

Mémoire de recherche préparé dans le cadre du DEA ASIH

Optimisation combinatoire et problèmes de capacité d'infrastructure ferroviaire

Xavier DELORME

Promoteurs : Xavier GANDIBLEUX (UVHC-LAMIH)

Joaquin RODRIGUEZ (INRETS-ESTAS)

Équipes d'accueil : LAMIH-ROI (Arnaud FRÉVILLE)

INRETS-ESTAS (Gérard COUVREUR)

Année académique 1999-2000

LAMIH, UMR CNRS 8530 - Université de Valenciennes
Le Mont Houy - 59313 VALENCIENNES CEDEX 9

Table des matières

Remerciements	3
Introduction	4
1 Présentation des problèmes d'optimisation combinatoire	6
1.1 Modèle général	6
1.1.1 Programmation linéaire	6
1.1.2 Problèmes en variables entières	7
1.1.3 Problèmes en variables binaires	7
1.2 Méthodes de résolution	8
1.2.1 Méthodes exactes	8
1.2.2 Méthodes approchées	8
1.3 Quelques problèmes classiques de l'optimisation combinatoire	9
1.3.1 Set Covering Problem	9
1.3.2 Set Packing Problem	10
1.3.3 Node Packing Problem	10
2 Résolution du Set Covering Problem	13
2.1 Résolution exacte : algorithme de Branch & Bound	13
2.1.1 Architecture du Branch & Bound	13
2.1.2 Règles de réductions	13
2.1.3 Principes de parcours de l'arbre	15
2.1.4 Évaluation d'un nœud	16
2.2 Résolution approchée : la métaheuristique GRASP	16
2.2.1 Architecture de GRASP pour le SCP	17
2.2.2 Obtention d'une solution initiale	17
2.2.3 Recherche locale	18
2.2.4 Intensification de la recherche locale	18
3 Expérimentation : comportement de GRASP sur le SCP	19
3.1 Algorithmes testés et critères d'évaluation	19
3.2 Matériel utilisé	20
3.3 Instances utilisées	20
3.4 Résultats obtenus	21
3.4.1 Comparaison des différentes versions de GRASP	21

3.4.2	Comparaison de GRASP avec des algorithmes gloutons	23
3.4.3	Comparaison de GRASP avec un algorithme exact	24
3.4.4	Comparaison de GRASP avec les meilleurs résultats connus	25
3.5	Conclusion de l'étude du comportement de GRASP sur le SCP	25
4	Le problème de capacité ferroviaire	27
4.1	Position du problème	27
4.2	Difficulté du problème	28
4.3	Les travaux des «Nederlandse Spoorwegen»	29
4.3.1	Description générale du modèle	30
4.3.2	Extensions du modèle	31
4.3.3	Techniques de résolution	32
4.4	Adaptation de GRASP au Set Packing Problem	32
5	Expérimentation : le nœud ferroviaire de Gonesse	34
5.1	Présentation de la situation	34
5.2	Présentation de l'expérimentation	35
5.2.1	Modèle utilisé	35
5.2.2	Techniques de résolution employées	36
5.2.3	Cas de figure traités	37
5.3	Résultats observés avec GRASP	39
	Conclusion et perspectives	41
	Bibliographie	43
	Annexes	45

Remerciements

Je tiens tout d'abord à remercier Monsieur Patrick Millot, Professeur à l'Université de Valenciennes et Directeur adjoint du LAMIH¹, et Monsieur Gérard Couvreur, Directeur de l'Unité de Recherche ESTAS² (INRETS³), de m'avoir accueilli dans leur laboratoire.

J'exprimerai aussi ma gratitude envers Monsieur Xavier Gandibleux, Maître de Conférences à l'Université de Valenciennes, et Monsieur Joaquin Rodriguez, Chargé de Recherche à l'INRETS, qui m'ont encadré durant ce travail pour leur aide et leurs conseils.

Enfin, je finirai en remerciant l'ensemble des personnes travaillant au LAMIH et à l'INRETS que j'ai rencontré cette année ainsi que les étudiants du DEA ASIH⁴ pour leur aide et leur bonne humeur quotidienne.

¹Laboratoire d'Automatique et de Mécanique Industrielles et Humaines

²Évaluation des Systèmes de Transports Automatisés et de leur Sécurité

³Institut National de Recherche sur les Transports et leur Sécurité

⁴Automatique des Systèmes Industriels et Humains

Introduction

Ce mémoire présente le travail de recherche que j'ai réalisé dans le cadre de mon stage de DEA Automatisation des Systèmes Industriels et Humains (ASIH) à l'Université de Valenciennes et du Hainaut-Cambrésis en collaboration avec l'Institut National de Recherche sur les Transports et leur Sécurité (INRETS) de Villeneuve-d'Ascq.

Ce travail aborde la problématique de la capacité d'une infrastructure ferroviaire. Cette problématique trouve tout son intérêt dans le domaine de la planification, notamment dans le contexte actuel de diminution des ressources financières et d'accroissement prévisible du trafic durant les prochaines années.

La notion de capacité d'une infrastructure ferroviaire intervient dans plusieurs questions telle que l'étude de la faisabilité du passage des trains prévus dans cette infrastructure, la fluidification de ce passage ou la programmation des travaux, mais aussi dans le cadre de la tarification des sillons ou pour évaluer l'intérêt de modifications de l'infrastructure. Nous avons travaillé durant cette année sur le problème de la faisabilité du passage des trains dans un nœud par rapport à un horaire établi. La modélisation retenue pour cette étude s'inspire de travaux antérieurs publiés et relève du domaine de l'optimisation combinatoire.

Après un court rappel de problèmes classiques rencontrés en optimisation combinatoire (Set Covering, Node Packing et Set Packing Problems) et des méthodes existantes permettant de les résoudre, nous nous intéressons plus en détail à la résolution du Set Covering Problem. En effet, celui-ci a fait l'objet d'une littérature importante et plusieurs méthodes de résolution ont été évaluées sur des instances numériques de référence. Nous étudions ainsi le comportement et les performances d'une heuristique de type glouton, d'une métaheuristique stochastique (GRASP) et d'une méthode exacte de Branch & Bound.

Les résultats relevés lors de nos expérimentations sont ensuite présentés et commentés dans l'esprit d'une analyse comparative. De plus une confrontation avec les meilleurs résultats connus dans la littérature est effectuée. Au vu de cette campagne expérimentale, il apparaît que la métaheuristique GRASP fournit des résultats de bonne qualité tout en conservant des temps de résolution acceptables sur les instances de grande taille. Nous retenons donc pour la suite du travail la configuration et le réglage des paramètres de GRASP qui ont conduit aux meilleurs performances de la méthode approchée.

L'intérêt de GRASP étant validé, nous pouvons nous intéresser au problème de capacité ferroviaire. Après une brève présentation du problème, nous étudions les travaux réalisés par les chemins de fer néerlandais dans le cadre d'un projet visant à aider la planification d'une

grille horaire sur l'ensemble du réseau des Pays-Bas, et plus spécialement la partie permettant de gérer le problème du routage des trains dans une gare. Le modèle qu'ils proposent est un Node Packing.

Pour finir, nous présentons une adaptation de ce modèle permettant de l'appliquer à une situation réelle française ainsi que sa résolution à l'aide de la métaheuristique GRASP. Dans ce but, nous avons retenu une modélisation du problème sous la forme d'un Set Packing qui se révèle plus adaptée dans le cadre d'une résolution approchée. De même, nous présentons une adaptation de GRASP à ce type de problème. Une première validation de ce travail a été obtenue sur base d'une expérimentation réalisée sur le nœud ferroviaire de Pierrefitte-Gonesse en considérant des situations de trafic réelles et fictives. Nous pouvons ainsi retenir les premiers résultats jugés pertinents vis-à-vis du vécu et les possibilités d'évolutions rapides de l'outil actuel vers un outil d'analyse de la saturation.

Notons que certains choix opérés lors du déroulement du travail proviennent du fait que nous ne disposions pas d'un code de calcul commercial. à défaut de ceci, nous avons travaillé avec des codes du domaine public qui, bien que performants, n'offrent pas tout le confort et les fonctionnalités attendues pour ce type d'étude.

Chapitre 1

Présentation des problèmes d'optimisation combinatoire

1.1 Modèle général

1.1.1 Programmation linéaire

Un problème d'optimisation exprimé sous la forme d'un programme linéaire en variables continues (Linear Program) [dW90, Teghem96] peut s'écrire sous la forme générale suivante (1.1) :

$$\left[\begin{array}{l} \text{'Opt'} z = \sum_{j=1}^n C_j X_j \\ \sum_{j=1}^n T_{ij} X_j \Delta_i d_i \quad , \forall i \\ l_j \leq X_j \leq u_j \quad , \forall j \\ \Delta_i \in \{ \leq ; = ; \geq \} \quad , \forall i \end{array} \right] \quad (1.1)$$

avec 'Opt' pouvant signifier Minimiser ou Maximiser selon le problème traité.

Notons que les contraintes peuvent aussi être exprimées sous forme matricielle. Dans ce cas $\sum_{j=1}^n T_{ij} X_j \Delta_i d_i, \forall i$ devient $TX \Delta d$ où T représente la matrice des contraintes. Dans la suite, les deux notations seront utilisées indifféremment.

Les variables utilisées dans ce problème prennent des valeurs réelles (quantité de produit ou temps par exemple). Le système de contraintes définissant un domaine éventuellement vide de solutions admissibles, l'objectif est d'obtenir la ou les solution(s) optimale(s) (quand il en existe) vis à vis de la fonction objectif. Des résultats d'algèbre linéaire ont montré que le domaine des solutions admissibles est convexe (sauf s'il est vide) et que la solution optimale est un sommet du domaine (ou une face s'il y a plusieurs solutions optimales).

1.1.2 Problèmes en variables entières

Cependant, dans de nombreuses situations réelles, des variables peuvent être astreintes au domaine des valeurs discrètes (nombre de pièces à produire par exemple). Ces problèmes sont dits en variables mixtes (Mixed Integer Linear Problem). Lorsque toutes les variables du problème sont entières, on parle d'un problème en variables entières (Integer Linear Program) [Teghem96] qui se présente sous la forme suivante (1.2) :

$$\left[\begin{array}{l} \text{'Opt'} \ z = \sum_{j=1}^n C_j X_j \\ \sum_{j=1}^n T_{ij} X_j \Delta_i d_i \quad , \forall i \\ l_j \leq X_j \leq u_j \quad , \forall j \\ X_j \text{ entier} \quad , \forall j \\ \Delta_i \in \{ \leq ; = ; \geq \} \quad , \forall i \end{array} \right] \quad (1.2)$$

Contrairement aux problèmes en variables continues, la résolution de ces problèmes n'est généralement pas aisée pour deux raisons :

- ce sont des problèmes NP difficiles
- la propriété de convexité du domaine des solutions n'est plus valable.

1.1.3 Problèmes en variables binaires

Enfin, certains problèmes comportent uniquement des variables de type booléen (pour des décisions par exemple). La formulation algébrique de ces problèmes appelés 0-1ILP est la suivante (1.3) :

$$\left[\begin{array}{l} \text{'Opt'} \ z = \sum_{j=1}^n C_j X_j \\ \sum_{j=1}^n T_{ij} X_j \Delta_i d_i \quad , \forall i \\ X_j = (0 ; 1) \quad , \forall j \\ \Delta_i \in \{ \leq ; = ; \geq \} \quad , \forall i \end{array} \right] \quad (1.3)$$

Même si une variable binaire peut-être considérée comme une variable aux valeurs restreintes à l'intervalle discret , les problèmes n'utilisant que des variables booléennes sont nombreux (par exemple des problèmes d'affectation, de transport, de transbordement) et font partie d'un domaine particulier appelé optimisation combinatoire [Sakarovitch84].

En plus des difficultés liées aux problèmes en variables entières ou mixtes, la principale difficulté de ces problèmes réside dans le grand nombre de variables nécessaires pour modéliser

des situations réelles. Cependant certains problèmes présentent des structures particulières qui permettent de simplifier leur résolution.

1.2 Méthodes de résolution

1.2.1 Méthodes exactes

Les méthodes de résolution exactes sont nombreuses et se caractérisent par le fait qu'elles permettent d'obtenir une ou plusieurs solutions dont l'optimalité est garantie.

Parmi ces méthodes, on peut remarquer l'algorithme du simplexe qui permet d'obtenir la solution optimale d'un problème en parcourant la fermeture convexe de l'ensemble de recherche (ensemble des solutions admissibles) et ce en passant de sommet en sommet. Malgré une complexité mathématique dans le pire des cas non polynomiale, il permet de résoudre la plupart des problèmes rapidement. Cependant il ne peut s'appliquer qu'aux problèmes ayant la propriété de convexité c'est-à-dire aux problèmes en variables continues ou à des problèmes en variables entières ayant une matrice des contraintes T unimodulaire (car dans ce cas, tous les sommets de l'ensemble de recherche sont entiers) comme les problèmes de transport ou d'affectation.

Pour les autres problèmes (ILP, MILP, 0-1ILP), il existe plusieurs méthodes :

- la programmation dynamique consistant à placer le problème dans une famille de problèmes de même nature mais de difficulté différente puis à trouver une relation de récurrence liant les solutions optimales de ces problèmes.
- le Branch & Bound consistant à faire une énumération implicite en séparant le problème en sous-problèmes et en évaluant ceux-ci à l'aide d'une relaxation (continue ou lagrangienne principalement) jusqu'à ne plus avoir que des problèmes faciles à résoudre ou dont on sait avec certitude qu'ils ne peuvent pas contenir de solution optimale.
- les méthodes polyédrales consistant à ajouter progressivement des contraintes supplémentaires afin de ramener le domaine des solutions admissibles à un domaine convexe (sans en enlever la ou les solutions optimales bien évidemment).

Ces méthodes sont générales et demande souvent une particularisation vis-à-vis d'un problème spécifique. Il existe aussi des applications génériques (AMPL, CPLEX, LINDO, MPL, OMP, XPRESS...) permettant de résoudre l'ensemble des problèmes pouvant s'écrire sous la forme algébrique d'un problème en variables binaires, entières ou mixtes.

Il faut aussi noter que la méthode consistant à effectuer une énumération explicite de toutes les solutions (c'est-à-dire de les tester une à une, méthode envisageable pour tous les problèmes à variables à valeurs bornées) montre très vite ses limites dès que le nombre de variables augmente puisque sa complexité est en k^n où k représente le nombre de valeurs que peut prendre une variable et n le nombre de variables du problème.

1.2.2 Méthodes approchées

Dans certaines situations, il est nécessaire de disposer d'une solution de bonne qualité (c'est-à-dire assez proche de l'optimale) dans un contexte de ressources (temps de calcul et/ou mémoire) limitées. Dans ce cas l'optimalité de la solution ne sera pas garantie, ni

même l'écart avec la valeur optimale. Cependant, le temps nécessaire pour obtenir cette solution sera beaucoup plus faible et pourra même être fixé (bien évidemment dans ce cas la qualité de la solution obtenue dépendra fortement du temps laissé à l'algorithme pour l'obtenir).

Typiquement ce type de méthodes, dites heuristiques¹ est particulièrement utile pour les problèmes nécessitant une solution en temps réel (ou très court) ou pour résoudre des problèmes difficiles sur des instances numériques de grande taille. Elles peuvent aussi être utilisées afin d'initialiser une méthode exacte (Branch & Bound par exemple).

Parmi ces méthodes, il faut distinguer les heuristiques ciblées sur un problème particulier et les métaheuristiques plus puissantes et adaptables pour résoudre un grand nombre de problèmes. Cependant une métaheuristique, pour être suffisamment performante sur un problème donné nécessitera une adaptation plus ou moins fine.

Ces méthodes approchées peuvent se classer en différentes catégories [Freville00] :

- Constructives (algorithmes glouton, méthode Pilote, GRASP)
- Recherche locale (algorithmes de descente, multi-départs, recuit simulé, algorithme à seuil, recherche Tabou, méthode de bruitage)
- Évolutionnistes (algorithmes génétiques, algorithmes d'évolution, recherche dispersée, méthode des chemins, systèmes de fourmis)
- Réseaux de neurones (Modèle de Hopfield-Tank, machine de Boltzmann, réseau auto-adaptatif, réseau élastique)
- Heuristiques Bayésiennes (optimisation globale, optimisation discrète)
- Superposition (perturbation des données, perturbation des paramètres d'une heuristique).

1.3 Quelques problèmes classiques de l'optimisation combinatoire

1.3.1 Set Covering Problem

Ce problème est aussi appelé problème de couverture [Teghem96, Sakarovitch84] ou encore problème de couverture d'ensemble pondéré (Weighted Set Covering Problem). En effet, si l'on suppose un ensemble $I = \{1, \dots, m\}$ et un ensemble de parties de I $P = \{P_1, \dots, P_n\}$ où $P_j \subseteq I$ pour $j \in J = \{1, \dots, n\}$, un sous-ensemble $J^* \subseteq J$ définit une couverture de I si et seulement si $\bigcup_{j \in J^*} P_j = I$. Un coût positif étant associé à chaque $j \in J$, l'objectif de ce problème est de déterminer une couverture de coût minimum. Un exemple d'application classique est celui de l'ouverture d'un nombre minimum de magasins dans une région pour couvrir l'ensemble de la zone.

Sa formulation algébrique (1.4) peut être obtenue en considérant des variables binaires qui définissent les parties retenues dans la couverture (la matrice des contraintes indique alors les différents éléments de couverts par chacune des variables) :

¹du grec heuriskein = trouver

$$\left[\begin{array}{l} \text{Min } z = \sum_{j=1}^n C_j X_j \\ \sum_{j=1}^n T_{ij} X_j \geq 1 \quad , \forall i \\ X_j = (0; 1) \quad , \forall j \end{array} \right] \quad (1.4)$$

Dans ce problème, ainsi que dans ceux que nous traiterons par la suite, les valeurs des termes de la matrice sont donc restreints à l'intervalle discret $[0; 1]$.

1.3.2 Set Packing Problem

Ce problème est aussi appelé problème d'emballage [Teghem96] ou encore couplage généralisé [Sakarovitch84]. En effet, l'objectif de ce problème est de déterminer un couplage (ensemble d'éléments indépendants) de valeur maximale. Un exemple d'application classique est celui de la mise en œuvre d'une production de valeur maximale dans un contexte de ressources limitées. Sa formulation algébrique est la suivante (1.5) :

$$\left[\begin{array}{l} \text{Max } z = \sum_{j=1}^n C_j X_j \\ \sum_{j=1}^n T_{ij} X_j \leq 1 \quad , \forall i \\ X_j = (0; 1) \quad , \forall j \end{array} \right] \quad (1.5)$$

1.3.3 Node Packing Problem

Ce problème est connu sous différents noms. On peut citer parmi eux les termes de Vertex Packing, Maximum Stable Set, Maximum Clique, Anti-Covering Problem et surtout Maximum Independent Set [Gondran et al.95, Murray et al.97].

L'objectif de ce problème est de déterminer un nombre maximum de sommets non adjacents dans un graphe $G = (V, E)$ où V représente l'ensemble des sommets et E l'ensemble des arcs. Par définition, deux sommets sont adjacents si et seulement si ils sont reliés par un arc. Sa formulation algébrique est la suivante (1.6) :

$$\left[\begin{array}{l} \text{Max } z = \sum_{j=1}^n C_j X_j \\ X_j + X_k \leq 1 \quad , \forall (j, k) \in E \\ X_j = (0; 1) \quad , \forall j \end{array} \right] \quad (1.6)$$

La formulation de ce problème est très proche de celle du Set Packing. D'ailleurs, il peut être exprimé comme tel. Dans ce cas la matrice des contraintes T possède certaines

particularités. Ainsi, la formulation algébrique de ce problème peut aussi être la suivante (1.7) :

$$\left[\begin{array}{l} \text{Max } z = \sum_{j=1}^n C_j X_j \\ \sum_{j=1}^n T_{ij} X_j \leq 1 \quad , \forall i \\ X_j = (0; 1) \quad , \forall j \\ \text{avec } \sum_{j=1}^n T_{ij} = 2 \quad , \forall i \end{array} \right] \quad (1.7)$$

Malgré tout, il ne faut pas confondre ces deux problèmes. En effet, la nuance peut sembler minime mais la propriété particulière dont hérite la matrice T est très importante pour la raison suivante :

Posons le changement de variable $X'_j = 1 - X_j, \forall j$. Dans ce cas, le problème peut se formuler de la manière suivante (1.8) :

$$\left[\begin{array}{l} \text{Max } z = \sum_{j=1}^n C_j (1 - X'_j) \\ \sum_{j=1}^n T_{ij} (1 - X'_j) \leq 1 \quad , \forall i \\ X'_j = (0; 1) \quad , \forall j \end{array} \right] \quad (1.8)$$

qui est de manière évidente équivalent au problème suivant (1.9) :

$$\left[\begin{array}{l} \text{Max } z = \sum_{j=1}^n -C_j X'_j \\ \sum_{j=1}^n T_{ij} - \sum_{j=1}^n T_{ij} X'_j \leq 1 \quad , \forall i \\ X'_j = (0; 1) \quad , \forall j \end{array} \right] \quad (1.9)$$

Et comme $\sum_{j=1}^n T_{ij} = 2$, ce problème peut aussi se formuler de la manière suivante (1.10) :

$$\left[\begin{array}{l} \text{Min } z = \sum_{j=1}^n C_j X'_j \\ \sum_{j=1}^n T_{ij} X'_j \geq 1 \quad , \forall i \\ X'_j = (0; 1) \quad , \forall j \end{array} \right] \quad (1.10)$$

C'est-à-dire comme un Set Covering Problem. Ce problème est donc en fait un cas particulier des Set Packing et Set Covering Problem.

Chapitre 2

Résolution du Set Covering Problem

Après avoir étudié plusieurs problèmes caractéristiques de l'optimisation combinatoire ainsi que les méthodes permettant de les résoudre, nous allons nous intéresser plus en détail à différentes méthodes de résolution d'un de ces problèmes ayant fait l'objet d'une littérature importante : le Set Covering Problem. Ainsi, nous nous focaliserons sur deux méthodes, l'une exacte et l'autre approchée, proposées par [Garfinkel et al.72] et par [Feo et al.94a].

2.1 Résolution exacte : algorithme de Branch & Bound

La méthode exacte que nous avons choisie pour résoudre le Set Covering Problem est un algorithme de Branch & Bound proposé par [Garfinkel et al.72]. Celui-ci reprend l'architecture générale d'un Branch & Bound mais en utilisant des règles d'évaluation et de parcours de l'arbre adaptées au Set Covering Problem. De plus, des règles de réductions permettent de diminuer la taille du problème initial. L'architecture de cet algorithme ainsi qu'un exemple didactique de son fonctionnement sont présentés en annexe I et II. Dans la suite, nous utiliserons les notations suivantes :

- τ_i correspond à la ligne $i \in I$ de la matrice T
- t_j correspond à la colonne $j \in J$ de la matrice T .

2.1.1 Architecture du Branch & Bound

Le principe général de l'algorithme (voir Figure 2.1) est celui d'un Branch & Bound classique auquel ont été ajoutés des tests de réductions.

2.1.2 Règles de réductions

Les règles utilisées pour réduire la dimension du problème sont au nombre de trois. Elles permettent de fixer certaines variables dont la valeur optimale est évidente et d'éliminer les contraintes redondantes.

La première règle (2.1) sert à fixer à 1 les variables nécessaires à l'admissibilité de la solution. Celles-ci sont détectées en recherchant des contraintes (lignes de la matrice) égales à un vecteur unité. Dans ce cas le coût de la variable est ajouté au coût fixe et elle est sortie

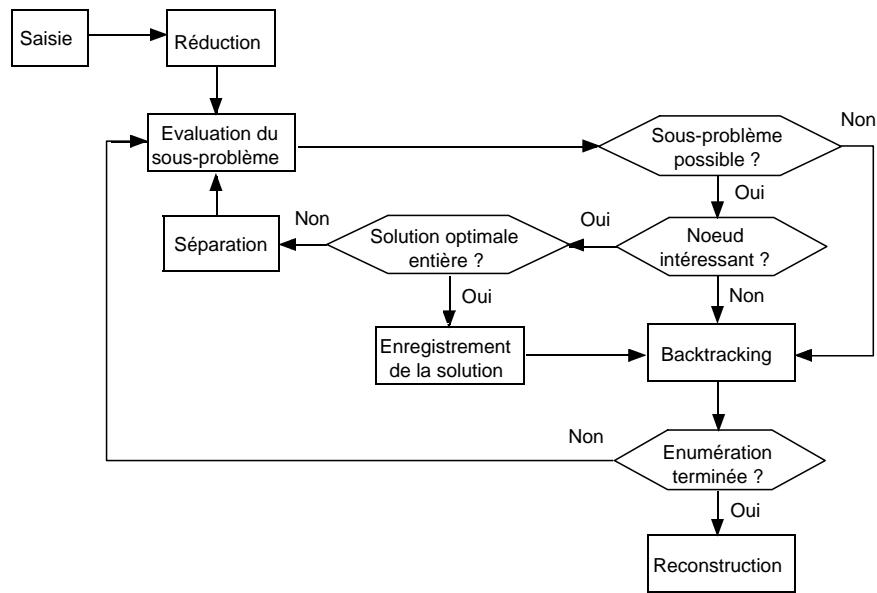


FIG. 2.1 – Architecture du Branch & Bound pour le SCP

du problème. De plus, l'ensemble des contraintes satisfaites par cette variable sont aussi supprimées.

Si τ_i est le vecteur unité e_k

$$\begin{cases} \Rightarrow X_k = 1 \text{ et } t_k \text{ est enlevé ; } \tau_i \text{ est enlevé} \\ \Rightarrow \text{si } T_{lk} = 1, \tau_l \text{ est enlevé} \end{cases} \quad (2.1)$$

La deuxième règle (2.2) permet d'éliminer les contraintes redondantes. Pour cela, il faut les comparer une à une afin de détecter si l'une d'elle est couverte par au moins toutes les variables couvrant l'autre. Dans ce cas elle peut être retirée du problème, la satisfaction de l'autre contrainte garantissant la satisfaction de la contrainte supprimée.

$$\begin{aligned} \text{Si } \tau_l \geq \tau_p \quad (\Leftrightarrow T_{lj} \geq T_{pj}, \forall j \in J) \\ \Rightarrow \tau_l \text{ est enlevé} \end{aligned} \quad (2.2)$$

Enfin, la troisième règle (2.3) sert à fixer à zéro les variables dites moins intéressantes. Pour les détecter, les variables sont comparées une à une afin de détecter si l'une d'elle couvre au moins les mêmes contraintes que l'autre pour un coût moindre ou égal (seulement moindre si l'on désire obtenir toutes les solutions optimales). Notons que ces comparaisons pourraient également être effectuées non pas seulement une à une mais entre des combinaisons de toutes tailles. Cependant, le temps nécessaire pour effectuer ces tests augmenterait de manière importante. Pour un problème à n variables, une comparaison une à une est de complexité n^2 , une comparaison deux à deux de complexité n^4 et une comparaison de toutes les combinaisons de complexité n^{n+1} . On voit bien que tester toutes les combinaisons est

une solution aussi coûteuse que l'énumération de toutes les solutions (et donc aussi peu envisageable pour des problèmes réels).

$$\begin{aligned} \text{Si } \exists j \in J, t_j \geq t_k \text{ et } C_j \leq C_k \\ \Rightarrow t_k \text{ est enlevé } (X_k = 0) \end{aligned} \quad (2.3)$$

Toutes ces règles sont ainsi testées jusqu'à ce qu'aucune d'entre elles ne puisse plus être activée. Bien évidemment, une fois le problème réduit résolu, la solution complète doit être reconstruite en y intégrant les variables fixées lors des tests de réduction (phase de *Reconstruction*).

2.1.3 Principes de parcours de l'arbre

Le principe utilisé pour ce Branch & Bound est celui de la profondeur d'abord. Les règles de sélection des nœuds à explorer sont fixées a priori et ne dépendent pas des évaluations obtenues pour les différents nœuds. De même, il n'y a pas de liste des nœuds à explorer car ceux-ci le sont selon l'ordre dans lequel ils sont obtenus, soit par une séparation, soit par un backtracking.

Un nœud est en fait défini par les variables qui y sont fixées en utilisant la notation usuelle de Balas-Geoffrion où l'ensemble des indices des variables fixées sont indiqués (les indices des variables fixées à zéro étant signalé par un signe négatif). Par exemple, le nœud de la Figure 2.2 correspond à l'ensemble $S = \{-3, 5, 8, -9\}$.

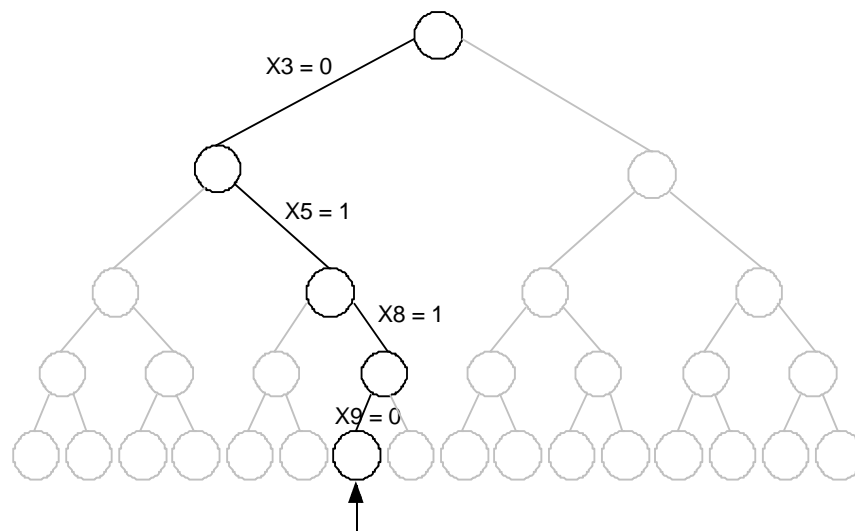


FIG. 2.2 – Exemple d'un nœud d'un Branch & Bound

Pour être complet, cette notation comprend aussi un pointeur permettant de connaître les nœuds déjà explorés. Cependant celui-ci n'est pas utile dans notre cas car comme nous le verrons, les règles de séparation et de backtracking employées évitent de revenir sur un nœud déjà rencontré.

Séparation d'un nœud intéressant

Lorsqu'un nœud a obtenu une évaluation permettant de supposer qu'il peut contenir une solution strictement meilleure que celles trouvées jusqu'à présent, et si la solution correspondant au problème relaxé n'est pas aussi une solution du sous-problème (dans ce cas la solution est entière et la règle de coïncidence peut s'appliquer), alors l'algorithme explore ce nœud. Pour cela, il cherche à fixer certaines variables encore libres en utilisant une heuristique de type glouton. Cette heuristique fonctionne en classant les variables dans l'ordre des $\frac{\sum_{i=1}^m T_{ij}}{C_j}$ décroissants. De plus elle fixe uniquement des variables à un, de sorte que le nœud obtenu est à une extrémité de l'arbre généré à partir du sous-problème évalué. Notons que cette heuristique est utilisée dès la première itération pour descendre vers une solution admissible.

Backtracking lorsqu'un nœud est sondé

Lorsque l'évaluation d'un nœud montre qu'il ne peut pas contenir de solution meilleure que celles trouvées ou que la solution optimale de ce nœud est connue, alors l'algorithme doit rechercher un nouveau nœud à explorer.

Pour cela il se déplace dans l'arbre en fixant à zéro des variables fixées à un. Il libère donc les dernières variables de S si celles-ci étaient fixées à zéro, et fixe à zéro la dernière variable fixée à un. Par exemple, si l'algorithme doit effectuer un backtracking à partir d'un nœud défini par l'ensemble $S^i = \{-3, 5, 8, -9\}$, il obtient le nœud défini par $S^{i+1} = \{-3, 5, -8\}$.

Notons que l'algorithme a fini de parcourir l'arbre lorsqu'il revient au nœud vide.

2.1.4 Évaluation d'un nœud

La relaxation continue est utilisée pour évaluer le sous-problème formé par les variables libres du nœud. Ce type de relaxation autorisant des évaluations peu coûteuses en temps, elle permet d'explorer un plus grand nombre de nœuds.

Malgré tout, lorsque le sous-problème à évaluer est très petit (peu de variables et/ou peu de contraintes), il est trivial et donc une solution entière optimale évidente peut facilement être trouvée sans qu'il soit nécessaire d'utiliser une relaxation.

2.2 Résolution approchée : la métaheuristique GRASP

La méthode approchée que nous avons choisi pour résoudre le Set Covering Problem est la métaheuristique GRASP¹ [Feo et al.94a]. Pour cela, nous avons repris et amélioré un algorithme développé par [Vancoppenolle98] et [Gandibleux et al.98] pour le Set Covering Problem. En particulier, nous avons travaillé sur des améliorations permettant de le rendre plus efficace sur les problèmes à coûts unitaires. L'architecture de l'algorithme de GRASP ainsi qu'un exemple didactique de son fonctionnement sont présentés en annexe III et IV.

¹Greedy Randomized Adaptative Search Procedure

GRASP peut en outre être facilement parallélisé pour une architecture multiprocesseur [Feo et al.94b]. Cependant, nous n'utiliserons pas cette propriété dans le cadre de notre travail.

2.2.1 Architecture de GRASP pour le SCP

Le principe général de GRASP (voir Figure 2.3) consiste à utiliser plusieurs fois un algorithme glouton choisissant les variables fixées de manière partiellement aléatoire puis à améliorer les différentes solutions ainsi trouvées grâce à une recherche locale. GRASP est donc un compromis entre un algorithme glouton² et un algorithme entièrement aléatoire, un paramètre à régler permettant de s'approcher plus ou moins de l'un ou de l'autre.

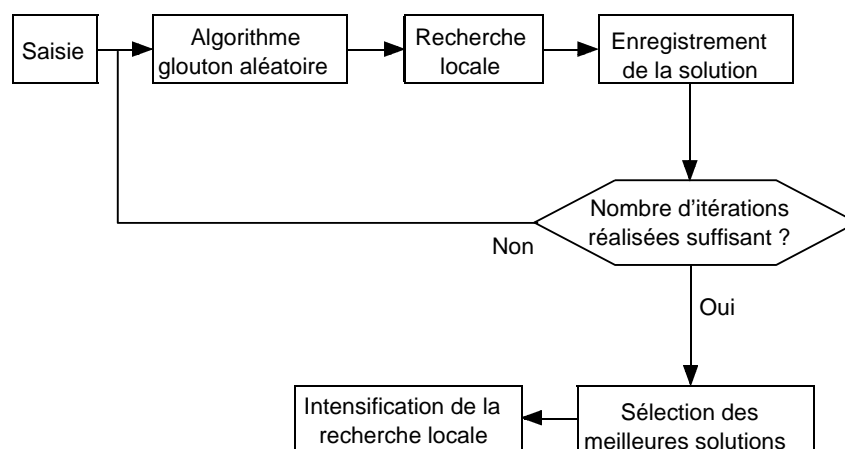


FIG. 2.3 – Architecture de GRASP pour le SCP

2.2.2 Obtention d'une solution initiale

La première phase de l'algorithme consiste à construire une solution initiale réalisable itérativement grâce à un algorithme glouton basé sur une fonction de sélection utilisant les principes suivants.

A chaque itération, les variables sont classées en fonction de leur intérêt à être sélectionnées dans une couverture minimum dans une liste de candidats CL (Candidate List). Cet intérêt est mesuré en faisant le rapport entre le nombre de contraintes satisfaites et le coût. Ce calcul est mis à jour à chaque itération en ne prenant en compte que les contraintes n'étant pas encore satisfaites par les variables déjà sélectionnées.

²Algorithme fixant successivement les variables par ordre d'intérêt uniquement jusqu'à obtenir une solution admissible

Une liste des meilleures candidates RCL (Restricted Candidate List) est ensuite obtenue en ne gardant que les variables ayant un intérêt supérieur ou égal à α

Notons aussi qu'un paramètre β peut être utilisé pour limiter le nombre de variables de la RCL. Cependant, ce paramètre n'a pas été utilisé.

Cette phase est répétée autant de fois que l'on désire de solutions initiales.

2.2.3 Recherche locale

La deuxième phase de l'algorithme consiste à améliorer les solutions obtenues avec l'algorithme glouton grâce à une recherche locale. En effet, rien ne garantit que ces solutions soient des optima locaux (au sens du voisinage considéré). La méthode de recherche locale utilisée est un algorithme de descente (k-p échanges³).

Évidemment, le choix d'un voisinage approprié sera très important pour obtenir un algorithme efficace. En effet, si celui-ci est trop restreint la recherche locale ne fournira pas de meilleures solutions, alors que s'il est trop vaste elle sera trop coûteuse en temps. Le voisinage proposé pour le SCP par [Vancoppenolle98] et [Gandibleux et al.98] est celui des 1-0 échanges (suppression des redondances).

2.2.4 Intensification de la recherche locale

Enfin, afin de compenser le fait que le voisinage choisi pour la recherche soit assez restreint, une nouvelle recherche locale est effectuée sur les meilleures solutions obtenues avec un voisinage plus vaste. Le fait que le nombre de recherches à effectuer soit moindre permet de conserver des temps de résolution corrects malgré le coût de ces recherches.

Cette partie de l'algorithme ne fait en réalité pas partie de l'algorithme original, mais est une modification proposée par [Vancoppenolle98]. Le voisinage qu'il utilisait était celui des 1-2 (suppression d'une variable de la solution et ajout de deux autres de coût total moindre tout en conservant une couverture), 1-1 et 1-0 échanges. Cependant, le choix de ce voisinage n'était pas toujours très approprié, en particulier pour les problèmes à coûts unitaires pour lesquels les 1-2 et 1-1 échanges ne peuvent être effectués étant donné qu'il n'existe pas de variables de coût moindre (toutes les variables étant de coût égal). Ces recherches ne sont donc pas effectuées dans ce cas. De plus, une recherche dans le voisinage des 2-1 échanges a été ajoutée pour l'ensemble des problèmes.

³Un k-p échange correspond au remplacement dans la solution de k variables par p variables n'appartenant pas à la solution

Chapitre 3

Expérimentation : comportement de GRASP sur le SCP

Afin de mesurer l'efficacité de GRASP pour résoudre des problèmes de type SCP, celui-ci a été testé sur une famille de problèmes de référence [Or library]. D'autres algorithmes ont aussi été testés sur ces problèmes afin de pouvoir comparer l'efficacité de GRASP par rapport à eux [Barr et al.95].

3.1 Algorithmes testés et critères d'évaluation

Trois types d'algorithmes ont été testés : des algorithmes gloutons, différentes variations de GRASP et un algorithme de Branch & Bound. Pour chacun d'eux la valeur du coût obtenue a été notée ainsi que le temps nécessaire à son obtention. De plus, pour l'algorithme de Branch & Bound, l'impact des tests de réduction (dimensions du problème réduit et temps utilisé) ainsi que le nombre d'itérations réalisées, de nœuds explorés et d'appels à l'algorithme du simplexe réalisés par l'algorithme ont aussi été notés. Enfin, le résultat de la relaxation linéaire des problèmes (et le temps nécessaire pour l'obtenir) a aussi été rapporté.

Trois variantes de l'algorithme glouton ont été utilisées. La première correspond à l'heuristique seule, alors que la deuxième inclut une relaxation linéaire (pour fixer des variables à 0) et la troisième une relaxation linéaire et des tests de réduction. Cette dernière version correspond donc à celle utilisée au début de l'algorithme de Branch & Bound.

Les deux premières versions de GRASP testées sont celles sans intensification de la recherche locale (GRASP 1), et avec l'intensification (GRASP 2) proposée par [Vancoppenolle98]. Une troisième version (GRASP 3) inclut l'amélioration de la recherche locale intensifiée. Les paramètres retenus pour ces trois versions sont les suivants :

- $\alpha = 0.85$
- 20 solutions initiales générées par l'algorithme glouton
- amélioration des 10 meilleures solutions trouvées

Une dernière version (GRASP5) a été réalisée suite aux premiers résultats enregistrés sur les autres versions de GRASP. Celle-ci favorise l'obtention de solutions initiales nombreuses

et assez diverses mais n'essaye d'améliorer que peu de solutions en raison du coût en temps de ces recherches locales intensifiées. Ses paramètres sont les suivants :

- $\alpha = \{0.85 ; 0.9 ; 0.95\}$
- 60 solutions initiales générées par l'algorithme gloutons (20 pour chaque valeur du coefficient a).
- amélioration des 3 meilleures solutions trouvées si elles sont de valeur égale. Sinon amélioration des 2 ou même uniquement de la meilleure valeur trouvée.

Enfin, l'algorithme du Branch & Bound faisait au départ appel au solveur en continu LPAKO [Lpako] afin d'effectuer les relaxations linéaires dont il avait besoin. En effet, ne disposant pas de produit commercial, nous avons choisi un code du domaine public et, ce solveur avait montré de bonnes performances (rapidité et robustesse face aux problèmes difficiles). Malheureusement il ne libère pas entièrement la mémoire qu'il utilise à chaque appel, ce qui pose des problèmes lors d'un grand nombre d'appels. Il a donc été remplacé par un autre solveur du domaine public, LPSOLVE [Lpsolve], qui s'il se révèle moins sophistiqué (et donc moins performant) se révèle stable au niveau de sa gestion de la mémoire. LPAKO a cependant été conservé pour les algorithmes gloutons qui ne nécessitent qu'une seule relaxation linéaire.

3.2 Matériel utilisé

L'ensemble des tests a été réalisé sur un Pentium III cadencé à 600 Mhz et disposant de 192 Mo de RAM et de 256 Ko de mémoire cache. Le système d'exploitation installé sur ce poste est Windows 98.

La plupart des algorithmes testés ont été implémentés dans le langage Ada 95 (Gnat v3.10) excepté les algorithmes LPAKO et LPSOLVE qui eux étaient écrits en langage C. Ce langage est celui qu'avait utilisé [Vancoppenolle98] en raison de la sécurité du code produit. Nous l'avons conservé pour les mêmes raisons.

3.3 Instances utilisées

Les instances utilisées ont été récupérées sur le site [Or library]. Celles-ci sont régulièrement utilisées pour réaliser des benchmarks. Le lecteur intéressé pourra trouver ces problèmes (et beaucoup d'autres) sur ce site ainsi que des références à des articles rapportant les résultats obtenues sur ces instances.

Les caractéristiques numériques prises en compte sont les dimensions de la matrice des contraintes (nombre de variables et de contraintes du problème), sa densité (c'est-à-dire le pourcentage de 1 dans la matrice) et aussi le nombre maximum de 1 par ligne. Nous avons sélectionné des instances parmi 7 familles (41, 61, a1, e1, nre1, cyc6, clr10) en nous assurant de couvrir une vaste gamme de caractéristiques numériques (voir Tableau 3.1).

Cependant, la taille de ces problèmes étant suffisamment importante pour mettre à mal, voire compromettre une méthode exacte, des versions réduites de ces instances ont aussi été produites. Dans ce cas un 'r' a été ajouté à la fin du nom de l'instance (voir même 'r1' et

Nom	Variables	Contraintes	Densité (%)	Max-Uns
SCP41	1000	200	2.0	30
SCP41r	500	100	2.0	16
SCP61	1000	200	5.0	68
SCP61r	500	100	4.9	38
SCPA1	3000	300	2.0	81
SCPA1r1	500	50	1.8	17
SCPA1r2	1000	100	2.0	30
SCPE1	500	50	20.0	116
SCPE1r1	200	20	24.8	67
SCPE1r2	100	10	29.3	41
SCPNRE1	5000	500	10.0	560
SCPNRE1r1	200	20	10.7	31
SCPNRE1r2	100	10	11.2	14
SCPCYC6	192	240	2.1	4-4
SCPCLR10	210	511	12.3	10-126

TAB. 3.1 – Caractéristiques numériques des instances sélectionnées

'r2' dans le cas de différentes réductions). Ces versions ont été obtenues en préservant au maximum les caractéristiques numériques de ces problèmes (rapport du nombre de variable sur le nombre de contraintes, densité).

De plus, les problèmes dont les coûts n'étaient pas unitaires ont aussi été testés avec des coûts unitaires. Un 'U' a alors été ajouté à la fin du nom de l'instance. Enfin, deux instances ayant des structures particulières et connues pour être difficiles ont été retenues. Au total 25 instances ont ainsi été utilisées.

3.4 Résultats obtenus

Afin d'évaluer le potentiel de GRASP, les résultats obtenus avec ses différentes variations ont été comparés, puis ont été successivement comparés à ceux obtenus avec les autres algorithmes et aux meilleurs résultats trouvés dans la littérature. Dans la suite, seuls les exemples caractéristiques des différents cas de figure rencontrés sont discutés. Cependant la liste exhaustive des résultats obtenus sur chacun des problèmes traités est rapportée en annexe V.

Du fait de son caractère non déterministe, toutes les expérimentations mettant en œuvre GRASP ont été répétées 10 fois afin de pouvoir considérer un comportement moyen.

3.4.1 Comparaison des différentes versions de GRASP

Les paramètres pris en compte pour comparer les différentes versions de GRASP sont les temps d'exécution et les valeurs obtenues.

Temps d'exécution

Des différences de temps significatives ont pu être observées sur la plupart des instances. Ces différences peuvent souvent être considérées comme négligeables sur les instances de petite taille. Par contre, sur les instances de plus grande taille, ce n'est plus le cas. Ainsi, si l'on prend les temps réalisés par GRASP 1 comme référence on obtient les rapports de temps suivants avec les autres versions :

- GRASP 2, 2 à 4 fois plus lent
- GRASP 3, 4 à 30 fois plus lent
- GRASP 5, 3 à 7 fois plus lent

De plus, si les temps obtenus avec GRASP 1, 2 et 3 varient peu d'une expérimentation à l'autre, ce n'est pas le cas pour GRASP 5 en raison du nombre variable de recherche locale réalisé. L'instance SCP41U caractérise bien ce phénomène avec des temps de résolution allant de 182.5 à 314 secondes suivant les expérimentations.

Qualité des solutions obtenues

Dans un certain nombre d'instances (comme le SCPNRE1 par exemple), la qualité des solutions obtenues est approximativement identique. Par contre, des différences peuvent être relevées sur les autres problèmes. On observe ainsi quatre cas différents (Figures 3.1 à 3.4).

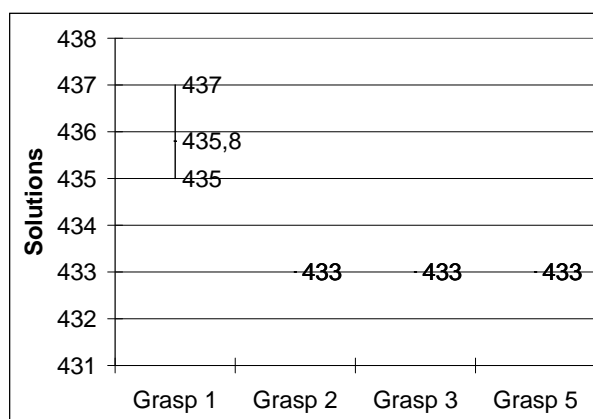


FIG. 3.1 – SCP41

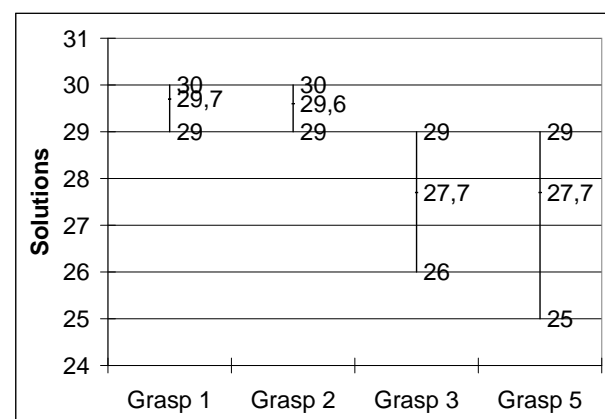


FIG. 3.2 – SCPCLR10

Dans tous ces cas, GRASP 1 se montre systématiquement le moins bon alors que GRASP 2 est plus nuancé en obtenant dans certains cas des résultats de bonne qualité (voir Figure 3.1). Dans le cas général, GRASP 3 et GRASP 5 fournissent des solutions de meilleure qualité que GRASP 1 et GRASP 2 (voir Figure 3.2).

Par contre il est difficile de prendre position entre GRASP 3 et GRASP 5. En effet, suivant les instances les meilleurs résultats sont obtenus soit avec l'un, soit avec l'autre (voir Figure 3.3 et 3.4).

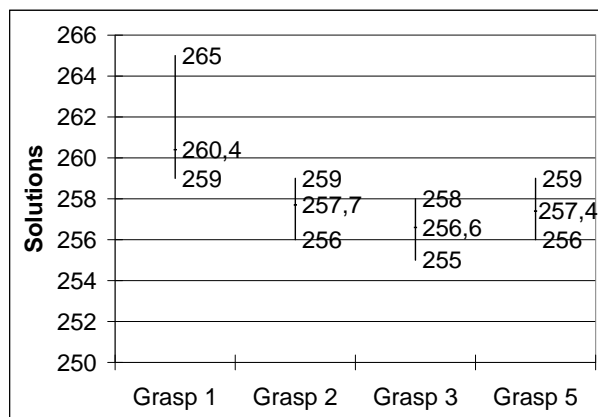


FIG. 3.3 – SCPA1

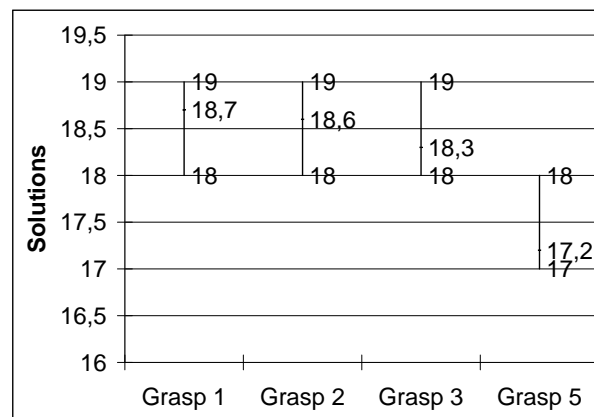


FIG. 3.4 – SCPNRE1U

Synthèse

Il n'existe pas une version de GRASP strictement dominante par rapport à une autre. Malgré tout, la version GRASP 5 présente un bon compromis entre la qualité de la solution obtenue et le temps de calcul. Pour une utilisation avec un temps de calcul très limité, GRASP 1 peut présenter une bonne alternative avec des résultats corrects et des temps d'exécution plus réduits. Les deux autres versions ont par contre moins d'intérêt. Ainsi, GRASP 2 présente des temps de réponse trop proches de GRASP 5 étant donné la différence de qualité des solutions obtenues et n'améliore que peu les solutions obtenues par GRASP 1 (notamment pour les problèmes à coûts unitaires). Quant à GRASP 3, il se montre beaucoup trop lent sans pour autant assurer une meilleure qualité que GRASP 5.

3.4.2 Comparaison de GRASP avec des algorithmes gloutons

Tout d'abord, aucun des trois algorithmes gloutons essayés n'est clairement plus performant que les autres en terme de qualité de solutions obtenues. Par contre, les temps d'exécution peuvent différer de manière assez importante, principalement entre le premier dont les temps de réponse ne dépassent pas 10 secondes et le troisième qui peut utiliser jusqu'à 3 minutes. Cette différence s'explique par le coût en temps des tests de réduction.

Par rapport à GRASP, les algorithmes gloutons sont beaucoup plus rapides (de l'ordre de quelques secondes contre quelques minutes pour GRASP). Au niveau de la qualité des solutions obtenues, on s'aperçoit que dans certains cas particuliers l'un des algorithmes glouton peut se révéler meilleur que GRASP (voir Figure 3.5 et 3.6).

Ces cas apparaissent soit pour des problèmes dont la structure semble bien adaptée à l'un des algorithmes gloutons (SCPCYC6), soit quand la solution obtenue après relaxation linéaire du problème est entière. Les algorithmes gloutons 2 et 3 parviennent alors à trouver la solution optimale et sont donc au moins aussi performants que GRASP (cas du SCP41 et de cinq problèmes réduits). Toutefois, il s'agit d'une situation isolée car les résultats obtenus

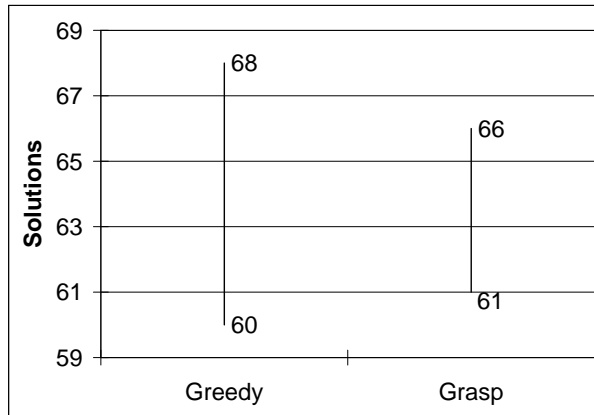


FIG. 3.5 – SCPCYC6

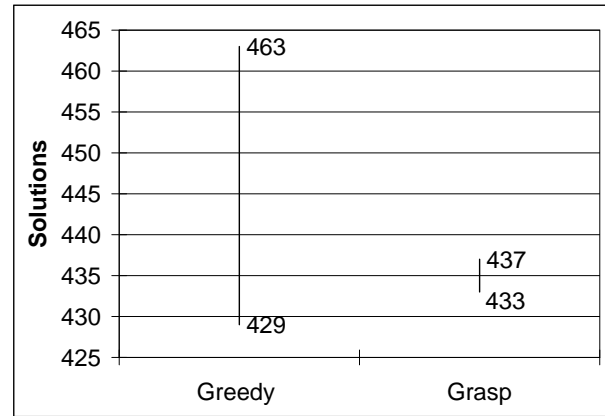


FIG. 3.6 – SCP41 (2)

par les autres algorithmes gloutons sont largement moins bons que ceux fournis par GRASP. Cependant, ces deux cas ne se présentent que rarement (7 instances sur 25 dont 2 seulement qui correspondent à des instances non réduites).

Dans le cas général (voir Figure 3.7 et 3.8), les solutions gloutonnes sont moins bonnes que celles de GRASP (SCPCLR10) et peuvent même en être très éloignées (SCP61).

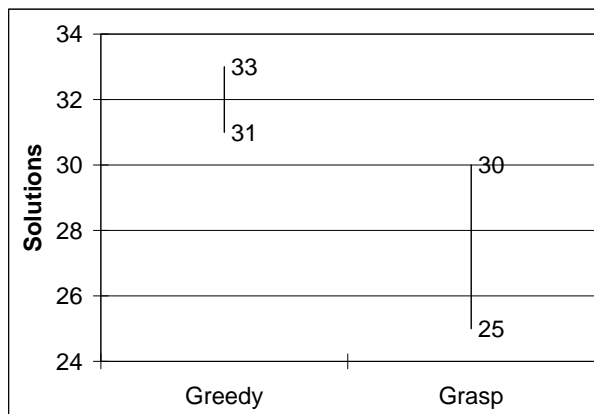


FIG. 3.7 – SCPCLR10 (2)

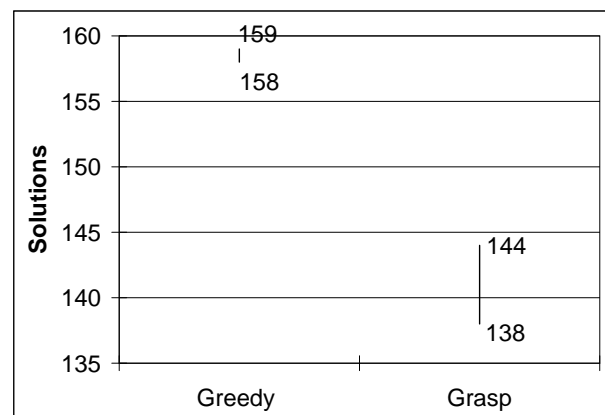


FIG. 3.8 – SCP61

3.4.3 Comparaison de GRASP avec un algorithme exact

Comme cela a déjà été mentionné, lorsque le problème relâché conduit à une solution entière, l'algorithme exact s'est montré plus rapide que GRASP. Hormis ce cas particulier de «faux problème», l'algorithme exact s'est révélé beaucoup plus lent voir incapable de résoudre la plupart des instances non réduites en un temps raisonnable (c'est-à-dire pour nous inférieur à deux ou trois jours) alors que GRASP pouvait fournir des solutions à ces instances en quelques minutes. De plus, il s'avère que les meilleurs résultats trouvés par

GRASP sont souvent assez proches voir même égaux à la solution optimale. Ainsi, sur le SCPA1Ur2 (par exemple), la meilleure solution trouvée par GRASP a une valeur de 25. C'est une solution optimale. Le temps de réponse de GRASP sur cette instance est de 60 secondes alors que celui du Branch & Bound est de plus de 172 000 secondes (environ 48 heures). Notons aussi que l'algorithme de Branch & Bound est mis à mal pour résoudre les problèmes à coûts unitaires.

3.4.4 Comparaison de GRASP avec les meilleurs résultats connus

Nos résultats peuvent être comparés aux valeurs des solutions optimales lorsque celles-ci sont connues (voir Tableau 3.2). Ces solutions optimales proviennent soit de la résolution de ces instances par l'algorithme de Branch & Bound, soit sont rapportées dans [Beasley 87]. Les résultats obtenus par GRASP se révèlent très proches des solutions optimales (pas plus d'1 % de différence).

	SCP41	SCP61	SCPA1	SCPE1
Solution Optimale	429	138	253	5
GRASP	433	138	255	5
Différence	0.9 %	0 %	0.8 %	0 %

TAB. 3.2 – Comparaison de GRASP avec les solutions optimales connues

Lorsque les solution optimales ne sont pas connues, les résultats peuvent malgré tout être confrontés avec les meilleurs solutions connues rapportées dans [Grossman et al.97]. La comparaison (voir Tableau 3.3) présente GRASP comme étant aussi performant que les autres heuristiques sur pratiquement toutes les instances, se révélant même parfois meilleur. La seule instance pour laquelle GRASP trouve un moins bon résultat est le SCPCYC6. Notons malgré tout que ce problème est particulier étant donné que la meilleure solution trouvée par [Grossman et al.97] a été obtenue à l'aide d'un simple algorithme glouton (et a d'ailleurs aussi été trouvée par l'un de nos algorithmes gloutons).

	SCP41U	SCP61U	SCPA1U	SCPE1	SCPNRE1U	SCPCYC6	SCPCLR10
Littérature	41	21	40	5	17	60	28
GRASP	40	21	40	5	17	61	25

TAB. 3.3 – Comparaison de GRASP avec les meilleurs résultats de la littérature

3.5 Conclusion de l'étude du comportement de GRASP sur le SCP

Au vu des expérimentations réalisées, GRASP se montre tout à fait capable de fournir des solutions de bonne qualité au contraire d'algorithmes moins sophistiqués comme les gloutons. De plus, il permet d'obtenir des solutions sur des problèmes de grande taille avec

des temps de calculs réalistes alors que les algorithmes exacts montrent leurs limites. Ces résultats valident donc l'intérêt de GRASP.

Nous allons donc maintenant pouvoir nous intéresser à un problème de capacité ferroviaire et à sa résolution par un algorithme basé sur GRASP.

Chapitre 4

Le problème de capacité ferroviaire

4.1 Position du problème

Parmi les problèmes liés à la problématique de l'exploitation ferroviaire, on peut distinguer deux grandes catégories : la planification (gestion prévisionnelle) et l'exploitation en temps réel (décisions opérationnelles). Un glossaire précisant la définition des principaux termes ferroviaires employés est présenté en annexe VI.

Les principales différences entre ces deux catégories se situent au niveau de leur horizon temporel et de la nature des informations utilisées. Ainsi, la planification a essentiellement pour finalité la conception et l'évaluation de projets d'aménagements (nouvelle **ligne**, saut de mouton, matériel roulant, abandon d'une partie des installations...) ou de la programmation de l'exploitation (horaires) à moyen et long terme. Par contre, l'exploitation en temps réel correspond à l'adaptation des horaires établis par rapport aux problèmes ponctuels rencontrés (**blanc-travaux**, retards de certains trains...) et, si elle a la même finalité que la programmation de l'exploitation, elle se situe dans un horizon temporel beaucoup plus réduit (conduite de l'exploitation).

Le problème de l'évaluation de la capacité trouve tout son intérêt dans le cadre d'un processus de planification ferroviaire. En effet, elle permet de concevoir et d'évaluer une offre répondant à la demande future (c'est-à-dire avec un taux de saturation acceptable¹) sans surdimensionner les investissements.

Selon [Hachemane97], la notion de capacité peut se définir comme le nombre maximum de trains pouvant circuler dans un intervalle de temps donné dans des conditions pratiques d'exploitation² et pour une structure de lignes, une structure d'horaire et une qualité de service données. C'est donc une notion complexe faisant intervenir de nombreux facteurs et paramètres pour lesquels on ne s'étendra pas. Le lecteur intéressé pourra trouver dans [Bellaïche97] et dans [Schneider97] des précisions sur leur importance et une étude de leur impact sur la capacité dans des cas réels :

- les infrastructures (nombre de voies, signalisation, vitesse autorisée, maintenance, ci-

¹Taux de saturation $S = \frac{\text{Nombre de trains}}{\text{Capacité pratique}}$, un niveau de saturation supérieur à 100 % n'étant pas aberrant mais correspondant à une saturation ne respectant pas strictement les normes de qualité recommandées.

²Incluant des marges pour tenir compte des variations inévitables dans la pratique (tension électrique, patinage des roues), du mécanicien (temps de réaction, conduite) et des conditions de sécurité.

saillements de voies...)

- le plan de transport (ordonnancement des trains, hétérogénéité des circulations, contraintes horaires dues aux correspondances ou aux dessertes de voyageurs...)
- la qualité de l'exploitation (ou **qualité de service**) qui dans le cas présent est assimilée à la stabilité de l'horaire vis à vis des perturbations (travaux, retards...) et donc est liée à l'importance des marges prévues lors de l'élaboration de la **grille horaire** (afin d'éviter les effets «boule de neige»).
- les caractéristiques techniques des trains (vitesse, accélération, freinage, longueur...)

La capacité ainsi définie correspond à une capacité pratique à la différence de la capacité théorique qui ne prend pas en compte la qualité de service et qui représente donc la capacité maximale atteignable techniquement.

Dans le cadre de l'étude d'un **nœud** ferroviaire (gare, point de convergence ou de divergence de lignes) pour une infrastructure donnée et un horaire établi, plusieurs questions liées à la capacité peuvent être étudiées :

- la faisabilité du passage des trains dans le nœud
- la fluidification du passage des trains dans le nœud (c'est-à-dire l'amélioration de la qualité de service)
- la capacité résiduelle ou absolue du nœud
- la modification d'un élément physique du système d'exploitation ferroviaire

Lors de ce travail, nous aborderons uniquement la question de la faisabilité du passage des trains.

4.2 Difficulté du problème

Suivant la typologie de l'infrastructure (voir Figure 4.1) dont on cherche à évaluer la capacité, la difficulté de cette évaluation peut énormément varier.

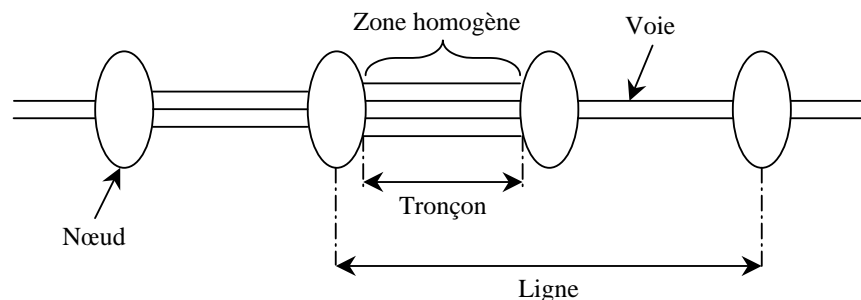


FIG. 4.1 – Éléments d'une infrastructure ferroviaire

Ainsi, si la capacité d'une zone homogène (c'est-à-dire ne contenant pas d'aiguillage et ne permettant ni l'arrêt ni le changement du sens de circulation des trains) comme un **tronçon** peut être facilement calculé en effectuant la somme des capacités des différentes voies parallèles qui le composent, l'additivité n'est plus applicable dès que l'on considère des zones hétérogènes. De même, il n'est pas possible de calculer la capacité d'une ligne ou d'un nœud (et a fortiori d'un **réseau** complet) en combinant à l'aide d'une fonction quelconque

(comme le minimum ou le maximum par exemple) les capacités des différents éléments qui la composent.

D'après [Bellaïche97], on peut distinguer quatre types de méthodes pouvant servir à évaluer la capacité :

- Les formules analytiques qui se basent sur une évaluation de la moyenne des temps minimums de succession³ entre les différents trains. On peut citer parmi elles les méthodes **UIC**, **CFF**, **SIMON**, **FS** et **NS**. Ces méthodes ont l'avantage d'être simples et rapides à mettre en œuvre mais imposent l'utilisation d'hypothèses souvent trop réductrices pour l'étude des zones hétérogènes.
- Les méthodes probabilistes qui se basent sur une évaluation probabiliste de la répartition des trains et font des hypothèses sur la distribution de circulation. On peut citer parmi elles les méthodes **DB**, **Schwanhäusser** et **DGCFF**. Là encore, si elles sont assez faciles à utiliser, ces méthodes imposent des hypothèses souvent trop contraignantes.
- Les méthodes de construction d'horaires qui partent d'une grille horaire donnée pour élaborer la grille la plus dense possible qui correspond à la situation où le niveau de saturation est maximum. On peut citer parmi elles les méthodes du compactage graphique, de la **saturation** graphique, la méthodologie de **CAPRES**, du **sillon** standard et de **SERGOB**. Ces méthodes peuvent être assez efficaces même sur des zones assez complexes. Leur inconvénient principal est soit de ne pas gérer les conflits éventuels, soit de nécessiter une base de données importante.
- Les méthodes de simulation qui n'effectuent aucun calcul théorique mais se contentent de simuler la circulation des différents trains connus et des différents événements survenant sur le réseau afin de se rendre compte «de visu» du niveau de qualité et de robustesse d'une grille. On peut citer parmi elles les logiciels **FASTA**, **RAILSIM** et **SISYFE**. Ces méthodes permettent une bonne évaluation de la plupart des cas mais nécessitent une base de donnée importante et surtout se limitent à l'évaluation des grilles horaires et non pas de la capacité en elle-même.

On voit donc bien que si ces différentes méthodes peuvent être utiles dans certains cas, aucune ne se révèle idéale pour tous les cas de figure.

4.3 Les travaux des «Nederlandse Spoorwegen»

Ils s'inscrivent dans le cadre du projet **DONS**⁴ développé par les chemins de fer néerlandais (NS) et dont l'objectif est d'aider à planifier une grille horaire pour l'ensemble du réseau hollandais. Le résultat de ce projet est un logiciel comprenant deux modules complémentaires. Le premier (appelé **CADANS**) tente d'établir une grille horaire (**cadencée** à une heure) pour le réseau complet, alors que le deuxième (appelé **STATIONS**) aide à résoudre les problèmes de **routage** dans chaque gare afin de vérifier la faisabilité de la grille horaire proposée par **CADANS**. Dans le cas où le routage de tous les trains ne peut pas être réalisé, il met en évidence les trains bloquant. Le module correspondant à notre champ de questions est

³Les temps minimum de succession sont déterminés grâce aux caractéristiques techniques des trains et des infrastructures et majorés en fonction des normes de sécurité et des niveaux de qualité souhaités.

⁴Design Of Network Schedules

STATIONS. L'objectif des travaux autour de STATIONS [Kroon et al.97] est de réaliser le routage d'un nombre maximum de trains dans un nœud ferroviaire sur une période de temps donnée.

4.3.1 Description générale du modèle

Un nœud ferroviaire est caractérisé par un certain nombre de points d'entrée, de quais et de points de sortie. Un train parcourant ce nœud emprunte donc un itinéraire d'entrée (composée de plusieurs sections) depuis son point d'entrée (dépendant de la provenance du train) jusqu'à un quai. Puis, il emprunte un itinéraire de sortie de ce quai jusqu'à son point de sortie (dépendant de sa direction de voyage). Un itinéraire complet correspond à la combinaison d'un itinéraire d'entrée et d'un itinéraire de sortie utilisant le même quai⁵. Les heures d'arrivée et de départs considérées pour un train correspondent aux moments où il s'arrête et où il quitte le quai (et non pas le nœud). Dans le cas d'un train au passage, ces deux heures sont égales.

Ainsi, on peut modéliser ce problème en considérant l'ensemble des trains T , l'ensemble des itinéraires R (l'ensemble $R_t \in R$ représentant les itinéraires possibles⁶ pour un train $t \in T$) et l'ensemble des variables de décision $X_{t,r} = \begin{cases} 1 & \text{si le train } t \in T \text{ utilise l'itinéraire } r \in R_t \\ 0 & \text{dans les autres cas} \end{cases}$. L'ensemble $F_{t,t'}$ contient toutes les combinaisons d'itinéraires possibles pour deux trains t et t' . Il peut donc se formuler comme le problème en variables binaires suivant (4.1) :

$$\left[\begin{array}{l} \text{Max } z = \sum_{t \in T} \sum_{r \in R_t} X_{t,r} \\ \sum_{r \in R_t} X_{t,r} \leq 1 \quad , \forall t \in T \quad (a) \\ X_{t,r} + X_{t',r'} \leq 1 \quad , \forall (t, t') \in T^2, r \in R_t, r' \in R_{t'}, (r, r') \notin F_{t,t'} \quad (b) \\ X_{t,r} \in \{0, 1\} \quad , \forall t \in T, r \in R_t \end{array} \right] \quad (4.1)$$

où les contraintes (a) modélisent le fait qu'un train ne peut prendre qu'un seul parcours et les contraintes (b) le fait que le passage de deux trains sur deux parcours non compatibles ne peut pas être réalisé en même temps.

C'est donc un Set Packing Problem. Le nombre de contraintes de ce problème peut d'ailleurs être diminué [Zwaneveld et al.96] en l'exprimant à l'aide de la formulation suivante (4.2) :

⁵La décomposition des itinéraires en itinéraires d'entrée et en itinéraires de sortie permet ainsi de modéliser le couplage ou le découplage de certains trains.

⁶La décomposition des itinéraires permet donc aussi de diminuer le nombre d'itinéraires : si un train a respectivement n_1 et n_2 itinéraires d'entrée et de sortie et $|P|$ quais possibles, le nombre d'itinéraires complets possibles passera de $n_1 * n_2 * |P|$ à $(n_1 + n_2) * |P|$.

$$\left[\begin{array}{l} \text{Max } z = \sum_{t \in T} \sum_{r \in R_t} X_{t,r} \\ \sum_{r \in R_t} X_{t,r} \leq 1 \quad , \forall t \in T \\ X_{t,r} + \sum_{r', (r,r') \notin F_{t,t'}} X_{t',r'} \leq 1 \quad , \forall (t, t') \in T^2, r \in R_t \quad (c) \\ X_{t,r} \in \{0, 1\} \quad , \forall t \in T, r \in R_t \end{array} \right] \quad (4.2)$$

où les contraintes (c) correspondent à un compactage des contraintes (b) de l'équation 4.1.

La formulation finalement retenue par [Zwaneveld et al.97] est celle d'un Node Packing Problem (4.3) :

$$\left[\begin{array}{l} \text{Max } z = \sum_{t \in T} \sum_{r \in R_t} X_{t,r} \\ X_{t,r} + X_{t',r'} \leq 1 \quad , \forall t \in T, (r, r') \in R_t^2, r \neq r' \\ X_{t,r} + X_{t',r'} \leq 1 \quad , \forall (t, t') \in T^2, r \in R_t, r' \in R_{t'}, (r, r') \notin F_{t,t'} \\ X_{t,r} \in \{0, 1\} \quad , \forall t \in T, r \in R_t \end{array} \right] \quad (d) \quad (4.3)$$

où les contraintes (d) correspondent aux contraintes (a) de l'équation 4.1 en considérant les parcours d'un même train deux à deux.

En effet, si cette formulation entraîne une augmentation importante du nombre de contrainte, celle-ci est compensée par les techniques de résolution utilisées qui profitent de cette structure particulière et des propriétés qui en découlent.

4.3.2 Extensions du modèle

Deux extensions ont aussi été proposés par [Zwaneveld et al.96] et par [Zwaneveld et al.97]. La première permet d'autoriser un léger décalage des horaires d'arrivée et de départ des trains [Zwaneveld et al.96]. Ces deux décalages sont exprimés par une déviation $\delta = (\delta_a, \delta_d)$. Ainsi, l'ensemble $F_{t,t'}$ contient toutes les combinaisons de itinéraires-déviation possibles pour les trains t et t' , et R_t les itinéraires-déviation possibles pour le train t . On obtient alors la formulation suivante (4.4) :

$$\left[\begin{array}{l} \text{Max } z = \sum_{t \in T} \sum_{(r,\delta) \in R_t} X_{t,r,\delta} \\ X_{t,r,\delta} + X_{t',r',\delta'} \leq 1 \quad , \forall t \in T, ((r, \delta), (r', \delta')) \in R_t^2, (r, \delta) \neq (r', \delta') \\ X_{t,r,\delta} + X_{t',r',\delta'} \leq 1 \quad , \forall (t, t') \in T^2, (r, \delta) \in R_t, (r', \delta') \in R_{t'}, ((r, \delta), (r', \delta')) \notin F_{t,t'} \\ X_{t,r,\delta} \in \{0, 1\} \quad , \forall t \in T, (r, \delta) \in R_t \end{array} \right] \quad (4.4)$$

La seconde permet d'exprimer les préférences de certains trains pour un itinéraire particulier (par exemple des choix de quais pour des correspondances ou des raisons de convenance des usagers qui préfèrent qu'un même type de train parte toujours du même quai) [Zwaneveld et al.97]. Dans ce cas, seule la fonction économique du modèle est modifiée par l'introduction d'un paramètre $\rho_{t,r}$ qui vient pondérer les différentes variables $X_{t,r}$.

4.3.3 Techniques de résolution

La méthode utilisée pour résoudre de manière exacte ce problème pour une grille horaire cadencée à l'heure [Zwaneveld et al.97] est composée de quatre phases :

- L'initialisation qui sert à déterminer toutes les variables (itinéraires pour un train) ainsi que les itinéraires compatibles. Pour cela, les seules sections à considérer sont celles contenant un aiguillage, une intersection, un point d'entrée, un quai ou un point de sortie, un conflit entre deux trains ne pouvant apparaître dans les autres. Ces sections sont dites déterminantes.
- Une phase de réduction qui utilise successivement quatre tests valides pour un Node Packing Problem :
 - Node dominance (suppression d'une variable dominée par une autre)
 - Set dominance (suppression d'une variable dominée par un ensemble de variables)
 - Iterative set dominance (suppression d'un ensemble de variables dominé par un autre ensemble)
 - Combining nodes (combinaison de deux variables en une seule si la sélection de l'une dans une solution optimale entraîne la sélection de l'autre)
- La formulation du problème sous la forme d'un Node Packing Problem.
- Un algorithme d'énumération implicite avec ajout de contraintes (Branch & Cut) utilisant une relaxation linéaire. Cependant, celui-ci pouvant nécessiter un temps important, le problème à résoudre ne doit plus être de trop grande taille.

4.4 Adaptation de GRASP au Set Packing Problem

Afin de résoudre des problèmes ferroviaires formulés selon la modélisation présentée ci-dessus (Description générale du modèle), nous avons développé un algorithme approché dérivé de la métaheuristique GRASP (dont nous avons pu mesurer l'efficacité vis à vis d'un algorithme exact dans la partie 3.4.3). En effet, comme nous l'avons vu, GRASP est une métaheuristique et peut donc être utilisée pour résoudre différents problèmes. Cependant, dans le contexte d'une telle résolution nous avons opté pour la formulation en Set Packing Problem, les propriétés du Node Packing n'étant plus exploitées du fait de la non utilisation des tests de réductions (peu intéressants dans le cadre d'une méthode approchée comme nous l'avons vu en partie 3.4.2).

A notre connaissance il n'existe pas d'adaptation de GRASP pour résoudre les Set Packing Problem dans la littérature. Nous avons réalisé cette adaptation en repartant de notre expérience sur le Set Covering Problem. De même, nous avons gardé les paramètres de la version de GRASP ayant donné les meilleurs résultats c'est-à-dire GRASP5. Les différences se situent à deux niveaux :

- L'algorithme glouton aléatoire permettant d'obtenir une solution initiale qui selon le même principe part d'une solution non admissible pour construire progressivement une solution admissible. Mais dans le cas d'un Set Packing Problem, la solution de départ retenue consiste à sélectionner toutes les variables (alors que pour le Set Covering on n'en sélectionnait aucune). Les variables sont ensuite classées par ordre décroissant selon le rapport entre le nombre de contraintes utilisées et leur valeur. Des variables sont ainsi retirées successivement de l'ensemble jusqu'à ce que la solution soit admissible.
- Les recherches locales utilisées qui sont des 0-1 échanges et des 1-2 échanges (ainsi que des 1-1 échanges et des 2-1 échanges pour les problèmes à coûts pondérés).

Chapitre 5

Expérimentation : le nœud ferroviaire de Gonesse

La situation que nous avons choisi d'étudier pour expérimenter l'aptitude de GRASP à résoudre des problèmes ferroviaires réels est le nœud de Pierrefitte-Gonesse. Ce nœud se révèle très intéressant à étudier en raison du nombre et de l'hétérogénéité des trains qui le parcourent et des difficultés qui en découlent. Il diffère quelque peu des travaux présentés au Chapitre 4 du fait qu'il ne s'agit pas d'une gare. Cependant une modélisation semblable peut être réalisée. Le plan détaillé du nœud est présenté en annexe VII.

5.1 Présentation de la situation

Le nœud de Pierrefitte-Gonesse [Auguet96] est situé à une dizaine de kilomètres de la gare de Paris Nord et s'étend sur environ seize kilomètres. Il représente une sorte de carrefour entre quatre grandes directions (hors lignes RER) et ses voies se croisent en de nombreux endroits (voir Figure 5.1). On peut principalement distinguer trois grands types de voies dans le nœud de Pierrefitte-Gonesse :

- les voies grande ligne reliant Paris Nord et Chantilly
- les voies grande vitesse reliant Paris Nord à Lille
- les voies reliant Chantilly à la Grande Ceinture

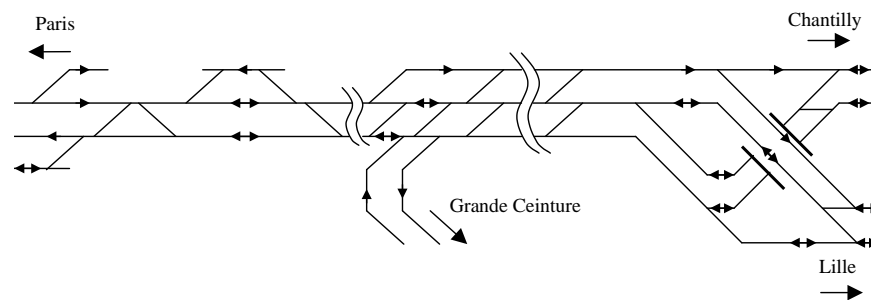


FIG. 5.1 – Schéma des voies du nœud de Pierrefitte-Gonesse

Ainsi, les différents types de trains suivants circulent à travers ce nœud de manière quotidienne :

- des trains grande vitesse comme le TGV (à destination du nord de la France), l'Eurostar (à destination de l'Angleterre) et Thalys (à destination de la Belgique) circulant dans les deux sens. Ceux-ci empruntent habituellement les lignes grande vitesse mais peuvent occasionnellement être détournés vers les voies grande ligne. Ce type de trafic est important et devrait encore augmenter dans les années à venir.
- des trains voyageurs «classiques» entre Paris Nord et Chantilly dans les deux sens. Là encore le trafic de ce type de train est important. Il ne devrait cependant pas trop évoluer durant les prochaines années.
- des trains marchandises entre Chantilly et la Grande Ceinture. Ce type de trafic est plus limité que les deux précédents mais ces trains sont aussi beaucoup plus lents.

De plus, d'autres trains peuvent aussi emprunter ce nœud de manière occasionnelle :

- des trains de banlieue détournés empruntant les voies grande ligne (la gare RER de Pierrefitte-Gonesse se trouvant dans la zone étudiée).
- des trains spéciaux ou exceptionnels dont les parcours dans le nœud peuvent être divers. La circulation d'une motrice pour un besoin de dépannage est un exemple de ce type de trafic.

On voit donc bien que cet important trafic peut occasionner de nombreux cisaillements, et notamment les trains de marchandises qui coupent l'ensemble des autres voies en se déplaçant à une vitesse très inférieure aux autres trains. De plus le grand nombre de voies autorisant la circulation dans les deux sens n'améliore pas la situation. La détermination de la capacité d'un tel nœud est donc loin d'être une chose aisée.

5.2 Présentation de l'expérimentation

5.2.1 Modèle utilisé

Pour l'essentiel, le modèle que nous utilisons est celui présenté dans les parties Description générale du modèle et Extensions du modèle. Cependant, quelques modifications ont dû y être apportées afin de l'adapter au problème étudié et aux techniques de résolution employées.

Les plus importantes modifications du modèle proviennent du fait que le nœud considéré n'est pas une gare et que donc les trains ne doivent a priori pas stationner le long d'un quai. La différenciation entre les itinéraires d'entrée et de sortie n'a plus d'utilité. Ainsi, nous ne conservons que la notion d'itinéraire complet. De même, les heures d'arrivées et de départs considérées pour déterminer les conflits potentiels ne peuvent plus être que celles correspondant à l'entrée et à la sortie du nœud. L'amélioration faisant appel aux différentes déviations possibles n'est donc plus envisageable sans répercuter les problèmes rencontrés à l'extérieur du nœud.

L'autre amélioration proposée n'a pas non plus été retenue. En effet, le problème qui nous intéressait dans le cadre de ce travail concernait uniquement la faisabilité du passage des trains. La prise en compte de l'intérêt des trains pour certains itinéraires n'entre donc pas en ligne de compte. De toute façon, l'absence de quai supprime les problèmes de correspondances et de convenance des usagers et limite donc grandement son utilité.

Comme nous avons vu dans la partie Adaptation de GRASP au Set Packing Problem qu'étant donné la méthode de résolution que nous employons, la forme algébrique la plus adaptée était celle d'un Set Packing Problem, nous avons retenu la formulation (5.1) décrite dans la partie 4.3.1 :

$$\left[\begin{array}{l} \text{Max } z = \sum_{t \in T} \sum_{r \in R_t} X_{t,r} \\ \sum_{r \in R_t} X_{t,r} \leq 1 \quad , \forall t \in T \\ X_{t,r} + \sum_{r', (r,r') \notin F_{t,t'}} X_{t',r'} \leq 1 \quad , \forall (t, t') \in T^2, r \in R_t \\ X_{t,r} \in \{0, 1\} \quad , \forall t \in T, r \in R_t \end{array} \right] \quad (5.1)$$

5.2.2 Techniques de résolution employées

Étant donné l'intérêt limité montré par les règles de réduction lors de résolutions par des méthodes approchées (temps nécessaire très important par rapport au temps de résolution) constaté lors des expérimentations sur le Set Covering Problem (voir partie 3.4.2), nous n'avons pas utilisé de tests de réduction. La méthode employée se décompose donc en trois phases :

- L'initialisation sert à déterminer l'ensemble des variables ainsi que les conflits potentiels entre certaines variables. Les conflits identifiés concernent deux variables correspondant à des trains différents. Cette phase diffère donc de celle présentée dans la partie 4.3.3 puisque l'on identifie les variables en conflit (c'est-à-dire ne pouvant être sélectionnées dans la même solution) et non pas celles qui sont compatibles. Cette phase est très importante car elle conditionne les phases suivantes. Les conflits doivent donc être calculés avec des données précises. Le logiciel utilisé lors de cette phase est une adaptation du simulateur SISYFE¹ [Rodriguez et al.98] qui simule l'ensemble des choix de parcours possibles pour les trains deux à deux afin de déterminer les retards éventuels engendrés par un conflit entre eux. La valeur de ce retard n'a pas d'importance pour nous étant donné que nous souhaitons faire passer les trains dans les nœuds sans les retarder.
- Le problème est ensuite formulé comme un Set Packing Problem dans un autre fichier (format matrice creuse). Pour cela un logiciel de conversion a été mis au point. Celui-ci identifie ainsi les variables du problème et remplit la matrice avec les différents conflits potentiels. La formulation algébrique retenue étant celle d'un Set Packing, une contrainte modélise l'ensemble des conflits possibles pour une variable et non pas seulement les conflits entre variables deux à deux.
- La résolution est effectuée à l'aide de la méthode approchée GRASP (voir partie 4.4).

¹SISYFE est un simulateur de marche de train «très fin». Il permet de reproduire fidèlement le déplacement des trains (avec toutes les caractéristiques réelles d'un convoi) en intégrant tous les éléments de l'infrastructure. Il est capable de fournir un compte-rendu très fin des paramètres d'un train sur l'infrastructure.

5.2.3 Cas de figure traités

Afin de valider le modèle choisi et de vérifier l'aptitude de GRASP à résoudre ce type de problème, nous avons testé 4 situations basées sur les itinéraires classiques des trains [Rodriguez98]. Rien ne garantit cependant qu'il existe une solution pour faire passer tous les trains demandés sans retard. La première situation (voir Tableau 5.1) correspond à un cas réel. L'objectif est de faire passer 6 trains arrivant dans le nœud dans une fenêtre temporelle d'environ 6 minutes. Ces trains sont de différents types et ont des origines et des destinations variées. La principale difficulté de ce problème se situe au niveau des quatre derniers trains arrivant dans un intervalle inférieur à 1 min 30.

Type de train	Heure d'arrivée dans le nœud	Origine	Destination
Classique	0 s	Paris	Chantilly
TGV	+ 172 s	Paris	Lille
Marchandise	+ 270 s	Grande ceinture	Chantilly
Classique	+ 313 s	Chantilly	Paris
TGV	+ 336 s	Lille	Paris
TGV	+ 353 s	Paris	Lille

TAB. 5.1 – Premier cas à 6 trains

L'instance générée pour cette situation a les caractéristiques numériques suivantes :

- 34 variables
- 53 contraintes
- Densité = 12 % (Max-Uns = 8)

La deuxième situation (voir Tableau 5.2) correspond au même cas réel mais en considérant une fenêtre temporelle plus large (près de 13 minutes). Le nombre de trains est donc supérieur puisque deux trains supplémentaires arrivent dans le nœud. Là encore, la principale difficulté se situe dans les quatre même trains. De plus, les nouveaux trains peuvent aussi perturber les trains allant en sens inverse.

Type de train	Heure d'arrivée dans le nœud	Origine	Destination
Classique	0 s	Paris	Chantilly
TGV	+ 172 s	Paris	Lille
Marchandise	+ 270 s	Grande ceinture	Chantilly
Classique	+ 313 s	Chantilly	Paris
TGV	+ 336 s	Lille	Paris
TGV	+ 353 s	Paris	Lille
Marchandise	+ 611 s	Chantilly	Grande ceinture
TGV	+ 769 s	Lille	Paris

TAB. 5.2 – Premier cas à 8 trains

L'instance générée pour cette situation a les caractéristiques numériques suivantes :

- 42 variables
- 58 contraintes
- Densité = 9,6 % (Max-Uns = 8)

On voit bien que l'augmentation du nombre de trains considérés provoque une augmentation du nombre de variables de l'instance. Par contre la faible augmentation du nombre de contraintes ainsi que la diminution de leur densité semblent montrer que les nouveaux trains ne provoquent que peu de conflits.

La troisième situation (voir Tableau 5.3) apporte une légère modification à la première situation en avançant de 30 secondes le deuxième train classique (à destination de Paris) afin d'évaluer l'impact d'un changement des horaires. A part ce décalage, les autres caractéristiques du problème n'ont pas été changées.

Type de train	Heure d'arrivée dans le nœud	Origine	Destination
Classique	0 s	Paris	Chantilly
TGV	+ 172 s	Paris	Lille
Marchandise	+ 270 s	Grande ceinture	Chantilly
Classique	+ 283 s	Chantilly	Paris
TGV	+ 336 s	Lille	Paris
TGV	+ 353 s	Paris	Lille

TAB. 5.3 – Deuxième cas à 6 trains

L'instance générée pour cette situation à les caractéristiques numériques suivantes :

- 34 variables
- 49 contraintes
- Densité = 12,1 % (Max-Uns = 8)

Le nombre de variables reste bien évidemment le même que pour la première situation. Par contre le nombre de contraintes diminue légèrement sans que la densité n'évolue de manière significative. De part ce changement, on peut s'attendre à être confronté à une situation de passage plus simple à réaliser.

Enfin, la dernière situation (voir Tableau 5.4) reprend la précédente mais en augmentant la saturation du nœud. Pour cela, deux trains supplémentaires sont ajoutés (un TGV et un train classique) sans que la fenêtre temporelle soit modifiée (toujours environ 6 minutes). Ainsi, à la précédente difficulté liée aux quatre derniers trains, vient s'en ajouter une nouvelle avec trois autres trains arrivant dans un intervalle de 49 secondes.

L'instance générée pour cette situation à les caractéristiques numériques suivantes :

- 44 variables
- 110 contraintes
- Densité = 9,2 % (Max-Uns = 8)

Comme pour la deuxième situation, l'augmentation du nombre de trains provoque une augmentation du nombre de variables. L'évolution du nombre de contraintes est cependant beaucoup plus significative (même si leur densité diminue un peu) et montre que le nombre

Type de train	Heure d'arrivée dans le nœud	Origine	Destination
Classique	0 s	Paris	Chantilly
TGV	+ 155 s	Lille	Paris
TGV	+ 172 s	Paris	Lille
Classique	+ 204 s	Chantilly	Paris
Marchandise	+ 270 s	Grande ceinture	Chantilly
Classique	+ 313 s	Chantilly	Paris
TGV	+ 336 s	Lille	Paris
TGV	+ 353 s	Paris	Lille

TAB. 5.4 – Deuxième cas à 8 trains

de conflits générés par ces deux nouveaux trains est important.

Les caractéristiques numériques des quatre instances testées sont donc assez proches. De plus, leur dimension reste assez modeste.

5.3 Résultats observés avec GRASP

Ces quatre instances ont ensuite été résolues à l'aide de l'algorithme de GRASP adapté au Set Packing Problem. Les résultats observés ont été rapportés dans le Tableau 5.5.

Situation	Temps de résolution	Nombre de solutions	Nombre de trains
1	3,3 s	3	5 sur 6
2	4,0 s	3	7 sur 8
3	3,4 s	3	6 sur 6
4	5,8 s	2	7 sur 8

TAB. 5.5 – Récapitulatif des résultats observés avec GRASP

On peut tout d'abord constater que les temps de résolution nécessaires pour résoudre ces instances sont très courts (de l'ordre de quelques secondes). Cela devrait permettre de considérer des situations faisant intervenir un plus grand nombre de trains, tout en conservant des temps de réponses raisonnables.

Les différentes solutions obtenues pour la situation 1 montrent que GRASP n'arrive pas à faire passer le train classique Chantilly-Paris et le TGV Lille-Paris en même temps. Il semble en effet difficile de faire circuler ces deux trains arrivant à des heures proches et ayant une même destination sans occasionner un retard pour l'un ou l'autre. Le même problème peut être constaté dans la situation 2 où GRASP parvient tout de même à faire circuler les deux trains supplémentaires.

Ainsi, on voit bien grâce à la situation 3 qu'un léger décalage de l'un des deux trains en conflits permet de trouver une solution pour faire circuler tous les trains sans retard. Les solutions trouvées par GRASP correspondent au passage des six trains de la situation, et sont donc des solutions optimales pour cette instance.

Enfin, la situation 4 permet de constater que le taux de saturation maximum n'est pas atteint dans les premières situations étant donné qu'il est possible de faire passer au moins un train supplémentaire (en l'occurrence le train classique). Le fait que le TGV ne puisse pas passer peut être dû à une heure d'arrivée trop proche des autres trains. Le fait que GRASP ne trouve que deux solutions dans ce cas montre toutefois que le taux de saturation doit être important.

GRASP permet donc d'obtenir rapidement de bonnes solutions. Comme il semble être peu sensible à la difficulté de la situation, on peut s'attendre à pouvoir augmenter la taille des problèmes à résoudre (fenêtre temporelle considérée, nombre de trains, nombre de parcours possibles, présence d'une gare...). Nous disposons ainsi d'une première validation d'une résolution approchée par GRASP sur une infrastructure réelle. En outre, une première confrontation des solutions obtenues auprès d'experts du domaine confirme leur validité dans le contexte réel. Ce premier retour d'expérience permettra aussi d'affiner les paramètres de notre algorithme.

De plus, ces résultats permettent d'envisager dans un avenir proche plusieurs utilisations de notre algorithme de résolution approchée :

- évaluation complète sur des problèmes de grande taille
- confrontation avec des méthodes de résolution exactes
- études de saturation

Conclusion et perspectives

Le travail de recherche réalisé au cours de ce stage a donc permis de valider l'aptitude des modèles issus de l'optimisation combinatoire à représenter des problèmes ferroviaires réels. Une modélisation des nœuds ferroviaires permettant d'évaluer la faisabilité du passage de trains, dont l'horaire a été établi, a ainsi pu être proposée. Elle trouve donc tout son intérêt dans le cadre d'une étude de capacité basée sur une saturation maximale de la grille horaire sur la fenêtre temporelle considérée et donc dans le domaine de la planification ferroviaire.

Dans la première partie, plusieurs méthodes de résolution (exactes ou approchées) ont pu être testées sur un problème d'optimisation combinatoire courant proche du problème étudié : le Set Covering Problem. Une comparaison de la qualité des solutions obtenues par GRASP et des temps de résolutions nécessaires a ainsi pu être effectuée avec les résultats obtenus par des algorithmes gloutons et par une méthode exacte de Branch & Bound. De même, la qualité de nos résultats a pu être mesurée vis à vis des résultats rencontrés dans la littérature.

Cette analyse comparative a permis de mettre en évidence les aptitudes de la métaheuristique GRASP. Pour des instances de grande taille, des solutions de bonne qualité ont ainsi été obtenues dans des temps de résolution raisonnables.

Dans la seconde partie, nous avons pu montrer que le problème de la faisabilité du passage des trains dans un nœud ferroviaire pouvait être modélisé sous la forme d'un Set Packing Problem. La métaheuristique GRASP peut donc être utilisée pour résoudre ce problème.

Une application sur une situation réelle connue pour sa difficulté, celle du nœud de Pierrefitte-Gonesse, a ainsi pu être réalisée grâce aux données issues du simulateur SISYFE. Les résultats obtenus lors de cette expérimentation ont confirmé l'intérêt de la méthodologie employée pour les problèmes de ce type. Toutefois, les instances considérées étaient de taille assez réduite et une résolution exacte de problèmes de cette dimension par des méthodes exactes est encore envisageable.

Ainsi, il serait tout de même intéressant d'évaluer son comportement sur des problèmes de plus grande taille correspondant principalement à des fenêtres temporelles plus étendues. De même, l'étude d'autres nœuds ferroviaires en voie de saturation comme le triangle d'Ostercourt à Douai ou le nœud Lillois est prévue.

Dans ce cadre, une comparaison des résultats obtenus avec GRASP et avec des algorithmes exacts pourrait être réalisée afin de mesurer les limites d'une résolution exacte. Pour

cela, le recours à un code de calculs général comme Cplex et à un algorithme dédié comme le Branch & Cut proposé par les chemins de fer néerlandais est envisagé.

De plus, un premier retour d'expérience permettrait sans doute d'améliorer la valeur des paramètres utilisés pour GRASP afin d'en augmenter l'efficacité.

Enfin, l'expérience acquise grâce aux premières expérimentations devraient permettre d'améliorer le modèle. De même, différentes évolutions du modèle pourraient être envisagées afin de compléter la réponse obtenue par un algorithme exact avec le modèle actuel.

Ainsi, dans les cas où la faisabilité du passage des trains est constatée, l'optimisation multicritère [Visee et al.98] du passage des trains dans le but de fluidifier le trafic pourrait être étudiée. Le choix d'indicateurs de mesure pertinents pour évaluer et comparer les différentes solutions envisageables serait alors très important.

Dans le cas contraire, il faudrait s'intéresser aux actions à entreprendre afin de permettre le passage des trains prévus. Il y a alors deux types de modifications envisageables : celles changeant la grille d'horaire et celles impliquant une modification des infrastructures.

Bibliographie

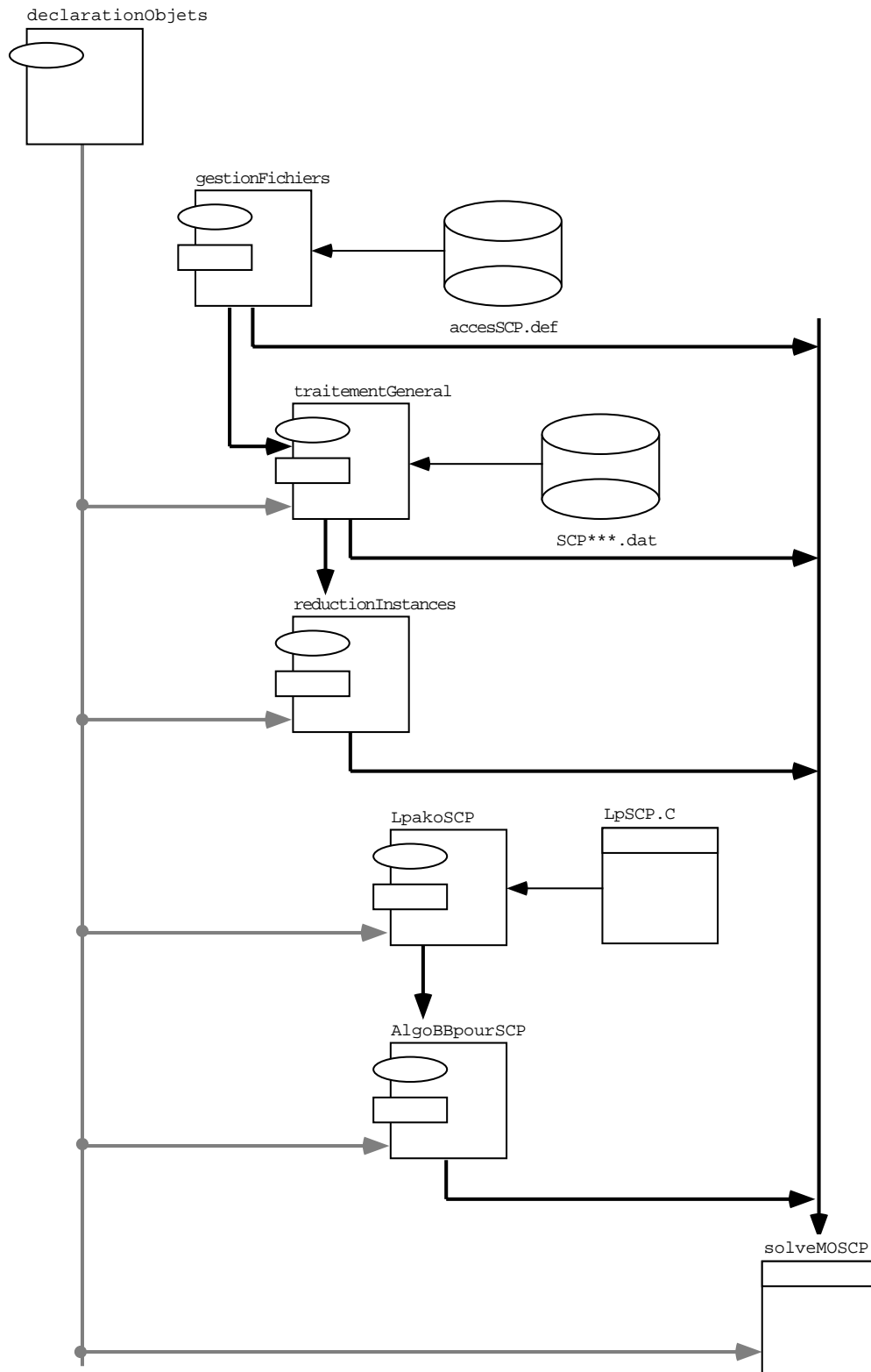
- [Auguet96] Auguet (Benoît). – *Modélisation d'un nœud ferroviaire complexe*. – Rapport de stage de 3^{ème} année, École Centrale de Lille, 1996.
- [Barr et al.95] Barr (Richard S.), Golden (Bruce L.), Kelly (James P.), Resende (Mauricio G.C.) et Stewart (William R.). – Designing and reporting on computational experiments with heuristic methods. *Journal of Heuristics*, vol. 1, 1995, pp. 9–32.
- [Beasley87] Beasley (J.E.). – An algorithm for set covering problem. *European Journal of Operational Research*, vol. 31, 1987, pp. 85–93.
- [Bellaïche97] Bellaïche (H.). – *Recherche sur la saturation des lignes ferroviaires (rapport d'étude de la phase 1)*. – Rapport technique, SYSTRA, 1997.
- [dW90] de Werra (Dominique). – *Éléments de programmation linéaire avec application aux graphes*. – Presses polytechniques romandes, 1990. 306 p.
- [Feo et al.94a] Féo (Thomas A.) et Resende (Mauricio G.C.). – Greedy randomized adaptative search procedures. *Journal of Global Optimization*, vol. 6, 1994, pp. 109–133.
- [Feo et al.94b] Féo (Thomas A.), Resende (Mauricio G.C.) et Smith (Stuart H.). – A greedy randomized adaptative search procedure for maximum independent set. *Operation Research*, vol. 42, 1994, pp. 860–878.
- [Freville00] Fréville (Arnaud). – Méthodes de recherche locale. – 2000. Journée AFPLC - École des Mines de Nantes.
- [Gandibleux et al.98] Gandibleux (Xavier), Vancoppenolle (David) et Tuyttens (Daniel). – A first making use of grasp for solving moco problems. – 1998. 14th MCDM International Conference.
- [Garfinkel et al.72] Garfinkel (Robert S.) et Nemhauser (George L.). – *Integer Programming*. – John Wiley & Sons, 1972. 427 p.
- [Gondran et al.95] Gondran (Michel) et Minoux (Michel). – *Graphes et algorithmes*. – Eyrolles, 1995. 588 p.
- [Grossman et al.97] Grossman (Tal) et Wool (Avishai). – Computational experience with approximation algorithms for the set covering problem. *European Journal of Operational Research*, vol. 101, 1997, pp. 81–92.

- [Hachemane97] Hachemane (Patrick). – *Évaluation de la capacité de réseaux ferroviaires*. – Thèse n° 1632, École Polytechnique Fédérale de Lausanne, 1997.
- [Kroon et al.97] Kroon (Léo G.), Romeijn (H. Edwin) et Zwaneveld (Peter J.). – Routing trains through railway stations : complexity issues. *European Journal of Operational Research*, vol. 98, 1997, pp. 485–498.
- [Lpako] Lpako : large-scale linear programming package (simplex) ver 4.61f.(orlab). <http://orlab.snu.ac.kr/software/index.html>.
- [Lpsolve] Lpsolve. ftp://ftp.ics.ele.tue.nl/pub/projects/lp_solve/.
- [Murray et al.97] Murray (Alan T.) et Church (Richard L.). – Facets for node packing. *European Journal of Operational Research*, vol. 101, 1997, pp. 598–608.
- [Or library] Or-library (j.e. beasley). <http://mscmga.ms.ic.ac.uk/info.html>.
- [Rodriguez et al.98] Rodriguez (Joaquin) et Kermad (L.). – Constraint programming for real-time train circulation management problems in railway nodes. *Advances in Transport*, vol. 2, 1998, pp. 597–606. – Computers in Railways VI.
- [Rodriguez98] Rodriguez (Joaquin). – *Techniques d'aide à la fluidification des circulations dans un nœud ferroviaire complexe : résultats obtenus et analyses (rapport de la phase 5)*. – Rapport technique, INRETS, décembre 1998.
- [Sakarovitch84] Sakarovitch (Michel). – *Optimisation combinatoire : méthodes mathématiques et algorithmiques*. – Hermann, 1984. 270 p.
- [Schneider97] Schneider (F.). – *Recherche sur la saturation des lignes ferroviaires (rapport d'étude de la phase 2)*. – Rapport technique, SYSTRA, 1997.
- [Teghem96] Teghem (Jacques). – *Programmation linéaire*. – Ellipses-Marketing, 1996. 374 p.
- [Vancoppenolle98] Vancoppenolle (David). – *Résolution par GRASP de problèmes d'optimisation combinatoire*. – Mémoire de 2^{ème} licence en informatique, Université de Mons-Hainaut, 1998.
- [Visee et al.98] Visée (M.), Teghem (J.), Pirlot (M.) et Ulungu (E.L.). – Two-phases method and branch and bound procedures to solve the bi-objective knapsack problem. *Journal of Global Optimization*, vol. 12, 1998, pp. 139–155.
- [Zwaneveld et al.96] Zwaneveld (Peter J.), Kroon (Léo G.), Romeijn (H. Edwin), Salomon (Marc), Dautère-Pérès (Stéphane), Van Hoesel (Stan P.M.) et Ambergen (Harrie W.). – Routing trains through railway stations : Model formulation and algorithms. *Transportation Science*, vol. 30, août 1996.
- [Zwaneveld et al.97] Zwaneveld (Peter J.), Kroon (Léo G.) et Van Hoesel (Stan P.M.). – *Routing trains through railway a station based on a Node Packing model*. – Research Memoranda n030, Maastricht Research School of Economics of Technology and Organization, 1997.

Annexes

Annexe I	
Architecture de l'algorithme Branch & Bound	46
Annexe II	
Exemple didactique du fonctionnement de l'algorithme de Branch & Bound	47
Annexe III	
Architecture de l'algorithme de GRASP pour le Set Covering Problem	50
Annexe IV	
Exemple didactique du fonctionnement de GRASP pour le Set Covering Problem	51
Annexe V	
Résultats des expérimentations sur le Set Covering Problem	53
Annexe VI	
Glossaire des principaux termes ferroviaires employés	59
Annexe VII	
Plan de voie du nœud de Pierrefitte-Gonesse	61

Annexe I : Architecture du Branch & Bound



Annexe II : Exemple du Branch & Bound

Phase de réduction du problème

C =	<table style="border-collapse: collapse; width: 100%; text-align: center;"> <tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td></tr> <tr><td>10</td><td>5</td><td>8</td><td>6</td><td>9</td><td>13</td><td>11</td><td>4</td><td>6</td></tr> </table>	1	2	3	4	5	6	7	8	9	10	5	8	6	9	13	11	4	6																																													
1	2	3	4	5	6	7	8	9																																																								
10	5	8	6	9	13	11	4	6																																																								
T =	<table style="border-collapse: collapse; width: 100%; text-align: center;"> <tr><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td></tr> <tr><td>2</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td></tr> <tr><td>3</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td></tr> <tr><td>4</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>5</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>6</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td></tr> <tr><td>7</td><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td></tr> </table>	1	1	1	0	1	0	1	1	0	2	0	1	0	0	0	0	1	0	3	0	1	0	1	1	0	1	1	4	0	0	1	0	0	0	0	0	5	1	0	1	0	1	1	0	1	6	0	1	1	0	0	0	1	1	7	1	0	0	1	1	0	1	1
1	1	1	0	1	0	1	1	0																																																								
2	0	1	0	0	0	0	1	0																																																								
3	0	1	0	1	1	0	1	1																																																								
4	0	0	1	0	0	0	0	0																																																								
5	1	0	1	0	1	1	0	1																																																								
6	0	1	1	0	0	0	1	1																																																								
7	1	0	0	1	1	0	1	1																																																								

$\tau_4 = e_4 \Rightarrow X_4 = 1$, suppression de τ_4 et de t_4

$T_{74} = 1 \Rightarrow$ suppression de τ_7

C =	<table style="border-collapse: collapse; width: 100%; text-align: center;"> <tr><td>1</td><td>2</td><td>3</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td></tr> <tr><td>10</td><td>5</td><td>8</td><td>9</td><td>13</td><td>11</td><td>4</td><td>6</td></tr> </table>	1	2	3	5	6	7	8	9	10	5	8	9	13	11	4	6																								
1	2	3	5	6	7	8	9																																		
10	5	8	9	13	11	4	6																																		
T =	<table style="border-collapse: collapse; width: 100%; text-align: center;"> <tr><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td></tr> <tr><td>2</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td></tr> <tr><td>3</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>5</td><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td></tr> <tr><td>6</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td></tr> </table>	1	1	1	1	0	1	1	0	2	0	1	1	0	0	1	0	3	0	1	0	1	1	0	1	5	1	0	1	1	1	0	1	6	0	1	1	0	0	1	1
1	1	1	1	0	1	1	0																																		
2	0	1	1	0	0	1	0																																		
3	0	1	0	1	1	0	1																																		
5	1	0	1	1	1	0	1																																		
6	0	1	1	0	0	1	1																																		

$\tau_1 \geq \tau_2 \Rightarrow$ suppression de τ_1

C =	<table style="border-collapse: collapse; width: 100%; text-align: center;"> <tr><td>1</td><td>2</td><td>3</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td></tr> <tr><td>10</td><td>5</td><td>8</td><td>9</td><td>13</td><td>11</td><td>4</td><td>6</td></tr> </table>	1	2	3	5	6	7	8	9	10	5	8	9	13	11	4	6																				
1	2	3	5	6	7	8	9																														
10	5	8	9	13	11	4	6																														
T =	<table style="border-collapse: collapse; width: 100%; text-align: center;"> <tr><td>2</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td></tr> <tr><td>3</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td></tr> <tr><td>5</td><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td></tr> <tr><td>6</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td></tr> </table>	2	0	1	1	0	0	0	1	0	3	0	1	0	1	1	0	1	1	5	1	0	1	1	1	0	0	1	6	0	1	1	0	0	1	0	1
2	0	1	1	0	0	0	1	0																													
3	0	1	0	1	1	0	1	1																													
5	1	0	1	1	1	0	0	1																													
6	0	1	1	0	0	1	0	1																													

$t_2 \geq t_7$ et $c_2 < c_7 \Rightarrow$ suppression de t_7

C =	<table style="border-collapse: collapse; width: 100%; text-align: center;"> <tr> <td style="padding: 0 10px;">1</td> <td style="padding: 0 10px;">2</td> <td style="padding: 0 10px;">3</td> <td style="padding: 0 10px;">5</td> <td style="padding: 0 10px;">6</td> <td style="padding: 0 10px;">8</td> <td style="padding: 0 10px;">9</td> </tr> <tr> <td style="padding: 0 10px;">10</td> <td style="padding: 0 10px;">5</td> <td style="padding: 0 10px;">8</td> <td style="padding: 0 10px;">9</td> <td style="padding: 0 10px;">13</td> <td style="padding: 0 10px;">4</td> <td style="padding: 0 10px;">6</td> </tr> </table>	1	2	3	5	6	8	9	10	5	8	9	13	4	6
1	2	3	5	6	8	9									
10	5	8	9	13	4	6									

T =	<table style="border-collapse: collapse; width: 100%; text-align: center;"> <tr> <td style="padding: 0 10px;">2</td> <td style="padding: 0 10px;">0</td> <td style="padding: 0 10px;">1</td> <td style="padding: 0 10px;">1</td> <td style="padding: 0 10px;">0</td> <td style="padding: 0 10px;">0</td> <td style="padding: 0 10px;">1</td> <td style="padding: 0 10px;">0</td> </tr> <tr> <td style="padding: 0 10px;">3</td> <td style="padding: 0 10px;">0</td> <td style="padding: 0 10px;">1</td> <td style="padding: 0 10px;">0</td> <td style="padding: 0 10px;">1</td> <td style="padding: 0 10px;">1</td> <td style="padding: 0 10px;">1</td> <td style="padding: 0 10px;">1</td> </tr> <tr> <td style="padding: 0 10px;">5</td> <td style="padding: 0 10px;">1</td> <td style="padding: 0 10px;">0</td> <td style="padding: 0 10px;">1</td> <td style="padding: 0 10px;">1</td> <td style="padding: 0 10px;">1</td> <td style="padding: 0 10px;">0</td> <td style="padding: 0 10px;">1</td> </tr> <tr> <td style="padding: 0 10px;">6</td> <td style="padding: 0 10px;">0</td> <td style="padding: 0 10px;">1</td> <td style="padding: 0 10px;">1</td> <td style="padding: 0 10px;">0</td> <td style="padding: 0 10px;">0</td> <td style="padding: 0 10px;">0</td> <td style="padding: 0 10px;">1</td> </tr> </table>	2	0	1	1	0	0	1	0	3	0	1	0	1	1	1	1	5	1	0	1	1	1	0	1	6	0	1	1	0	0	0	1
2	0	1	1	0	0	1	0																										
3	0	1	0	1	1	1	1																										
5	1	0	1	1	1	0	1																										
6	0	1	1	0	0	0	1																										

$t3 \geq t1$ et $c3 < c1 \Rightarrow$ suppression de t1

C =	<table style="border-collapse: collapse; width: 100%; text-align: center;"> <tr> <td style="padding: 0 10px;">2</td> <td style="padding: 0 10px;">3</td> <td style="padding: 0 10px;">5</td> <td style="padding: 0 10px;">6</td> <td style="padding: 0 10px;">8</td> <td style="padding: 0 10px;">9</td> </tr> <tr> <td style="padding: 0 10px;">5</td> <td style="padding: 0 10px;">8</td> <td style="padding: 0 10px;">9</td> <td style="padding: 0 10px;">13</td> <td style="padding: 0 10px;">4</td> <td style="padding: 0 10px;">6</td> </tr> </table>	2	3	5	6	8	9	5	8	9	13	4	6
2	3	5	6	8	9								
5	8	9	13	4	6								

T =	<table style="border-collapse: collapse; width: 100%; text-align: center;"> <tr> <td style="padding: 0 10px;">2</td> <td style="padding: 0 10px;">1</td> <td style="padding: 0 10px;">1</td> <td style="padding: 0 10px;">0</td> <td style="padding: 0 10px;">0</td> <td style="padding: 0 10px;">1</td> <td style="padding: 0 10px;">0</td> </tr> <tr> <td style="padding: 0 10px;">3</td> <td style="padding: 0 10px;">1</td> <td style="padding: 0 10px;">0</td> <td style="padding: 0 10px;">1</td> <td style="padding: 0 10px;">1</td> <td style="padding: 0 10px;">1</td> <td style="padding: 0 10px;">1</td> </tr> <tr> <td style="padding: 0 10px;">5</td> <td style="padding: 0 10px;">0</td> <td style="padding: 0 10px;">1</td> <td style="padding: 0 10px;">1</td> <td style="padding: 0 10px;">1</td> <td style="padding: 0 10px;">0</td> <td style="padding: 0 10px;">1</td> </tr> <tr> <td style="padding: 0 10px;">6</td> <td style="padding: 0 10px;">1</td> <td style="padding: 0 10px;">1</td> <td style="padding: 0 10px;">0</td> <td style="padding: 0 10px;">0</td> <td style="padding: 0 10px;">0</td> <td style="padding: 0 10px;">1</td> </tr> </table>	2	1	1	0	0	1	0	3	1	0	1	1	1	1	5	0	1	1	1	0	1	6	1	1	0	0	0	1
2	1	1	0	0	1	0																							
3	1	0	1	1	1	1																							
5	0	1	1	1	0	1																							
6	1	1	0	0	0	1																							

$t5 \geq t6$ et $c5 < c6 \Rightarrow$ suppression de t6

C =	<table style="border-collapse: collapse; width: 100%; text-align: center;"> <tr> <td style="padding: 0 10px;">2</td> <td style="padding: 0 10px;">3</td> <td style="padding: 0 10px;">5</td> <td style="padding: 0 10px;">8</td> <td style="padding: 0 10px;">9</td> </tr> <tr> <td style="padding: 0 10px;">5</td> <td style="padding: 0 10px;">8</td> <td style="padding: 0 10px;">9</td> <td style="padding: 0 10px;">4</td> <td style="padding: 0 10px;">6</td> </tr> </table>	2	3	5	8	9	5	8	9	4	6
2	3	5	8	9							
5	8	9	4	6							

T =	<table style="border-collapse: collapse; width: 100%; text-align: center;"> <tr> <td style="padding: 0 10px;">2</td> <td style="padding: 0 10px;">1</td> <td style="padding: 0 10px;">1</td> <td style="padding: 0 10px;">0</td> <td style="padding: 0 10px;">1</td> <td style="padding: 0 10px;">0</td> </tr> <tr> <td style="padding: 0 10px;">3</td> <td style="padding: 0 10px;">1</td> <td style="padding: 0 10px;">0</td> <td style="padding: 0 10px;">1</td> <td style="padding: 0 10px;">1</td> <td style="padding: 0 10px;">1</td> </tr> <tr> <td style="padding: 0 10px;">5</td> <td style="padding: 0 10px;">0</td> <td style="padding: 0 10px;">1</td> <td style="padding: 0 10px;">1</td> <td style="padding: 0 10px;">0</td> <td style="padding: 0 10px;">1</td> </tr> <tr> <td style="padding: 0 10px;">6</td> <td style="padding: 0 10px;">1</td> <td style="padding: 0 10px;">1</td> <td style="padding: 0 10px;">0</td> <td style="padding: 0 10px;">0</td> <td style="padding: 0 10px;">1</td> </tr> </table>	2	1	1	0	1	0	3	1	0	1	1	1	5	0	1	1	0	1	6	1	1	0	0	1
2	1	1	0	1	0																				
3	1	0	1	1	1																				
5	0	1	1	0	1																				
6	1	1	0	0	1																				

$t9 \geq t5$ et $c9 < c5 \Rightarrow$ suppression de t5

C =	<table style="border-collapse: collapse; width: 100%; text-align: center;"> <tr> <td style="padding: 0 10px;">2</td> <td style="padding: 0 10px;">3</td> <td style="padding: 0 10px;">8</td> <td style="padding: 0 10px;">9</td> </tr> <tr> <td style="padding: 0 10px;">5</td> <td style="padding: 0 10px;">8</td> <td style="padding: 0 10px;">4</td> <td style="padding: 0 10px;">6</td> </tr> </table>	2	3	8	9	5	8	4	6
2	3	8	9						
5	8	4	6						

T =	<table style="border-collapse: collapse; width: 100%; text-align: center;"> <tr> <td style="padding: 0 10px;">2</td> <td style="padding: 0 10px;">1</td> <td style="padding: 0 10px;">1</td> <td style="padding: 0 10px;">1</td> <td style="padding: 0 10px;">0</td> </tr> <tr> <td style="padding: 0 10px;">3</td> <td style="padding: 0 10px;">1</td> <td style="padding: 0 10px;">0</td> <td style="padding: 0 10px;">1</td> <td style="padding: 0 10px;">1</td> </tr> <tr> <td style="padding: 0 10px;">5</td> <td style="padding: 0 10px;">0</td> <td style="padding: 0 10px;">1</td> <td style="padding: 0 10px;">0</td> <td style="padding: 0 10px;">1</td> </tr> <tr> <td style="padding: 0 10px;">6</td> <td style="padding: 0 10px;">1</td> <td style="padding: 0 10px;">1</td> <td style="padding: 0 10px;">0</td> <td style="padding: 0 10px;">1</td> </tr> </table>	2	1	1	1	0	3	1	0	1	1	5	0	1	0	1	6	1	1	0	1
2	1	1	1	0																	
3	1	0	1	1																	
5	0	1	0	1																	
6	1	1	0	1																	

$\tau6 \geq \tau5 \Rightarrow$ suppression de $\tau6$

$$C = \begin{array}{cccc} & 2 & 3 & 8 & 9 \\ \hline & 5 & 8 & 4 & 6 \end{array}$$

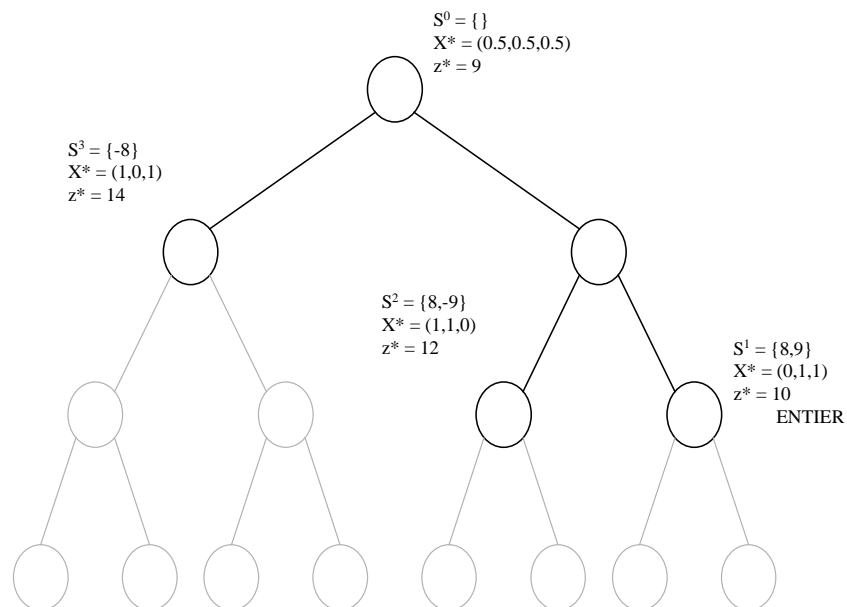
$$T = \begin{array}{cccc} 2 & 1 & 1 & 1 & 0 \\ 3 & 1 & 0 & 1 & 1 \\ 5 & 0 & 1 & 0 & 1 \end{array}$$

$t_8 \geq t_2$ et $c_8 < c_2 \Rightarrow$ suppression de t_2

$$C = \begin{array}{ccc} & 3 & 8 & 9 \\ \hline & 8 & 4 & 6 \end{array}$$

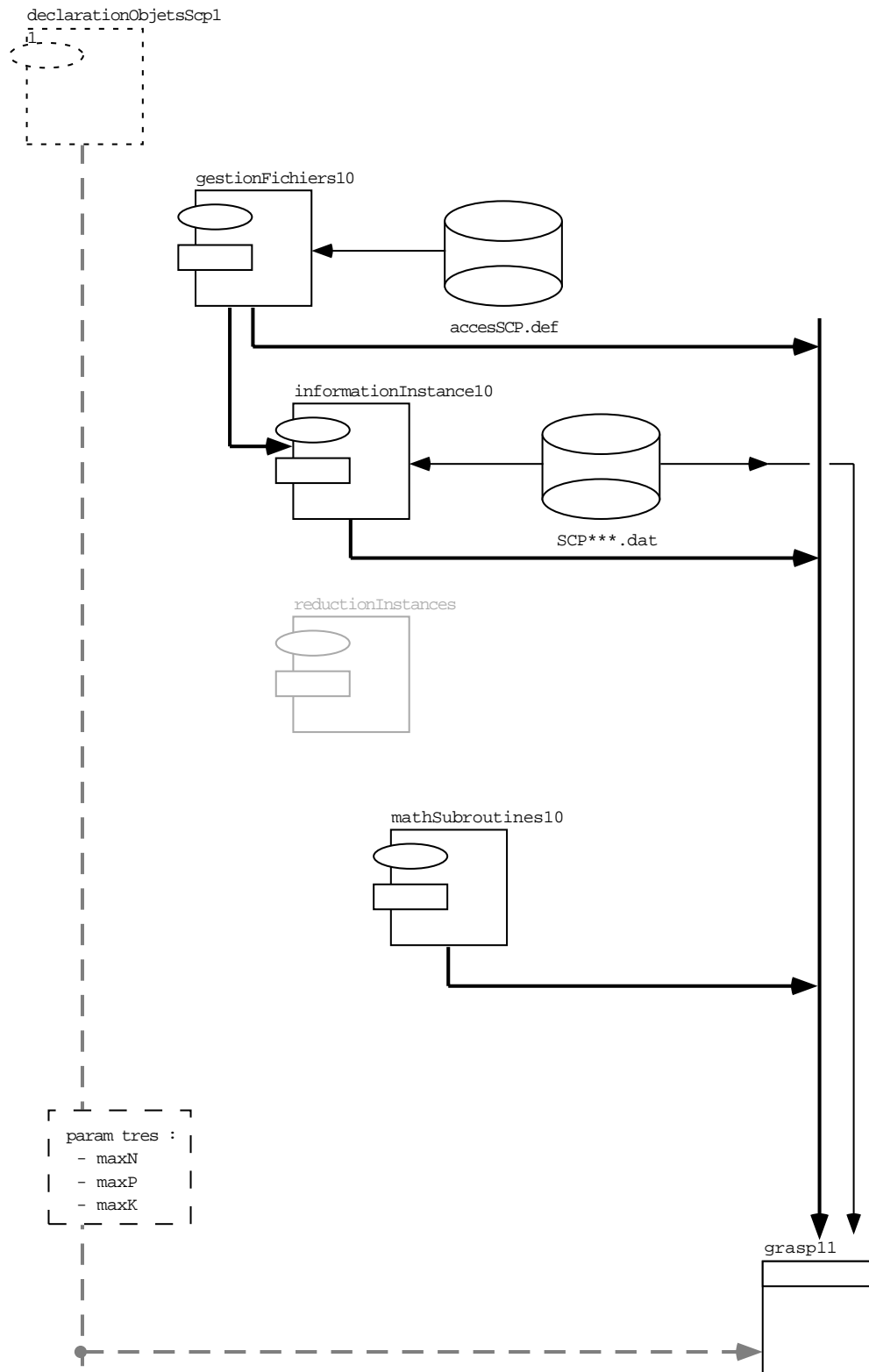
$$T = \begin{array}{ccc} 2 & 1 & 1 & 0 \\ 3 & 0 & 1 & 1 \\ 5 & 1 & 0 & 1 \end{array}$$

Phase de résolution du problème



La solution optimale trouvée a donc un coût de 16 et correspond à la sélection des variables 4, 8 et 9.

Annexe III : Architecture de GRASP



Annexe IV : Exemple de GRASP pour le SCP

Génération d'une solution initiale avec $\alpha = 0.7$

C =	1	2	3	4	5	6	7	8	9
	10	5	8	6	9	13	11	4	6

T =	1	2	3	4	5	6	7	8	9
	1	1	1	0	1	0	1	1	0
	2	0	1	1	0	0	0	1	0
	3	0	1	0	0	1	1	0	1
	4	0	0	0	1	0	0	0	0
	5	1	0	1	0	1	1	0	1
	6	0	1	1	0	0	0	1	1
	7	1	0	0	1	1	0	0	1

Intérêt 0.30 **0.80** 0.50 0.33 0.44 0.15 0.18 **1.00** 0.67

Maximum = 1.00 \Rightarrow on retient ceux dont l'intérêt est supérieur à $0.7 * 1.00 = 0.70$

Choix aléatoire entre X2 et X8 $\Rightarrow X2 = 1$

C =	1	3	4	5	6	7	8	9
	10	8	6	9	13	11	4	6

T =	4	5	7					
	0	0	1	0	0	0	0	0
	1	1	0	1	1	0	0	1
	1	0	1	1	0	0	1	1

Intérêt 0.20 0.13 **0.33** 0.22 0.08 0.00 **0.25** **0.33**

Maximum = 0.33 \Rightarrow on retient ceux dont l'intérêt est supérieur à $0.7 * 0.33 = 0.23$

Choix aléatoire entre X4, X8 et X9 $\Rightarrow X8 = 1$

C =	1	3	4	5	6	7	9
	10	8	6	9	13	11	6

T =	4	5					
	0	0	1	0	0	0	0
	1	1	0	1	1	0	1

Intérêt 0.10 **0.13** **0.17** 0.11 0.08 0.00 **0.17**

Maximum = 0.17 \Rightarrow on retient ceux dont l'intérêt est supérieur à $0.7 * 0.17 = 0.12$

Choix aléatoire entre X3, X4 et X9 $\Rightarrow X4 = 1$

$$C = \begin{array}{cccccc} & 1 & 3 & 5 & 6 & 7 & 9 \\ \hline & 10 & 8 & 9 & 13 & 11 & 6 \end{array}$$

$$T = 5 \begin{array}{cccccc} \hline 1 & 1 & 1 & 1 & 0 & 1 \end{array}$$

$$\text{Intérêt} \quad 0.10 \quad \mathbf{0.13} \quad 0.11 \quad 0.08 \quad 0.00 \quad \mathbf{0.17}$$

Maximum = 0.17 \Rightarrow on retient ceux dont l'intérêt est supérieur à $0.7 * 0.17 = 0.12$

Choix aléatoire entre X3 et X9 $\Rightarrow X9 = 1$

La solution initiale obtenue a un coût de 21 et correspond à la sélection des variables 2, 4, 8 et 9.

Recherche locale (1-0 échanges)

Essai de suppression de X2 de la solution :

X4, X8 et X9 constituent une solution admissible \Rightarrow suppression de X2

Essai de suppression de X4 de la solution :

X8 et X9 ne constituent pas une solution admissible

Essai de suppression de X8 de la solution :

X4 et X9 ne constituent pas une solution admissible

Essai de suppression de X9 de la solution :

X4 et X8 ne constituent pas une solution admissible

La solution finale trouvée a donc un coût de 16 et correspond à la sélection des variables 4, 8 et 9.

Annexe V : Résultats des expérimentations sur le SCP

Abréviations utilisées :

Z	Valeur optimale trouvée
Tps	Temps de résolution (en seconde)
N	Nombre de variables du problème
K	Nombre de contraintes du problème
Coef. C	Coûts unitaires ou pondérés
Nr	Nombre de variables du problème après les tests de réduction
Kr	Nombre de contraintes du problème après les tests de réduction
LP	Valeur optimale de la relaxation linéaire
B&B	Résultats du Branch & Bound

GRASP 2	Nom	Expe1		Expe2		Expe3		Expe4		Expe5		Expe6		Expe7		Expe8		Expe9		Expe10		Valeur			Temps					
		Z	Tps	Z	Tps	Z	Tps	Z	Tps	Z	Tps	Z	Tps	Z	Tps	Z	Tps	Z	Tps	Z	Tps	Z	Tps	Moy	μ	Min	Max	μ	Min	Max
SCP41	433	202,7	433	198	433	204,9	433	217,5	433	200,9	433	186	433	183	433	205,6	433	189,5	433	173	433	0,0	433	433	196,1	13,1	173,0	217,5		
SCP41r	247	34,3	247	34,7	248	33,4	248	33,5	247	38,8	247	33	247	34,2	247	38,3	247	36,6	248	34,1	247	247,3	0,5	247	248	35,1	2,1	33,0	38,8	
SCP41U	41	73,2	41	72,8	42	73,3	41	73,4	42	74,2	42	73,4	42	73,3	42	74	41	73,9	40	73,6	40	73,6	41,4	0,7	40	42	73,5	0,4	72,8	74,2
SCP41Ur	32	14,6	31	14,6	32	14,8	31	14,7	32	14,7	32	14,7	32	14,7	32	14,7	32	14,7	31	14,7	32	14,8	31,7	0,5	31	32	14,7	0,1	14,6	14,8
SCP61	142	103,8	142	96,8	142	93,7	142	91,2	142	104,7	142	92	143	90,2	142	99,1	142	96	142	95,6	142	95,6	142,1	0,3	142	143	96,3	5,0	90,2	104,7
SCP61r	83	15,3	83	15,3	83	15,3	83	15,2	89	15,3	83	15,2	83	15,3	83	15,3	83	15,6	83	15,4	83	15,4	84,0	2,2	83	89	15,3	0,1	15,2	15,6
SCP61U	22	41,5	22	41,4	22	40,4	22	40,6	22	41,1	21	40,6	22	40,7	22	40,8	22	40,8	22	40,6	22	41,1	21,9	0,3	21	22	40,9	0,4	40,4	41,5
SCP61Ur	17	8,6	17	8,7	17	8,6	17	8,5	17	8,5	17	8,6	17	8,6	17	8,6	17	8,6	16	8,6	16	8,7	16,8	0,4	16	17	8,6	0,1	8,5	8,7
SCPAl	258	1106	256	1097	256	1242	257	1101	259	1101	258	1096	259	1042	258	1160	258	1143	258	1157	258	1157	257,7	1,1	256	259	1124,2	54,1	1041,6	1241,6
SCPAl1r	91	16,9	91	15,5	91	15,5	91	15,6	91	15,7	91	16,8	91	15,8	91	15,6	91	15,6	91	15,6	91	15,6	91,0	0,0	91	91	15,9	0,5	15,5	16,9
SCPAl12	133	69,5	133	69,4	133	69,6	133	69,4	134	69,5	133	69,5	133	69,6	134	69,5	133	69,7	133	69,4	133	69,4	133,2	0,4	133	134	69,5	0,1	69,4	69,7
SCPAlU	42	49,5	42	49,5	42	49,6	42	49,4	42	49,2	42	49,6	42	49,6	41	49,4	42	49,8	42	49,3	42	49,6	41,9	0,3	41	42	49,5	2,0	49,2	49,8
SCPAlUr	18	6,7	18	6,8	19	6,8	19	6,7	19	6,7	18	6,7	18	6,7	19	6,8	19	6,8	19	6,8	19	6,8	18,6	0,5	18	19	6,8	0,1	6,7	6,8
SCPAlUr2	26	34,3	25	34,4	26	34,5	26	34,5	26	34,1	26	34,3	26	34,8	26	34,3	25	33,9	26	34,3	26	34,3	25,7	0,5	25	26	34,3	0,2	33,9	34,8
SCPAlU2	5	2,9	5	2,9	5	2,9	5	2,9	5	2,9	5	2,9	5	2,9	5	2,9	5	2,9	5	2,9	5	2,9	5,0	0,0	5	5	2,9	0,0	2,9	2,9
SCPAlU2r	3	1,1	4	1,2	3	1,2	3	1,2	3	1,1	3	1,2	3	1,2	3	1,2	3	1,2	3	1,2	3	1,2	3,1	0,3	3	4	1,2	0,0	1,1	1,2
SCPAlU2r	3	1	3	1	3	1	3	1	3	1	3	1	3	1	3	1	3	1	3	1	3	1	3,0	0,0	3	3	1,0	0,0	1,0	1,0
SCPAlU2r	29	1003	29	982	29	995,5	29	1009	29	1003	29	977,2	29	1004	29	1003	29	996,2	30	999,5	30	997,1	29,1	0,3	29	30	997,1	10,1	977,2	1008,5
SCPAlU2r	7	1,3	7	1,3	7	1,3	7	1,3	7	1,3	7	1,3	7	1,3	7	1,3	7	1,3	7	1,3	7	1,3	7,0	0,0	7	7	1,3	0,0	1,3	1,3
SCPAlU2r	4	1	4	1	4	1	4	1	4	1	4	1	4	1	4	1	4	1	4	1	4	1	4,0	0,0	4	4	1,0	0,0	1,0	1,0
SCPAlU2r	19	646,8	19	647	18	643,8	19	648,6	19	647,3	18	642,4	19	645	18	643,6	19	645,6	18	644,1	19	645,4	18,6	0,5	18	19	645,4	1,9	642,4	648,6
SCPAlU2r	5	1,2	5	1,2	5	1,2	5	1,2	5	1,2	5	1,2	5	1,2	5	1,2	5	1,2	5	1,2	5	1,2	5,0	0,0	5	5	1,2	0,0	1,2	1,2
SCPAlU2r	3	1	3	1	3	1	3	1	3	1	3	1	3	1	3	1	3	1	3	1	3	1	3,0	0,0	3	3	1,0	0,0	1,0	1,0
SCPAlU2r	64	16,1	64	16,1	63	16	64	16	64	16,1	63	16	63	16	63	16	64	16,1	65	16,1	66	16,2	64,0	0,9	63	66	16,1	0,1	15,9	16,2
SCPAlU2r	29	19	30	19	29	19	29	18,9	30	19,1	29	18,8	30	19,2	30	19	30	19,2	30	19,2	30	19	29,6	0,5	29	30	19,0	0,1	18,8	19,2

GRASP 3	Nom	Expe1		Expe2		Expe3		Expe4		Expe5		Expe6		Expe7		Expe8		Expe9		Expe10		Valeur				Temps				
		Z	Tps	Z	Tps	Z	Tps	Z	Tps	Z	Tps	Z	Tps	Z	Tps	Z	Tps	Z	Tps	Z	Tps	Z	Tps	Moy	μ	Min	Max	μ	Min	Max
SCP41	433	353,6	433	362	433	334,2	433	347,4	433	383,9	433	366,7	433	379	433	344,1	433	362,5	433	363,2	433	433,0	0,0	433	433	359,7	17,4	334,2	393,9	
SCP41r	244	81,6	244	86,1	244	80,2	244	84,4	244	82,7	244	86,4	247	88	247	95,9	247	86,4	244	86,5	244	244,9	1,4	244	247	85,8	4,3	80,2	95,9	
SCP41U	41	876,5	41	761	41	857,2	41	788,5	41	896,2	42	829	40	826	41	911,4	41	918,2	41	826,8	41	826,8	40,9	0,6	40	42	849,1	52,3	760,7	918,2
SCP41Ur	32	104,1	32	103	32	117	31	116,9	31	124,3	32	126,3	31	124	31	132,2	30	135,6	32	143	32	31,4	0,7	30	32	122,6	12,8	103,0	143,0	
SCP61	138	136,5	141	130	138	130,3	138	119	138	137,8	138	132,5	138	128	141	125,9	141	125,4	138	139,8	138,9	138,9	138,9	1,4	138	141	130,5	6,4	119,0	139,8
SCP61r	86	25,2	83	24,3	83	23,6	87	24,2	83	23,3	83	24,1	83	24	83	23,6	83	23,4	83	23	83,7	1,5	83	87	23,9	0,6	23,0	25,2		
SCP61U	22	236,9	22	235	22	237,4	22	265	22	237,6	22	241,7	22	241	22	246,8	22	260	22	272,5	22,0	0,0	22	22	246,7	12,1	235,4	272,5		
SCP61Ur	17	36,7	17	36,7	17	37,4	17	36,9	17	34,4	17	35,7	17	38	17	35,6	17	36,9	17	36,1	17,0	0,0	17	17	36,8	1,4	34,4	38,9		
SCPAl	256	1597	257	1481	258	1490	256	1710	257	1634	257	1505	257	1542	256	1473	257	1548	255	1505	256,6	0,8	255	258	1548,5	76,9	1473,0	1709,9		
SCPAl1	89	31	89	32	89	35,3	89	34,9	89	34,8	89	33,8	89	34,5	89	34,9	89	34,4	89	35,7	89,0	0,0	89	89	34,1	1,5	31,0	35,7		
SCPAl12	132	144,4	132	153	132	146	132	149,7	132	145	131	149,2	131	144	132	146,1	132	142,2	132	141,8	131,8	0,4	131	132	146,2	3,6	141,8	153,4		
SCPAlU	40	364,6	42	352,5	40	372,4	42	372,3	42	378,5	42	354,2	42	427,0	42	352,8	43	357,3	42	355,1	41,7	0,9	40	43	368,6	225,5	352,5	426,9		
SCPAlU1	18	20,9	18	20	18	20,1	18	19,3	18	20,6	18	23,2	18	20,6	18	20,1	18	21,7	18	19,2	18,0	0,0	18	18	20,6	1,2	19,2	23,2		
SCPAlU2	25	155,1	26	146	25	142,9	25	161,7	25	159,8	25	173,9	26	145	25	150,4	26	159,8	26	147,4	25,4	0,5	25	26	154,3	9,7	142,9	173,9		
SCPAlU2	5	3,7	5	3,8	5	3,6	5	3,7	5	3,7	5	3,5	5	3,7	5	3,7	5	3,7	5	3,9	5,0	0,0	5	5	3,7	0,1	3,5	3,9		
SCPAlU2	3	1,3	3	1,2	4	1,2	3	1,2	4	1,2	3	1,2	3	1,2	3	1,2	4	1,2	4	1,2	3,4	0,5	3	4	1,2	0,0	1,2	1,3		
SCPAlU2	3	1	3	1	3	1	3	1	3	1	3	1	3	1	3	1	3	1	3	1	3,0	0,0	3	3	1,0	0,0	1,0	1,0		
SCPAlU2	29	1011	29	997	29	1000	29	1003	29	1000	29	1001	29	1001	29	1017	29	1003	29	982,8	29,0	0,0	29	29	1001,7	9,0	982,8	1017,3		
SCPAlU2	7	1,6	7	1,4	7	1,4	7	1,4	7	1,4	7	1,4	7	1,4	7	1,4	7	1,4	7	1,4	7,0	0,0	7	7	1,4	0,1	1,4	1,6		
SCPAlU2	4	1	4	1	4	1	4	1	4	1	4	1	4	1	4	1	4	1	4	1	4,0	0,0	4	4	1,0	0,0	1,0	1,0		
SCPAlU2	19	2386	18	2360	18	2340	18	2406	18	2366	18	2385	18	2295	19	2388	18	2296	19	2389	18,3	0,5	18	19	2361,1	39,1	2295,1	2406,0		
SCPAlU2	5	1,6	5	1,3	5	1,3	5	1,3	5	1,3	5	1,3	5	1,3	5	1,3	5	1,3	5	1,3	5,0	0,0	5	5	1,3	0,1	1,3	1,6		
SCPAlU2	3	1	3	1	3	1	3	1	3	1	3	1	3	1	3	1	3	1	3	1	3,0	0,0	3	3	1,0	0,0	1,0	1,0		
SCPAlU2	62	483,5	63	523	63	446,5	63	507,2	63	351,2	62	451,5	63	430	63	496,6	63	380,7	64	364,9	62,9	0,6	62	64	443,5	61,3	351,2	523,4		
SCPAlU2	28	313,3	27	324	29	295,8	28	276,2	28	325,1	28	316,3	26	314	28	312	28	268,1	27	328,8	27,7	0,8	26	29	307,4	20,8	268,1	328,8		

GRASP 5	Expe1		Expe2		Expe3		Expe4		Expe5		Expe6		Expe7		Expe8		Expe9		Expe10		Valeur			Temps						
	Z	Tps	Z	Tps	Z	Tps	Z	Tps	Z	Tps	Z	Tps	Z	Tps	Z	Tps	Z	Tps	Z	Tps	Z	Tps	Moy	μ	Min	Max	μ	Min	Max	
SCP41	433	241,5	433	241,5	433	239,6	433	239,4	433	239,6	433	239,4	433	241,7	433	239,5	433	239,6	433	240,2	433	240,2	433,0	0,0	433	433	240,7	1,2	239,4	242,7
SCP41r	247	44,7	247	44,7	247	42,7	247	42,6	247	42,7	247	42,9	247	44,7	244	45,1	247	44,6	247	44,8	247	44,8	246,4	1,3	244	247	44,0	1,1	42,6	45,2
SCP41U	41	314	40	280	41	249,4	41	249,2	41	182,7	41	313,1	41	314	41	313,3	41	313,6	41	182,5	41	182,5	40,9	0,3	40	41	271,1	53,4	182,5	314,0
SCP41Ur	30	50,1	31	50,8	31	33,4	31	42	32	52,5	31	33,3	32	52,3	31	58,3	31	33,4	30	32,8	30	32,8	31,0	0,7	30	32	43,9	10,0	32,8	58,3
SCP61	141	137,3	141	130	141	137,4	141	138,2	141	136,9	141	136,8	141	136	141	136,2	141	137,2	141	137,1	141	137,1	141,0	0,0	141	141	136,4	2,2	130,2	138,2
SCP61r	83	24,5	83	24,7	83	24,6	86	24,9	83	24,9	83	24,7	83	24,6	83	24,6	83	24,9	83	24,7	83	24,7	83,3	0,9	83	86	24,7	0,1	24,5	24,9
SCP61U	22	125,4	22	126	22	125,4	21	83	21	83	22	124,7	22	126	22	125,2	22	125,1	22	125,2	22	125,2	21,8	0,4	21	22	116,9	17,8	83,0	125,9
SCP61Ur	17	22,9	16	24,6	17	23	16	16,9	17	22,8	17	23	17	22,8	17	22,8	16	16,8	17	22,9	17	22,9	16,7	0,5	16	17	21,9	2,7	16,8	24,6
SCP61U1	258	1158	256	1249	258	1153	256	1157	257	1146	258	1248	256	1348	259	1349	258	1331	258	1159	257,4	1,1	256	259	1229,7	86,6	1145,9	1348,7		
SCP61U1r	89	17,7	89	17,7	89	17,7	89	17,7	89	17,7	89	17,7	89	17,7	89	17,6	89	11,7	89	17,7	89	17,7	89,0	0,0	89	89	17,1	1,9	11,7	17,7
SCP61U1U	132	80	132	100	132	100,6	132	100,5	132	90,1	132	90,2	132	90,1	132	90	132	90,1	132	79,9	132,0	0,0	132	132	91,2	7,6	79,9	100,6		
SCP61U1U1	41	1462	41	1150	41	1462	41	1459	41	838,6	41	1460	41	1149	41	840,3	41	1458	40	826,6	40,9	0,3	40	41	1210,5	287,1	826,6	1462,3		
SCP61U1U1r	18	14,6	19	13,1	19	13	18	9,7	18	9,6	19	13	18	9,6	19	13	18	12,5	18	14,3	18,4	0,5	18	19	12,2	1,9	9,6	14,6		
SCP61U1U1U	25	61,8	25	62	26	76,8	25	50,2	25	62	25	61,9	25	50,1	25	50,1	25	49,9	26	76,7	25,2	0,4	25	26	60,2	10,4	49,9	76,8		
SCP61U1U1U1	5	4,6	5	4,6	5	4,6	5	4,6	5	4,6	5	4,6	5	4,6	5	4,6	5	4,6	5	4,6	5,0	0,0	5	5	4,6	0,0	4,6	4,6		
SCP61U1U1U1U	3	2,3	3	2,4	3	2,4	3	2,4	3	2,3	3	2,3	3	2,3	3	2,3	3	2,3	3	2,3	3,0	0,0	3	3	2,3	0,0	2,3	2,4		
SCP61U1U1U1U1	3	2,1	3	2,1	3	2,1	3	2,1	3	2,1	3	2,1	3	2,1	3	2,1	3	2,1	3	2,1	3,0	0,0	3	3	2,1	0,0	2,1	2,1		
SCP61U1U1U1U1U	29	1310	29	1250	29	1323	29	1318	29	1325	29	1311	29	1330	29	1254	29	1315	29	1314	29,0	0,0	29	29	1304,9	28,5	1250,0	1329,9		
SCP61U1U1U1U1U1	7	2,6	7	2,6	7	2,6	7	2,6	7	2,6	7	2,6	7	2,6	7	2,6	7	2,6	7	2,6	7,0	0,0	7	7	2,6	0,0	2,6	2,6		
SCP61U1U1U1U1U1r	4	2,1	4	2,2	4	2,2	4	2,2	4	2,2	4	2,2	4	2,2	4	2,2	4	2,2	4	2,1	4,0	0,0	4	4	2,2	0,0	2,1	2,2		
SCP61U1U1U1U1U1U	17	878	18	1286	17	875,2	17	1046	18	1284	17	1049	17	1050	17	1219	17	1050	17	1219	17,2	0,4	17	17	1095,6	151,5	875,2	1285,8		
SCP61U1U1U1U1U1U1	5	2,4	5	2,5	5	2,5	5	2,5	5	2,5	5	2,5	5	2,5	5	2,5	5	2,5	5	2,5	5,0	0,0	5	5	2,5	0,0	2,4	2,5		
SCP61U1U1U1U1U1U1U	3	2,1	3	2,2	3	2,2	3	2,2	3	2,2	3	2,1	3	2,1	3	2,1	3	2,1	3	2,1	3,0	0,0	3	3	2,1	0,1	2,1	2,2		
SCP61U1U1U1U1U1U1U1	63	87,2	64	68,4	63	67,7	61	90,7	63	67,6	64	93,4	63	67,6	62	66,9	63	67,7	64	68,6	63,0	0,9	61	64	74,6	11,0	66,9	93,4		
SCP61U1U1U1U1U1U1U1U	28	72,2	25	71,2	28	74,4	29	105,2	28	79,3	28	68,6	29	88,2	27	105,5	27	85,3	28	116,6	27,7	1,2	25	29	86,7	16,9	68,6	116,6		

Problème	Caractéristiques										B&B		Greedy1			Greedy2			Greedy3			Grasp1			Grasp2			Grasp3			Grasp5		
	N	K	Densité	Coef. C	Nr	Kr	LP	Z	Tps	Z	Tps	Z	Tps	Z	Tps	Z	Tps	Z	Tps	Z	Tps	Z	Tps	Z	Tps	Z	Tps	Z	Tps	Z	Tps		
SCP41	1000	200	2,0	WSCP	939	200	429	429	8	463	2,7	429	1,8	429	8,9	436	74,0	433	196,1	433	359,7	433	359,7	433	359,7	433	359,7	433	359,7	433	240,7		
SCP41r	500	100	2,0	WSCP	293	100	244	244	1,1	274	1	244	1,5	249	13,0	247	35,1	245	85,8	246	85,8	246	85,8	246	85,8	246	85,8	246	85,8	246	44,0		
SCP41U	1000	200	2,0	USCP	905	200	32,8			41	1,7	44	3,2	44	17,1	41,4	40,4	41,4	73,5	40,9	849,1	40,9	849,1	40,9	849,1	40,9	849,1	40,9	849,1	40,9	271,1		
SCP41Ur	500	100	2,0	USCP	267	100	28,1	29	179,8	34	0,9	37	1,1	37	1,9	31,6	8,6	31,7	14,7	31,4	122,6	31	122,6	31	122,6	31	122,6	31	122,6	31	43,9		
SCP61	1000	200	5,0	WSCP	999	200	133	138	1171,8	159	1,9	159	2,9	159	6,3	143	41,9	142	96,3	139	130,5	141	136,4	141	136,4	141	136,4	141	136,4	141	136,4		
SCP61r	500	100	4,9	WSCP	460	100	81	83	27,9	95	0,9	101	1,1	100	1,8	84,6	8,2	84	15,3	83,7	23,9	83,3	23,9	83,3	23,9	83,3	23,9	83,3	23,9	83,3	24,7		
SCP61U	1000	200	5,0	USCP	997	200	14,8			23	1,2	25	6,9	25	13,7	22,2	22,6	21,9	40,9	22	246,7	21,8	116,9	21,8	116,9	21,8	116,9	21,8	116,9	21,8	116,9		
SCP61Ur	500	100	4,9	USCP	447	100	13	16	75224	18	0,8	17	1,4	17	2,4	16,9	5,1	16,8	8,6	17	36,8	16,7	21,9	16,7	21,9	16,7	21,9	16,7	21,9	16,7	21,9		
SCPA1	3000	300	2,0	WSCP	2903	300	247			288	10,1	285	11,3	285	90,3	260	355,9	258	1124,2	257	1548,5	257	1548,5	257	1548,5	257	1548,5	257	1548,5	257	1229,7		
SCPA1r1	500	50	1,8	WSCP	119	50	89	89	0,5	103	0,8			89	1	92	4,5	91	15,9	89	34,1	89	34,1	89	34,1	89	34,1	89	34,1	89	17,1		
SCPA1r2	1000	100	2,0	WSCP	564	100	130	130	2,3	144	1,4			130	2,7	134	24,0	133	69,5	132	146,2	132	146,2	132	146,2	132	146,2	132	146,2	132	91,2		
SCPA1U	3000	300	2,0	USCP	2871	300	29,5			42	5,4	44	25,9	44	177	41,8	181,9	41,9	495,3	41,7	3686,6	40,9	1210,5	40,9	1210,5	40,9	1210,5	40,9	1210,5	40,9	1210,5		
SCPA1Ur1	500	50	1,8	USCP	101	50	17,7	18	1,4	20	0,7			19	1,1	18,7	3,0	18,6	6,8	18	20,6	18,4	12,2	18,4	12,2	18,4	12,2	18,4	12,2	18,4	12,2		
SCPA1Ur2	1000	100	2,0	USCP	518	100	22,1	25	172442,2	26	1			28	4	25,4	13,1	25,7	34,3	25,4	154,3	25,2	60,2	25,2	60,2	25,2	60,2	25,2	60,2	25,2	60,2		
SCPE1	500	50	20,0	USCP	492	50	3,5	5	266,4	5	0,7	6	1,1	6	1,6	5,1	1,7	5	2,9	5	3,7	5	4,6	5	4,6	5	4,6	5	4,6	5	4,6		
SCPE1r1	200	20	24,8	USCP	140	20	2,6	3	13,6	4	0,7	4	0,9	4	1,1	3,1	1,0	3,1	1,2	3,4	1,2	3	2,3	3	2,3	3	2,3	3	2,3	3	2,3		
SCPE1r2	100	10	29,3	USCP	29	10	2,1	3	1,6	3	0,7	3	0,9	3	0,9	3	1,0	3	1,0	3	1,0	3	2,1	3	2,1	3	2,1	3	2,1	3	2,1		
SCPNRE1	5000	500	10,0	WSCP												29	395,9	29,1	997,1	29	1001,7	29	1304,9	29	1304,9	29	1304,9	29	1304,9	29	1304,9		
SCPNRE1r1	200	20	10,7	WSCP	80	20	6,6	7	0,6	7	0,7	7	1	7	0,9	7	1,1	7	1,3	7	1,4	7	2,6	7	2,6	7	2,6	7	2,6	7	2,6		
SCPNRE1r2	100	10	11,2	WSCP	20	10	4	4	0,4	4	0,7			4	1	4	1,0	4	1,0	4	1,0	4	2,2	4	2,2	4	2,2	4	2,2	4	2,2		
SCPNRE1U	5000	500	10,0	USCP												18,7	254,6	18,6	645,4	18,3	2361,1	17,2	1095,6	17,2	1095,6	17,2	1095,6	17,2	1095,6	17,2	1095,6		
SCPNRE1Ur1	200	20	10,7	USCP	72	20	4,3	5	1,1	5	0,7	5	0,9	5	0,9	5	1,1	5	1,2	5	1,3	5	2,5	5	2,5	5	2,5	5	2,5	5	2,5		
SCPNRE1Ur2	100	10	11,2	USCP	18	10	3	3	0,3	4	0,7			3	0,9	3	1,0	3	1,0	3	1,0	3	2,1	3	2,1	3	2,1	3	2,1	3	2,1		
SCPCYC6	192	240	2,1	USCP	192	240	48			60	1	68	1,4	68	2,1	64,3	15,0	64	16,1	62,9	443,5	63	74,6	63	74,6	63	74,6	63	74,6	63	74,6		
SCPCLR10	210	511	12,3	USCP	210	456	21			33	1	31	4,4	33	5	29,7	18,1	29,6	19,0	27,7	307,4	27,7	86,7	27,7	86,7	27,7	86,7	27,7	86,7	27,7	86,7		

Annexe VI : Glossaire des termes ferroviaires employés

Blanc-travaux : période pendant laquelle aucune circulation n'est prévue sur une voie donnée, de façon à permettre l'exécution de la maintenance courante de l'infrastructure. Généralement cette période est d'au moins 1h50 par voie et par jour.

Cadence : durée du cycle de l'horaire. Un horaire cadencé est conçu de façon à pouvoir se répéter à la fin de chaque cycle : il ne possède ni début ni fin. Dans la pratique, il va de soi que l'horaire possède un début et une fin, ce qui rompt bien entendu sa cadence.

Grille horaire : terme caractérisant en général le graphique d'un tronçon de ligne et représentant les heures de passage prévues des trains sur chacune des voies de la zone considérée.

Ligne : suite ordonnée de tronçons formant un chemin : l'origine de chaque tronçon doit coïncider avec la destination du tronçon précédent. Généralement, le parcours d'un train coïncide partiellement ou totalement avec une ligne ferroviaire bien définie.

Nœud : point critique du réseau, tels une gare, une bifurcation, une jonction, un point de croisement, y compris tous les appareils de voie et voies qui y sont rattachés.

Qualité de service : notion qualitative difficile à définir. Du point de vue de la clientèle du rail (voyageurs ou fret), la qualité de service s'appuie sur les axes suivants :

- des temps de parcours réduits (de point à point, c'est-à-dire y compris les transbordements)
- des fréquences étoffées
- une bonne ponctualité (respect de l'horaire)
- un confort de voyage et d'attente agréable (sièges, bruit, possibilités offertes...)
- un prix réduit

Par rapport à l'étude de la capacité, la qualité de service est prise en compte par les temps de parcours, les fréquences et la ponctualité (stabilité de l'horaire).

ROUTAGE : affectation de chacun des trains arrivant dans une zone à un parcours lui permettant de rejoindre sa destination.

Saturation : pratique consistant à introduire dans le réseau une série de trains, dits saturants, jusqu'à ce qu'il ne soit plus possible de le faire. La capacité du réseau est définie par le nombre de trains à l'horaire saturé.

Sillon : capacité d'infrastructure requise (voie et horaire) pour faire circuler un train donné d'un point à un autre à un moment donné (directive 95/19/CE du conseil de l'union européenne).

Réseau : ensemble de lignes.

Tronçon : partie d'une ligne composée éventuellement de plusieurs voies reliant deux nœuds du réseau.

UIC ou Union Internationale des Chemins de fer : organisme international faisant office de référence sur certains points précis . La majorité des réseaux ferrés du monde sont adhérents à l'UIC.

Annexe VII : Plan de voie de Pierrefitte-Gonesse [Auguet96]

L'objectif de ce mémoire est de présenter l'application de méthodes issues du domaine de l'optimisation combinatoire à des problèmes de capacité d'infrastructure ferroviaire. Pour cela, nous nous inspirons d'une modélisation proposée dans le cadre d'un projet développé par les chemins de fer néerlandais.

Après un court rappel de problèmes classiques rencontrés en optimisation et des méthodes existantes permettant de les résoudre, nous nous intéressons plus en détail à la résolution du Set Covering Problem. Une expérimentation nous permet ainsi d'étudier le comportement et les performances de plusieurs algorithmes (gloutons, GRASP, Branch & Bound) et de mettre en évidence l'intérêt de la métaheuristique GRASP. Nous utilisons ensuite cette méthode pour évaluer le problème de la faisabilité du passage des trains (modélisée sous la forme d'un Set Packing Problem). Les résultats obtenus sur le cas réel du nœud ferroviaire de Pierrefitte-Gonesse permettent d'envisager l'étude de problèmes de plus grande taille et d'autres questions liées à la notion de capacité (fluidification du trafic, modifications à entreprendre).

Mots clés : Optimisation combinatoire
 Algorithme de Branch & Bound
 Métaheuristique GRASP
 Capacité d'infrastructure ferroviaire