple, nos méthodes sont capables de gérer et planifier efficacement les déplacements d'un robot dans un laboratoire chimique de traitements d'échantillons. De plus, elles ont été appliquées avec succès en milieu industriel pour la préparation automatisée d'échantillons et de vaccins.

Les approches évolutives qui ont été élaborées dans le cadre de cette thèse font partie d'un domaine situé entre la biologie, l'informatique et les mathématiques. Cette nouvelle discipline, appelée *la vie artificielle*, est en plein essor et constitue un formidable outil d'investigation scientifique. Pour deux problèmes différents d'optimisation combinatoire, nous avons montré qu'il était possible d'obtenir des solutions d'excellente qualité si l'on exploite intelligemment les propriétés et les caractéristiques d'un ensemble de solutions distinctes qui, lorsqu'elles sont considérées individuellement, ne sont pas forcément très bonnes.

CURRICULUM VITAE

Nom, prénom :	ROCHAT Yves
Date de naissance :	9 juillet 1966
Lieu d'origine :	L'Abbaye (Vaud)
État civil :	Célibataire
Adresse :	Chemin de la Clergère 36, CH-1009 Pully

Formation

1982	Diplôme d'études secondaires au collège de Moudon
1985	Certificat de maturité de type C du gymnase du Bugnon à Lausanne
1991	Diplôme d'ingénieur mathématicien de l'École Polytechnique Fédérale de Lausanne (option aide à la décision)
1993	Certificat du 1er cours postgrade en Recherche Opérationnelle organisé dans le cadre de la Convention Transfrontalière Universitaire Rhône-Alpes par l'École Polytechnique Fédérale de Lausanne, l'Institut National Polytechnique et l'Université Joseph Fourier de Grenoble

Activités professionnelles

Dès 1991 :	Assistant en Recherche Opérationnelle de l'École Polytechnique Fédérale de Lausanne (Prof. A. Hertz et D. de Werra, Département de Mathématiques)
Dès 1994 :	Chargé de cours du Département de Systèmes de Communications de l'École Polytechnique Fédérale de Lausanne (cours de 2e cycle) : *Compléments de Recherche Opérationnelle.*