Optimisation combinatoire multi-objectif : des méthodes aux problèmes, de la Terre à (presque) la Lune

Nicolas Jozefowiez

Institut National Polytechnique de Toulouse

Numéro d'Ordre : Année : 2013

Habilitation à diriger les recherches

préparée au

Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS

présentée et soutenue publiquement,
le 03/12/2013, par

Nicolas Jozefowiez

Maître de conférences en informatique à

Institut National des Sciences Appliquées de Toulouse

Titre :

Optimisation combinatoire multi-objectif : des méthodes aux problèmes, de la Terre à (presque) la Lune

Jury :

Rapporteurs :	M. Gendreau	Professeur titulaire, Ecole Polytechnique de Montréal
	P. Mahey	Professeur des universités, Université Blaise Pascal
	D. Vanderpooten	Professeur des universités, Université Paris Dauphine
Examinateurs :	G. Laporte	Professeur titulaire, HEC Montréal
	P. Lopez	Directeur de recherche, LAAS-CNRS
	F. Messine	Maître de Conférences-HdR, ENSEEIHT-INPT
	F. Semet	Professeur des universités, Ecole Centrale de Lille
	D. Vigo	Professeur, Università di Bologna

Remerciements

Je remercie les membres du Jury pour avoir accepté de participer à ce jury et d'avoir consacré une partie de leur temps à cette tâche, notamment Frédéric Messine qui a eu la gentillesse d'accepter d'être mon correspondant INPT. Je présente tous mes remerciements à Frédéric Semet qui, après avoir été mon directeur de thèse, m'a fait l'honneur et la gentillesse d'être président de ce jury.

Je remercie aussi l'équipe ROC qui est plus qu'une simple équipe de recherche et ses membres, dont certains sont plus que des collègues : Christian, Cyril, Emmanuel, Laurent, Marie-José, Pierre, Sandra. Je remercie aussi mes doctorants passés et présents : Panwadee, Boadu, Leonardo et Leticia. Travailler à Toulouse est agréable aussi de par la qualité des personnes que l'on y côtoie : Alain, Aude, Catherine, Marcel, Pierre-Emmanuel, Sonia... Je tiens aussi à saluer ici tous les "amis de recherche" ; ceux de la thèse notamment Laetitia et Matth ou qui nous "surveillait" comme Clarisse ; ceux rencontrés au loin dans le froid Québecquois : Fausto, Gunes, Maria, Massimo, Stefan, Tolga ... ; et ceux que j'ai eu la chance de rencontrer ici ou ailleurs : Adrien, Murat, Thierry ... Je remercie aussi les personnes qui ont su être une aspiration de par leur rencontre ou la collaboration avec eux et qui sont trop nombreux pour être cités ici. Je tiens cependant à remercier particulièrement Michel et Théo pour m'avoir offert une chance et à Gilbert.

Enfin, je remercie toutes ces personnes qui font que les journées ne se résument pas à tenter de résoudre des problèmes : Célia, Dana, Dim, Jean-Paul, Julie, Naïm, Nicolas ... Et last but not least, il me reste à remercier ma famille.

Il est probable que j'ai oublié de nombreuses personnes. A tous, merci.

"Questions don't have to make sense [...] but answers do."
[Terry Pratchett, *Thief of time*]

"Logic is a wonderful thing but doesn't always beat actual thought."
[Terry Pratchett, *The last continent*]

Table des matières

Liste des tableaux

Table des figures

Parcours depuis le doctorat

Ma thèse, effectuée entre 2001 et 2004 au LIFL de l'Université de Lille 1, portait sur l'utilisation de méta-heuristiques évolutionnistes multi-objectif appliquées à des problèmes de tournées de véhicules et améliorées via l'utilisation du parallélisme et de la coopération entre différentes méthodes. Différentes méthodologies avaient été proposées et appliquées à deux problèmes représentatifs de l'utilisation de l'optimisation multi-objectif. A la suite de ma thèse, j'ai effectué une série de postdoctorats. Tout d'abord, j'ai passé une année à l'Université du Colorado à Boulder, financée par une bourse Fulbright. Puis, j'ai occupé un poste d'ATER à l'Université de Valenciennes et du Hainaut-Cambrésis et finalement j'ai été stagiaire postdoctoral au Centre Interuniversitaire de Recherche sur les Réseaux d'Entreprise, la Logistique et le Transport (CIRRELT) à Montréal, Canada.

J'ai été recruté en février 2008 sur un poste de maître de conférences au Département de Génie Electrique et Informatique de l'Institut National des Sciences Appliquées de Toulouse. Au niveau recherche, j'ai été affecté à l'équipe Modélisation, Optimisation et Gestion Intégrée des Systèmes d'Activité (MOGISA) du Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS-CNRS). L'équipe MOGISA travaille sur l'optimisation combinatoire avec un intérêt initial pour la programmation par contrainte et l'ordonnancement. Cependant avant mon arrivée, des recrutements et un élargissement des domaines d'application ont poussé l'équipe à s'intéresser à la recherche opérationnelle plus globalement, notamment la programmation mathématique et son application au transport. C'est donc par rapport à ce point que mon recrutement a eu lieu.

J'ai continué à m'intéresser à l'optimisation multi-objectif, en devenant notamment co-animateur du groupe de travail PM2O [1] de la ROADEF. Cependant, dans un premier temps, je me suis moins focalisé sur les méta-heuristiques et plus sur les méthodes de recherche arborescente pour l'optimisation multi-objectif. J'ai aussi élargi les domaines applicatifs en menant des études sur des problèmes, non nécessairement multi-objectif, dans le cadre du transport aérien et du spatial. Actuellement, mes recherches s'inscrivent sur deux axes qui ne sont pas complètement hermétiques l'un envers l'autre : i) la proposition de méthodologies pour traiter des problèmes multi-objectif ; ii) l'étude de problèmes en transport terrestre ou dans le spatial. Cela se reflète dans les étudiants en doctorat que j'encadre sur les sujets suivants : génération de colonnes pour l'optimisation multi-objectif, méta-heuristiques multi-objectif pour l'observation de la Terre par satellites, gestion de réseau de services porte-à-porte et la proposition de nouvelles méta-heuristiques multi-objectif pour l'humanitaire.

1. http ://www.lifl.fr/PM2O/ (Programmation Mathématique Multi-Objectif).

Introduction

Le thème le plus général de cette thèse est l'optimisation combinatoire. Cela consiste à trouver parmi un ensemble discret de solutions respectant des contraintes une solution qui optimise une fonction objectif. Par optimiser, on entend trouver la plus petite valeur (problème de minimisation) ou la plus grande valeur (problème de maximisation). De nombreux problèmes peuvent se modéliser comme des problèmes d'optimisation combinatoire. De manière large, les problèmes d'optimisation combinatoire peuvent se ranger dans deux classes de complexité : la classe $\mathcal{P}$ et la classe $\mathcal{NP}$-difficile. Dans la première classe, on trouve les problèmes considérés comme "faciles", c'est-à-dire pour lesquels il existe un algorithme polynômial pour trouver la solution. On ne connaît pas de tels algorithmes [2] pour les problèmes de la seconde classe. Dans ce document, on s'intéresse principalement à des problèmes $\mathcal{NP}$-difficiles.

Les méthodes d'optimisation pour ces problèmes peuvent alors être classées en deux catégories : les méthodes exactes et les méthodes approchées. Les méthodes de la première famille garantissent que la solution renvoyée est bien optimale, c'est-à-dire qu'il n'existe pas d'autres solutions pour lesquelles la fonction objectif a un meilleur score. Une méthode exacte basique serait d'énumérer toutes les solutions possibles du problème. Cependant, du fait du très grand nombre de solutions, exponentiel par rapport à la taille du problème, cette approche n'est pas envisageable, y compris pour des instances de petites tailles. On parle d'explosion combinatoire. Il est cependant possible de définir des méthodes permettant de repousser la taille des instances pouvant être résolue à l'optimalité. On peut citer notamment les énumérations implicites qui exploitent des informations fournies par des bornes inférieures et supérieures sur la valeur de l'objectif pour ne pas explorer toutes les solutions. Dans ce document, on utilisera notamment des approches utilisant la programmation linéaire en nombres entiers [141, 104]. Cependant, les méthodes dites exactes atteindront nécessairement une limite dans la taille des problèmes qu'elles peuvent traiter en temps de calcul "raisonnable", en dépit des avancées méthodologiques et de l'augmentation de la puissance de calcul des ordinateurs. Il est alors possible d'utiliser des méthodes approchées, soit des heuristiques ad-hoc soit des méthodes exploitant les stratégies plus généralistes que sont les méta-heuristiques. Ces approches trouvent des solutions sans garantie sur leur qualité en contrepartie d'un temps de calcul moins prohibitif ou plus facilement maîtrisable qu'une méthode exacte [127].

Ce document s'intéresse particulièrement aux problèmes combinatoires qui comportent plusieurs objectifs. On parle alors d'optimisation combinatoire multi-objectif. Ce domaine possède ses sources dans les travaux de Edgeworth [42] et de Pareto [108] dans le cadre d'études d'économie au $19^{\text{ème}}$ siècle. Cependant, l'optimisation multi-objectif connaît un intérêt croissant depuis le milieu des années 1980 [125] et le domaine connaît une expansion importante depuis le milieu des

2. ce qui ne signifie pas qu'il n'en existe pas.

années 1990 avec l'apparition de méthodes évolutionnaires pour l'optimisation multi-objectif [30]. Actuellement, l'optimisation multi-objectif est appliquée dans de nombreux domaines académiques et industriels. De manière formelle, un problème d'optimisation multi-objectif est un problème de la forme $\min F(x) = (f_1(x), f_2(x), \ldots, f_n(x))$ tel que x soit une solution réalisable. La solution d'un problème multi-objectif n'est pas une unique solution mais un ensemble de solutions appelé ensemble non-dominé. Les composantes du vecteur F sont les différentes fonctions à optimiser. Les solutions de cet ensemble sont nommées solutions non-dominées. Ces solutions sont celles pour lesquelles l'amélioration d'un des objectifs entraîne systématiquement la détérioration de la qualité d'au moins un autre objectif. La présence d'objectifs contradictoires apporte de nouveaux défis à relever pour les méthodes d'optimisation. Notamment, s'il existe une littérature foisonnante dédiée aux méthodes approchées, l'application de méthodes exactes est moins répandue.

Ce document n'est pas un reflet complet de l'ensemble des travaux que j'ai effectués depuis mon doctorat. Il se focalise sur mes travaux portant sur des méthodes pour l'optimisation multi-objectif et leur application à des problèmes de tournées de véhicules multi-objectif ou à une problématique dans le spatial. Des études portant sur des problèmes de tournées multi-objectif ont toutefois été omises ici. Il s'agit d'études de problèmes de tournées avec profits [69] et du problème d'anneau étoilé [95]. Des travaux purement mono-objectif sur un problème de tournées sur arc [2] et dans une problématique de transport aérien [72] sont aussi omis. Enfin une étude sur un problème d'ordonnancement bi-objectif [6] ne se retrouve pas non plus ici.

Le manuscrit s'articule ainsi. Il se divise en deux. La première moitié, composée de la première partie, est dédiée aux concepts d'optimisation combinatoire multi-objectif et aux méthodes. La seconde moitié, composée des seconde et troisième parties, est consacrée aux applications à des problèmes. Cette seconde moitié se divise elle-même en deux entre les applications en transport terrestre illustrées par des problèmes de tournées et une application spatiale. Cette dernière porte sur l'observation de la Terre par des satellites. Les satellites d'observation volent à 700-1000 kms d'altitude environ, on est donc un peu loin de la lune promise dans le titre du document mais on s'en approche.

Le chapitre 1 pose le contexte du travail : l'optimisation combinatoire multi-objectif. Les principales définitions sont présentées ainsi que les problématiques spécifiques par rapport à l'optimisation standard. Les principales stratégies de prise en compte de la présence de plusieurs objectifs sont aussi exposées. Un panorama des méthodes de la littérature est aussi fait. Les différents mécanismes, outils et mesures utiles pour le reste du document sont également introduits dans ce chapitre.

Le chapitre 2 est dédié aux méthodes pour la résolution de problèmes d'optimisation combinatoire multi-objectif. Dans un premier temps, les caractéristiques recherchées dans les méthodes proposées sont expliquées ainsi qu'une approche pour traiter des problèmes multi-objectif. Ensuite, trois méthodes sont expliquées. La première est un algorithme génétique utilisant le concept de clefs aléatoires. Son utilisation est illustrée sur des problèmes de tournées de véhicules. La seconde méthode est un algorithme exact de type séparations et coupes. Enfin, la troisième méthode porte sur la génération de colonnes pour l'optimisation combinatoire multi-objectif notamment pour calculer des bornes inférieures.

Le chapitre 3 traite du problème du voyageur de commerce avec labels. Le problème est défini comme un problème bi-objectif et une modélisation par programmation linéaire en nombres entiers est proposée. Ensuite, une analyse polyèdrale est effectuée. Un algorithme de séparations et coupes est donné pour plusieurs variantes mono-objectif du problème. Le chapitre se conclut

par la description de l'implémentation de l'algorithme de séparations et coupes présenté dans le chapitre 2 pour le problème dans sa version bi-objectif. Des résultats expérimentaux comparent les résultats de la méthode bi-objectif face à une approche basique itérant l'algorithme mono-objectif guidé par une méthode ϵ-contrainte.

Le chapitre 4 s'intéresse au problème de tournées couvrantes et à l'étude de sa généralisation bi-objectif. Il s'agit aussi d'un exemple d'une famille de problèmes de tournées où la visite de tous les sommets n'est pas obligatoire. Dans un premier temps, un algorithme de branch-and-price est proposé pour la version mono-objectif pour évaluer le modèle utilisé en termes de qualité de la borne inférieure fournie. Puis les stratégies pour la génération de colonnes proposées dans le chapitre 2 sont appliquées à la version bi-objectif et comparées entre elles.

Le chapitre 5 porte sur un problème d'observation de la Terre par satellite. Le but est de prendre des vues de la Terre. Cependant, l'ensemble des vues demandées ne peuvent pas être toutes prises du fait par exemple de conflits entre leurs fenêtres de visibilité ou de limitation matérielle du satellite. Ainsi, un profit est associé à chaque acquisition et l'objectif vise à maximiser le profit. Dans cette étude, on se place dans un cas multi-utilisateurs, ce qui entraîne un second objectif dont le but est d'équilibrer les profits des acquisitions prises entre les utilisateurs. Deux méta-heuristiques multi-objectif, un algorithme génétique et une recherche locale, sont implémentés et comparés.

Le dernier chapitre, conclusions et perspectives, fait une synthèse des principaux apports des travaux présentés dans le manuscrit et offre des perspectives à long terme.

Le reste du manuscrit est composé de trois annexes :

- un CV détaillé,
- une liste des publications réalisées,
- un ensemble de publications.

Première partie

Principes généraux et méthodes

Chapitre 1

Optimisation combinatoire multi-objectif

Les apports méthodologiques des travaux présentés dans le chapitre 2 portent sur l'optimisation combinatoire multi-objectif. L'optimisation multi-objectif trouve ses racines dans les travaux en économie de Edgeworth [42] et de Pareto [108]. L'étude de méthodes d'optimisation avancées et dédiées à ces problèmes a pris son essor dans les années 80 [125], mais surtout avec l'application des algorithmes évolutionnistes multi-objectif dans la seconde moitié des années 90 [36]. Si les études se sont principalement centrées dans un premier temps sur les méta-heuristiques, récemment un intérêt croissant s'est porté sur les méthodes de recherche arborescente et les méthodes issues de la programmation mathématique. Ce chapitre ne revendique pas présenter l'ensemble des techniques ou des avancées récentes dans le domaine de l'optimisation multi-objectif. Il sert uniquement à rappeler les principes de base de l'optimisation combinatoire multi-objectif ainsi que les principaux mécanismes employés dans ce document. Pour une présentation plus complète, le lecteur est invité à consulter des documents de référence comme le livre de Ehrgott [43]. La section 1.1 fournit les définitions de base, tandis que les sections 1.2 et 1.3 abordent les possibilités et l'intérêt d'utiliser l'optimisation multi-objectif. Les techniques de prise en compte des objectifs sont décrites dans la section 1.4. La section 1.5 se focalise sur les méthodes d'optimisation employées couramment en optimisation combinatoire multi-objectif.

1.1 Définitions de base

Un problème d'optimisation multi-objectif est un problème de la forme suivante :

$$(PMO)\ \text{minimiser}_{x \in D}\ F(x) = (f_1(x), f_2(x), \ldots, f_n(x)) \tag{1.1}$$

avec $n \geq 2$ le nombre de fonctions objectif, $x = (x_1, x_2, \ldots, x_r)$ un vecteur de variables de décision ou solution, D est l'ensemble des solutions réalisables et $F(x)$ est le vecteur objectif. L'ensemble $O = F(D)$ correspond aux images des solutions réalisables dans l'espace des objectifs et $y = (y_1, y_2, \ldots, y_n)$, avec $y_i = f_i(x)$, est le point de l'espace des objectifs correspondant à la solution x.

La principale différence avec l'optimisation mono-objectif vient de la définition d'optimalité. Il s'agit de l'optimalité de Pareto qui est définie par la dominance de Pareto [108] :

Définition 1.1. *Une solution x domine ($\preceq$) au sens de Pareto une solution z si et seulement si $\forall i \in \{1, \ldots, n\}, f_i(x) \leq f_i(z)$ et $\exists i \in \{1, \ldots, n\}$ tel que $f_i(x) < f_i(z)$.*

Dans le reste de ce manuscrit, par abus de notation, il sera parfois écrit $x \preceq z$ alors que x ou z représentent directement un point dans l'espace des objectifs et non pas une solution. Le principal impact de cette dominance est qu'il n'y a plus un ordre total entre les solutions débouchant sur un optimum global (si la fonction objectif est bornée), mais uniquement un ordre partiel. Une solution réalisable dominée par aucune autre solution réalisable est dite **efficace** ou **Pareto optimale**. Son image dans l'espace des solutions est qualifiée de **non-dominée**. La solution d'un problème multi-objectif est donc un ensemble de toutes les solutions efficaces appelé **l'ensemble efficace** ou **Pareto optimal** et l'ensemble de points correspondant dans l'espace des objectifs est **l'ensemble des points non-dominés**. De la même manière que l'on cherche généralement un optimum et une solution optimale et non toutes les solutions optimales, on recherche souvent l'ensemble non-dominé et un ensemble efficace comportant une solution par point non-dominé. Si on recherche toutes les solutions efficaces, on parle d'ensemble efficace **complet**.

Lorsque l'on ne recherche pas l'ensemble efficace mais une approximation, cette dernière est dite **potentiellement efficace**. C'est-à-dire qu'elle est formée des solutions trouvées par l'heuristique qui ne sont pas dominées par une autre solution trouvée par l'heuristique sans garantie d'optimalité. Le but sera alors de proposer une méthode combinant une qualité de convergence vers l'ensemble non-dominé tout en assurant une diversification des solutions pour obtenir un ensemble représentatif. Ces notions de convergence/diversification dans un cadre bi-objectif sont illustrées dans la figure 1.1.

Lors du design de méthodes heuristiques, il est nécessaire d'évaluer la qualité des résultats retournés. En optimisation mono-objectif, il s'agit de comparer la meilleure valeur trouvée avec celle d'une autre méthode, la meilleure valeur connue pour une instance, l'optimum ou une borne. Cette comparaison est directe puisqu'il s'agit de comparer deux valeurs. Dans le cadre multi-objectif, cela est plus complexe. Il n'y a pas forcément d'optimum connu pour un problème du fait de la complexité de le calculer et de l'attention limitée portée aux méthodes exactes jusque récemment. De même, la définition d'une borne duale n'est pas nécessairement immédiate. Comparer deux approximations n'est pas trivial non plus. Les approximations étant un ensemble de points, il est possible que deux ensembles ne soient pas comparables si l'un d'entre eux ne domine pas complètement le second. Une conséquence est que les mesures existantes ne sont pas toujours simples à interpréter car leur échelle peut changer d'une instance à l'autre par exemple et un écart donné peut ne pas représenter la même différence. De nombreuses mesures ont été proposées dans la littérature mais elles ne sont pas toutes valables et ne permettent pas toutes de classer les résultats de manière significative [148]. Parmi les mesures existantes, certaines donnent un score absolu à une approximation, d'autres un score relatif à un ensemble ou un point de référence, tandis que d'autres calculent l'apport d'une approximation par rapport à une autre. De plus, certaines mesures se concentrent sur l'aspect de convergence, d'autres sur l'aspect de diversification, tandis que certaines mesures évaluent les deux aspects simultanément. Le but n'est pas ici de lister l'ensemble des mesures existantes et leurs atouts et faiblesses. Nous nous contenterons d'introduire une mesure type qui sera utilisée dans le manuscrit : l'hypervolume ou $\mathcal{S}$-métrique [144].

Cette mesure calcule l'hypervolume de la région multidimensionnelle fermée par un ensemble potentiellement non-dominé $\mathcal{P}_A$ et un point de référence ; c'est-à-dire la taille de l'espace des objectifs que $\mathcal{P}_A$ domine. Ce concept est illustré dans la figure 1.2. L'avantage de cette mesure,

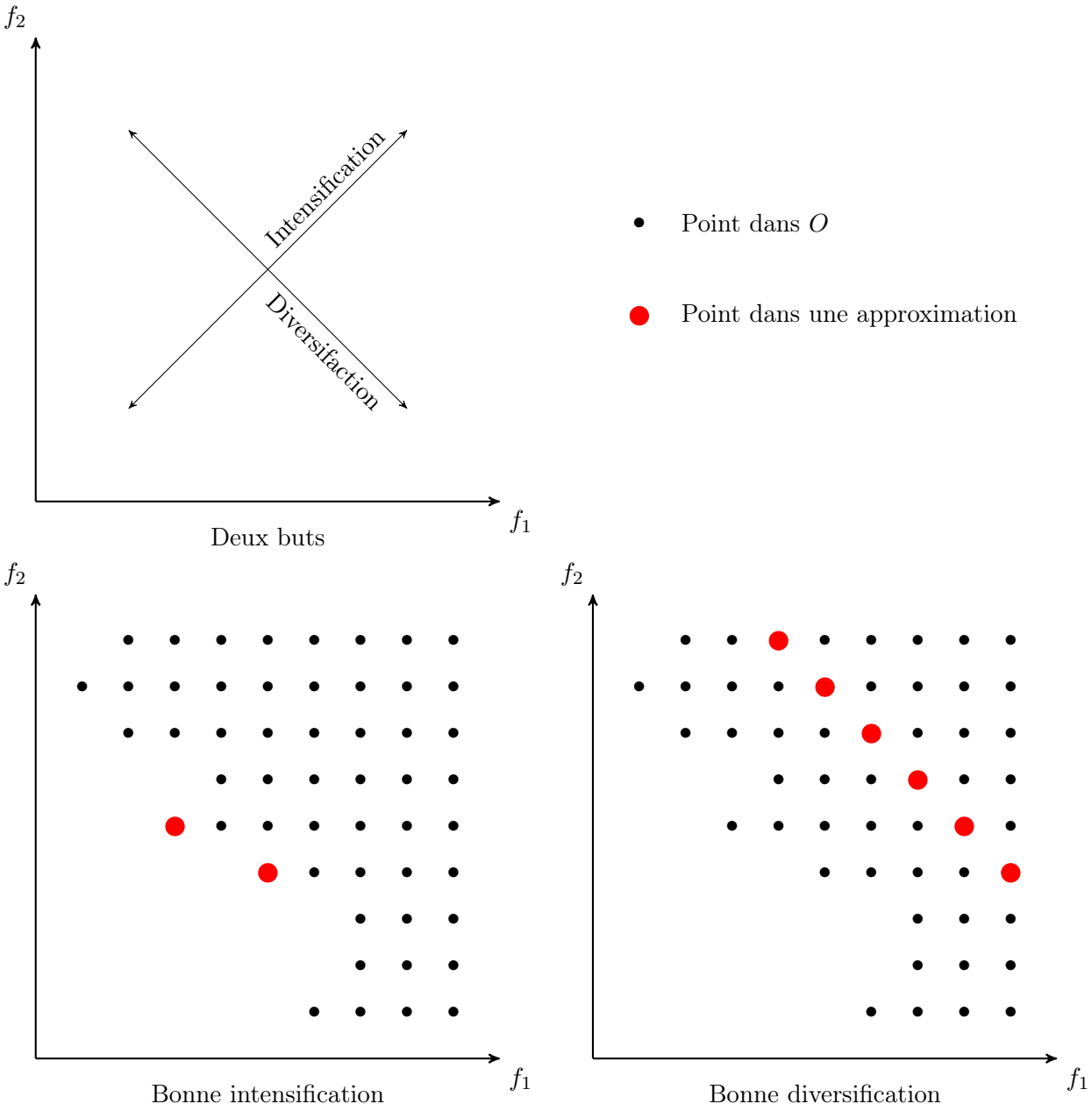


Fig. 1.1 – Les deux buts de l'optimisation multi-objectif dans la recherche d'une approximation.

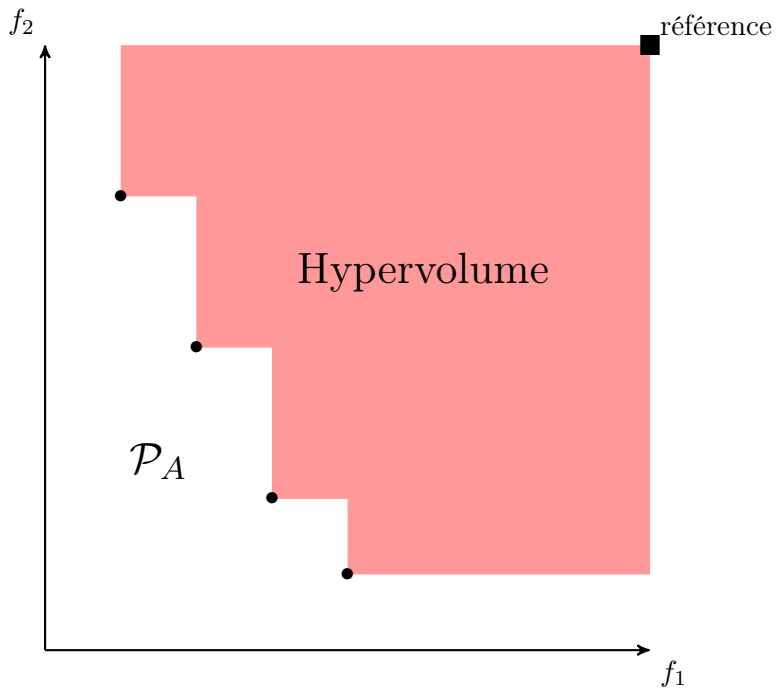


Fig. 1.2 – Représentation de la surface dominée par un ensemble par rapport à un point de référence calculé par l'hypervolume.

dont la signification est intuitive, est qu'elle offre un ordre total entre différentes approximations. Cependant, elle nécessite la définition d'une borne supérieure de la région dans laquelle se trouvent tous les points réalisables. De plus, le temps de calcul est important : $O(|\mathcal{P}_A|^{n+1})$. La mesure est impraticable lorsque de nombreux objectifs sont considérés ou si la cardinalité des ensembles est trop importante. Il est alors possible de calculer une approximation de la mesure. Dans leur étude [80], Knowles et Corne recommandent l'utilisation de l'hypervolume.

1.2 Approches de résolution

La solution d'un problème multi-objectif est un ensemble de solutions. Cependant, pour un problème réel, une seule solution pourra être déployée. Un choix par un décideur doit donc être effectué ; le décideur peut intervenir en amont de la résolution, après celle-ci, ou de manière interactive :

- Préférence *a priori* : le décideur définit ses préférences entre les différents objectifs avant d'utiliser la méthode d'optimisation.
- Préférence *progressive* : le décideur affine son choix de compromis au fur et à mesure du déroulement de la méthode d'optimisation.
- Préférence *a posteriori* : le décideur choisit la solution de son choix parmi l'ensemble des solutions fournies par la méthode d'optimisation.

TABLE 1.1 – Valeurs des objectifs pour les meilleures solutions trouvées par Taburoute et l'AG de Prins.

	Taburoute		AG de Prins	
Instance	Distance	Equilibre	Distance	Equilibre
E51-05e	524.61	20.07	524.61	20.07
E76-10e	835.32	78.10	835.26	91.08
E101-08e	826.14	97.88	826.14	97.88
E151-12c	1031.17	98.24	1031.63	100.34
E200-17c	1311.35	106.70	1300.23	82.31
E121-07c	1042.11	146.67	1042.11	146.67
E101-10c	819.56	93.43	819.56	93.43

Il est à noter que certaines méthodes n'entrent pas forcément dans une seule catégorie. En effet, la méthode qui consiste à agréger les différents objectifs à l'aide de poids est une méthode à préférence a priori ; le décideur affectant les poids de manière à favoriser tel ou tel objectif. Cependant, si les poids sont affectés aléatoirement et si la méthode est itérée en changeant les poids à chaque exécution, il s'agit alors d'une méthode à préférence a posteriori.

Nous nous intéressons ici à *l'optimisation a priori* de problèmes multi-objectif, c'est-à-dire l'étude de méthodes pour générer un ensemble de solutions efficaces ou potentiellement efficaces. Le choix d'une solution parmi cet ensemble n'est pas étudié dans ce document. On peut classifier cela - de manière un peu abusive - *d'aide à la décision multicritère*. On s'intéressera donc ici à l'optimisation et non à la décision dans le cadre de problèmes multi-objectif.

1.3 Utilité et utilisation de l'optimisation multi-objectif

Il est possible de s'interroger sur l'utilité de l'optimisation multi-objectif. Le but ici n'est pas de donner une réponse absolue mais de donner deux éléments de réponse. Le premier consiste à poser la question : tous les problèmes peuvent-ils se limiter à un seul objectif ? La réponse est clairement non. Pour illustrer cela, considérons un problème classique : le problème d'élaboration de tournées de véhicules avec capacité (CVRP). Il s'agit de trouver un ensemble de tournées partant d'un dépôt pour une flotte de véhicules définis par une capacité pour aller répondre à des demandes de clients. La demande totale étant plus grande que la capacité d'un véhicule, il est nécessaire de séparer les clients en plusieurs tournées et de les attribuer aux véhicules. L'objectif classique est la minimisation de la somme des longueurs des tournées. Si on s'intéresse à une équité entre les conducteurs via l'équilibre en termes de distance parcourue [73, 74], on peut chercher à minimiser la différence entre la longueur de la plus longue tournée et celle de la plus courte. Le tableau 1.1 reporte les valeurs des deux objectifs pour les meilleures solutions trouvées par Taburoute [56] et l'algorithme génétique de Prins [109] sur des instances standard de la littérature [28, 29]. Il apparaît clairement qu'optimiser la longueur conduit à une solution de piètre qualité pour l'équilibre entre les tournées.

La seconde question vis-à-vis de l'optimisation multi-objectif consiste à se demander si, dans ce cas, une simple approche *a priori* ou même progressive ne suffirait pas et s'il est utile de

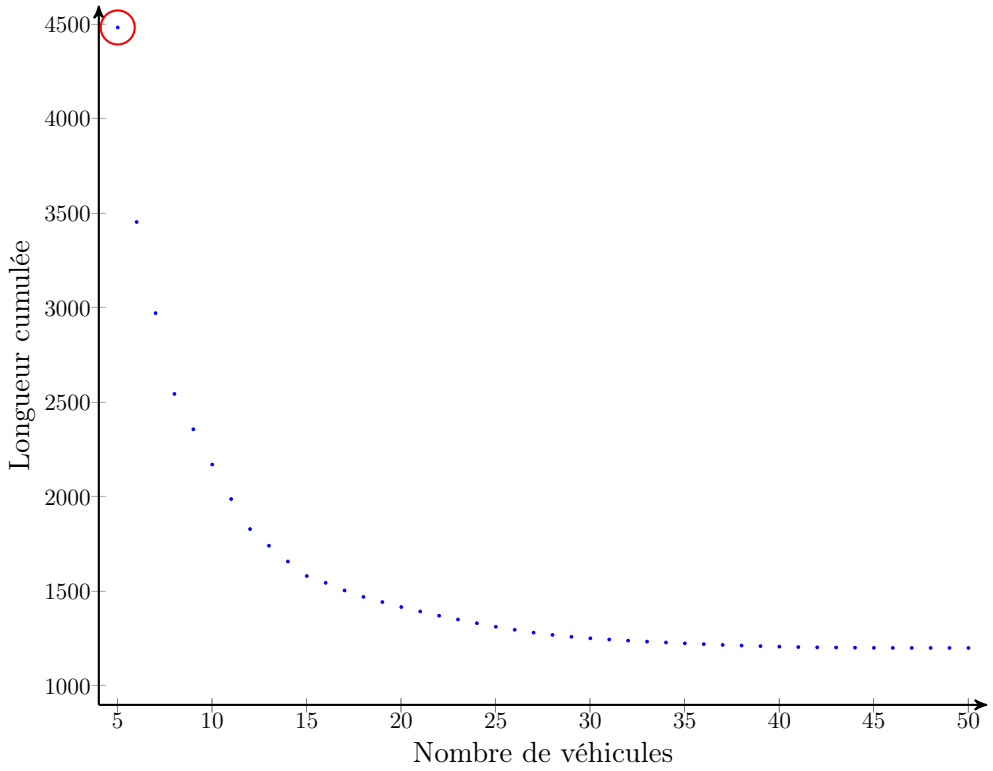


Fig. 1.3 – Un ensemble potentiellement non-dominé pour le CCVRP.

rechercher des méthodes *a posteriori* qui par nature sont plus complexes à définir et plus coûteuses en temps de calcul. Il faut d'abord remarquer que, souvent, émettre une préférence et encore plus une préférence précise n'est pas évident si l'on ne connaît pas le comportement des objectifs les uns envers les autres. Dans ce cas, les approches *a priori* ont un fort intérêt en tant qu'outil pour la décision. Nous allons illustrer cela sur le problème d'élaboration de tournées de véhicules cumulatives avec capacité (CCVRP) [105, 115]. Ce problème se différencie du CVRP par sa fonction objectif et une contrainte supplémentaire. Ici, la longueur d'une tournée est égale à la somme de la distance pour atteindre le premier client, la distance pour atteindre le second et ainsi de suite. Le but est de minimiser le temps d'atteinte d'un client. Une application typique provient de la logistique humanitaire en cas de catastrophe où il faut atteindre le plus rapidement possible l'ensemble des populations touchées. La nature de l'objectif impose une nouvelle contrainte qui est le nombre de véhicules à utiliser. En effet, de manière générale, la solution optimale utilisera autant de véhicules que possible pour accélérer le temps de réponse. Dans les deux études, le nombre de véhicules est fixé au nombre minimum de véhicules pour répondre à la demande totale. Si on considère les deux objectifs - minimisation des longueurs cumulées et minimisation du nombre de véhicules - et que l'on résout le problème par une approche *a priori*, on obtient une approximation de l'ensemble non-dominé telle que celle de la figure 1.3. La solution obtenue par les méthodes mono-objectif de la littérature correspond (à la qualité près) à la solution cerclée de rouge. L'optimisation multi-objectif permet de se rendre compte qu'en augmentant un peu le nombre de véhicules et en ne s'imposant pas une restriction qui peut être artificielle, on

peut gagner substantiellement en termes de temps de réponse. Par exemple, dans le cadre de la logistique humanitaire, cela peut aider un décideur à définir le nombre de véhicules issus d'une flotte de taille finie à déployer sur une zone d'opération.

1.4 Stratégies de prise en compte des objectifs

Il existe plusieurs stratégies pour prendre en compte la présence de plusieurs objectifs. Une approche sommaire est de garder à un moment donné toutes les solutions potentiellement efficaces trouvées. C'est le cas par exemple de la méthode de recherche locale employée dans [126] ou de Pareto Local Search (PLS) présentée dans [107]. Il est possible de coupler ce genre d'approche avec des techniques comme du clustering pour limiter la taille de l'ensemble de solutions géré. Un exemple d'une telle approche est Pareto Archived Evolution Strategy (PAES) [79]. Les stratégies ci-dessous ont pour but de prendre en compte la présence de plusieurs objectifs, soit en utilisant des mécanismes pour gérer leur présence comme dans les approches non-scalaires, soit en affectant un "score" à une solution qui peut être obtenu en combinant les objectifs (approches scalaires) ou directement selon la dominance de Pareto (approches Pareto). Enfin, il est possible d'évaluer la contribution d'une solution à la qualité générale d'un ensemble potentiellement efficace (approches basées sur les indicateurs).

1.4.1 Approches non-scalaires

Les méthodes **non-scalaires** (en opposition aux approches scalaires décrites ci-dessous) ont des mécanismes spécifiques pour chaque objectif et choisissent en considérant les objectifs individuellement [43]. Un exemple classique est Vector Evaluated Genetic Algorithm (VEGA) [122]. Cette méthode utilise un principe de "sélection parallèle" où la population principale est divisée en sous-populations, une pour chaque objectif. Chaque sous-population est formée des meilleurs individus pour l'objectif associé. La nouvelle population est alors générée à partir de ces sous-populations. Une autre approche non-scalaire couramment utilisée est la méthode lexicographique. Cette méthode, proposée par Fourman [54], classe les objectifs en fonction d'un ordre d'importance proposé par le décideur. Ensuite, les fonctions objectif sont traitées dans cet ordre pour obtenir l'optimum. La méthode lexicographique peut s'exprimer de la manière suivante : supposons que l'on ait les fonctions objectif f_i avec $i = 1, \ldots, n$ classées de telle sorte que si $i < j$ alors f_i est prioritaire sur f_j. On résout d'abord le problème :

$$(PMO_{l1}) = \begin{cases} min\ f_1(x) \\ t.q. \\ x \in \Omega \end{cases}$$

On obtient alors x_1^* la meilleure solution trouvée pour f_1, associée à la valeur $f_1^* = f_1(x_1^*)$. f_1^* devient alors une contrainte du problème associé à f_2 :

$$(PMO_{l2}) = \begin{cases} min\ f_2(x) \\ t.q. \\ x \in \Omega \\ f_1(x) = f_1^* \end{cases}$$

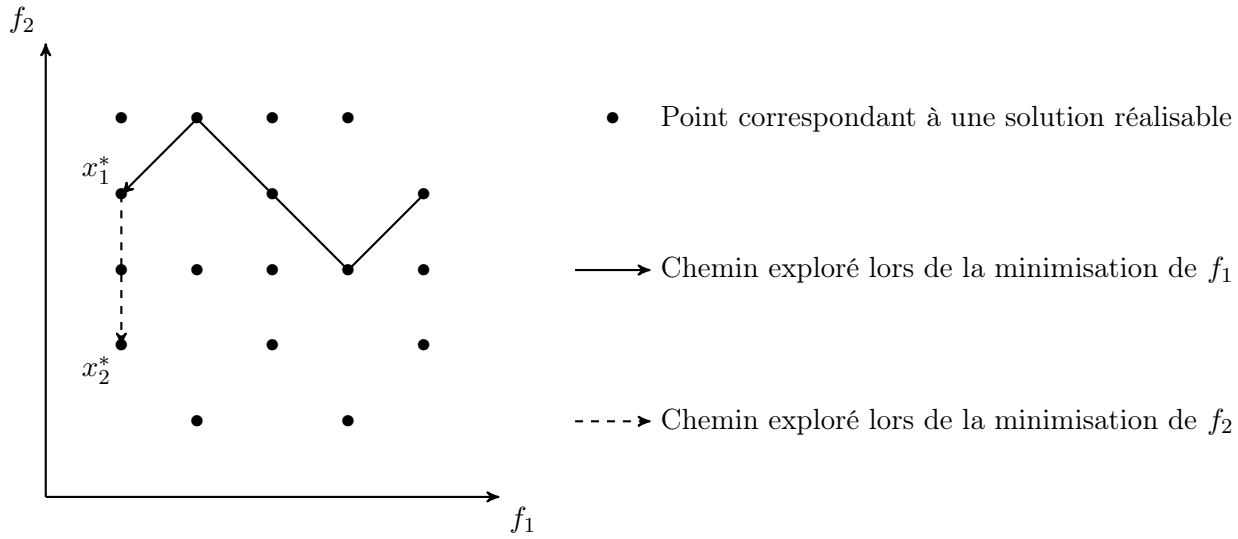


Fig. 1.4 – La méthode lexicographique dans le cas bi-objectif.

La meilleure solution trouvée est alors notée x_2^* et on pose $f_2^* = f_2(x_2^*)$ comme contrainte supplémentaire pour la résolution de f_3. La procédure est itérée jusqu'à ce que tous les objectifs aient été traités. Ainsi le problème associé à la fonction objectif f_i est le suivant :

$$(PMO_{li}) = \begin{cases} min\ f_i(x) \\ t.q. \\ x \in \Omega \\ f_1(x) = f_1^* \\ f_2(x) = f_2^* \\ \dots \\ f_{i-1}(x) = f_{i-1}^* \end{cases}$$

où $f_j^* = f_j(x_j^*)$ avec x_j^* la meilleure solution trouvée en optimisant la fonction objectif f_j avec $f_1(x) = f_1^*, f_2(x) = f_2^*, \dots, f_{j-1}(x) = f_{j-1}^*$ comme contraintes additionnelles. La figure 1.4 illustre le fonctionnement de la méthode lexicographique dans le cas d'un problème bi-objectif.

1.4.2 Approches scalaires

Une autre catégorie forme les approches dites **scalaires**. Le principe est de revenir à un problème mono-objectif via un ensemble de paramètres (poids ou contraintes sur les objectifs par exemple). Nous présentons ici deux méthodes scalaires : la somme pondérée et la méthode ϵ-contrainte [64, 27] . Il en existe d'autres telle que la méthode de Benson [14] qui impose des contraintes sur les objectifs à l'instar de la méthode ϵ-contrainte ou des méthodes cherchant à minimiser l'écart avec un point de référence [96, 128].

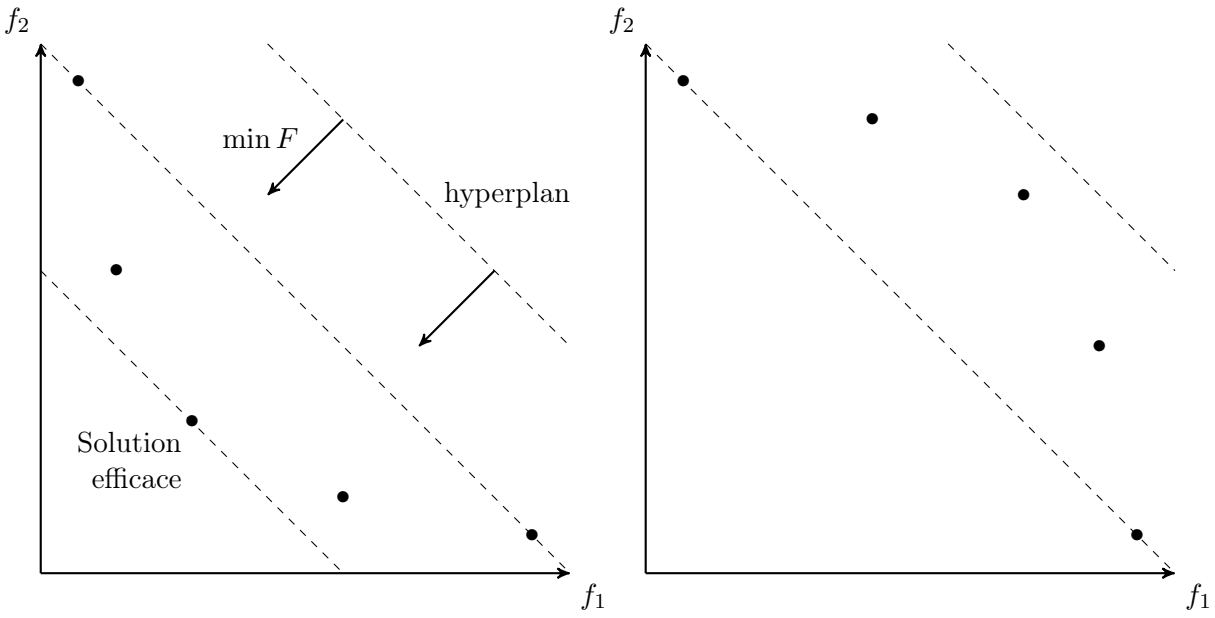


Fig. 1.5 – La méthode d'agrégation pour $\lambda_1 = \lambda_2 = 0.5$.

Somme pondérée

L'approche scalaire type et l'une des plus couramment utilisée est la somme pondérée. Elle consiste à transformer un problème multi-objectif en un problème qui combine les différentes fonctions objectif du problème en une seule fonction U de façon linéaire :

$$U(x) = \sum_{i=1}^{n} \lambda_i f_i(x)$$

où les poids λ_i sont compris dans l'intervalle $[0, 1]$ et vérifient $\sum_{i=1}^{n} \lambda_i = 1$. Différents poids fournissent différentes solutions ; une même solution pouvant être générée en utilisant des poids différents. Il existe d'autres approches agrégatives mais l'agrégation linéaire est la plus couramment utilisée.

La figure 1.5 illustre le fonctionnement de la méthode d'agrégation. Choisir un vecteur de poids revient à trouver un hyperplan dans l'espace objectif (une droite pour un problème bi-objectif) avec une orientation fixée. La solution optimale est le point où l'hyperplan possède une tangente commune avec l'espace réalisable (point x dans la figure).

L'avantage de cette méthode est sa facilité d'implémentation et le fait qu'elle puisse être utilisée avec les méthodes et mécanismes définis pour l'optimisation mono-objectif. Cependant, Das et Dennis [34] ont montré que cette méthode est incapable de trouver les zones concaves du front Pareto quels que soient les poids choisis. Dans la figure 1.5b, où le problème possède une frontière Pareto non convexe, seules les solutions extrêmes y et z peuvent être générées. Les solutions qui se trouvent sur cette enveloppe convexe de l'ensemble non-dominé sont appelées solutions supportées. Un algorithme n'utilisant que des agrégations ne pourra donc jamais trouver les solutions qui ne sont pas sur cette enveloppe et qui sont dites non-supportées.

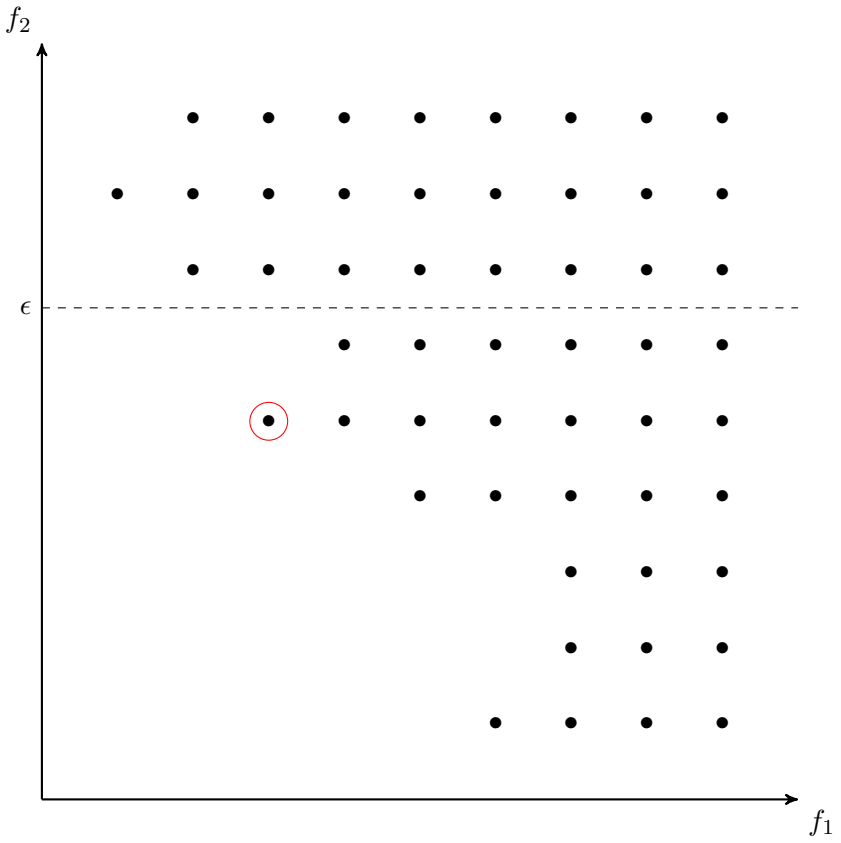


Fig. 1.6 – La méthode ϵ-contrainte.

Méthode ϵ-contrainte

Une autre méthode scalaire utilisée dans ce document est la méthode ϵ-contrainte. Dans cette méthode, une seule fonction objectif f_k est optimisée tandis que les autres fonctions objectif sont sujettes à des contraintes. Le problème d'optimisation s'écrit alors :

$$(PMO_\epsilon) = \begin{cases} min\ f_k(x) \\ t.q. \\ x \in \Omega \\ f_j(x) \leq \epsilon_j,\ j = 1, \ldots, n,\ j \neq k \end{cases}$$

La figure 1.6 illustre la méthode ϵ-contrainte où $F(C)$ est l'espace des objectifs original, restreint à $F'(C)$ par la transformation du problème, en minimisant la fonction f_1 et en ajoutant la contrainte $f_2(c) \leq \epsilon_2$.

L'objectif f_k représente l'objectif primaire ou l'objectif préféré. L'ensemble Pareto optimal est généré en faisant varier les valeurs des ϵ_j. Il est ainsi possible de générer l'ensemble Pareto optimal en n'ayant à résoudre qu'un ensemble de problèmes d'optimisation mono-objectif. Il est à noter que cette méthode est capable de générer les solutions non-supportées à la différence de la

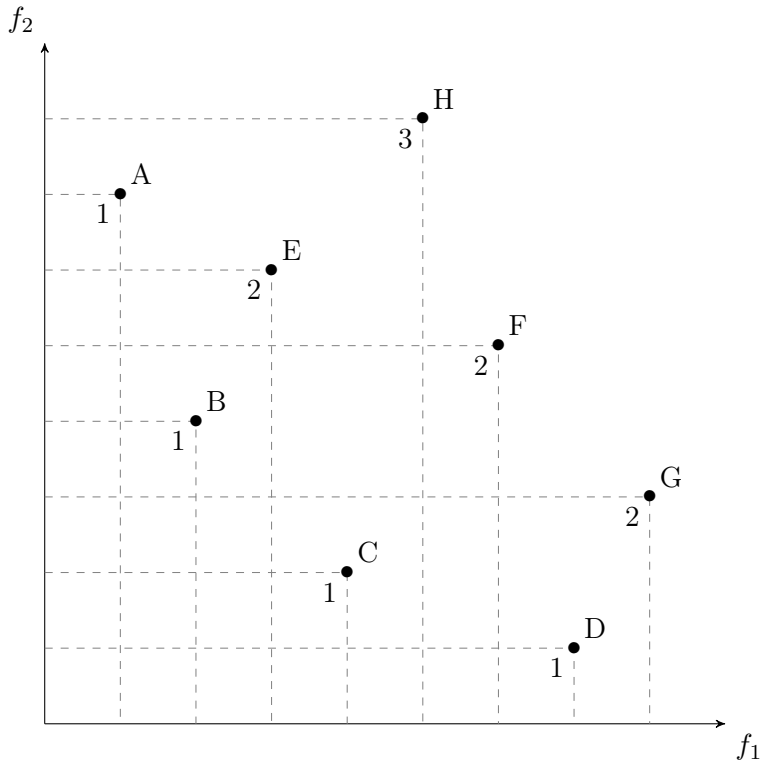


Fig. 1.7 – Exemple de ranking.

méthode d'agrégation. Cependant, l'ajout des contraintes modifie le polyèdre, ce qui peut avoir un effet négatif sur le calcul d'une borne inférieure par exemple.

1.4.3 Approches Pareto

Les approches Pareto utilisent la notion de dominance pour comparer les solutions et leur affecter un score ou sélectionner des solutions. Goldberg [58] a été l'un des premiers à utiliser cette notion. Ces approches ont connu un important développement en conjonction avec les algorithmes évolutionnistes à population à partir de la seconde moitié des années 90 [36]. Elles sont devenues la principale approche employée pour résoudre les problèmes multi-objectif du fait de leur capacité à trouver un ensemble potentiellement efficace via la recherche menée sur une population de solutions. De manière générale, les algorithmes évolutionnistes multi-objectif affectent un score à une solution selon qu'elle est dominée ou non par d'autres solutions de la population courante et éventuellement si elle domine d'autres solutions. Un exemple classique est NSGA II [39]. Cet algorithme génétique utilise une méthode de *ranking* qui attribue aux solutions non-dominées de la population courante le meilleur score. Puis les solutions, qui ne sont dominées que par les solutions potentiellement efficaces, reçoivent le second meilleur score et ainsi de suite. De cette manière, la population est organisée en *couches* où chaque couche contient des solutions non comparables entre elles. Le ranking d'une population est illustré dans la figure 1.7. Dans cet exemple, les solutions A, B, C, D reçoivent le rang 1 car elles ne sont dominées par aucune

solution. Les solutions E, F et G ont le rang 2 car elles ne sont dominées que par des solutions de rang 1. Enfin, H est affectée au rang 3. NSGA II utilise aussi une seconde mesure, la distance de *crowding*, visant à améliorer la diversité de la population dans l'espace des objectifs. NSGA II n'est qu'un exemple parmi beaucoup employant une approche Pareto [30].

1.4.4 Approches basées sur des indicateurs

Plus récemment, des approches [52, 81, 145] utilisant des indicateurs de performance comme l'hypervolume pour donner un score à des solutions ont été proposées. Le principe est de définir le problème multi-objectif comme un problème visant à maximiser la valeur de l'indicateur associé à l'approximation. Cela diffère des approches standard qui cherchent plus à minimiser la distance à l'ensemble efficace et à maximiser la diversité. D'autre part, cela se fait par des comparaisons "locales" : on affecte un ou deux scores (intensification et diversification) à une solution et non pas à l'approximation globalement obtenue. Ainsi par exemple, si on considère deux solutions potentiellement efficaces, NSGA II leur donnera le même score en termes d'intensification et aura besoin d'un second indicateur (la distance de crowding) pour les départager en tentant de prendre en compte la diversité. Ce qui peut paraître artificiel. Dans une approche basée sur les indicateurs, ces deux aspects sont pris en compte au sein de l'indicateur de performance. On pourra par exemple plutôt considérer la contribution de chaque solution à la valeur de l'hypervolume obtenu par l'approximation selon que l'on choisit l'une ou l'autre solution. C'est ainsi que travaille par exemple Indicator Based Evolutionary Algorithm (IBEA) [145]. Il faut d'abord définir un indicateur de performance binaire $I : 2^{|O|} \times 2^{|O|} \to \mathbb{R}$. Cet indicateur mesure ce qui est perdu en termes d'une mesure donnée pour un ensemble de solutions par rapport à un autre ensemble de solutions. Si on utilise par exemple l'hypervolume, on obtient en utilisant les notations de [145] :

$$I_{HD} = \begin{cases} I_H(B) - I_H(A) & \text{si } \forall x^2 \in B, \exists x^1 \in A : x^1 \preceq x^2 \\ I_H(A+B) - I_H(A) & \text{sinon,} \end{cases} \tag{1.2}$$

où $I_H(A)$ est l'hypervolume de l'ensemble A et $A + B$ représente l'ensemble obtenu en prenant l'union de A et B et en ne gardant que les solutions non-dominées. Cet indicateur est illustré dans la figure 1.8 pour deux ensembles composés chacun d'une seule solution.

Dans cette méthode, le fitness $F(x^1)$ d'une solution x^1 par rapport à une solution P est calculé de la manière suivante :

$$F(x^1) = \sum_{x^2 \in P \setminus \{x^1\}} -e^{-I(\{x^2\},\{x^1\})/\kappa}. \tag{1.3}$$

Cette formule est dérivée de la formule suivante dans un but de minimiser l'influence des solutions qui ne sont pas potentiellement dominées :

$$F'(x^1) = \sum_{x^2 \in P \setminus \{x^1\})} I(\{x^2\},\{x^1\}) \tag{1.4}$$

qui vise à donner le plus grand score possible à une solution selon sa contribution par rapport aux autres solutions de la population.

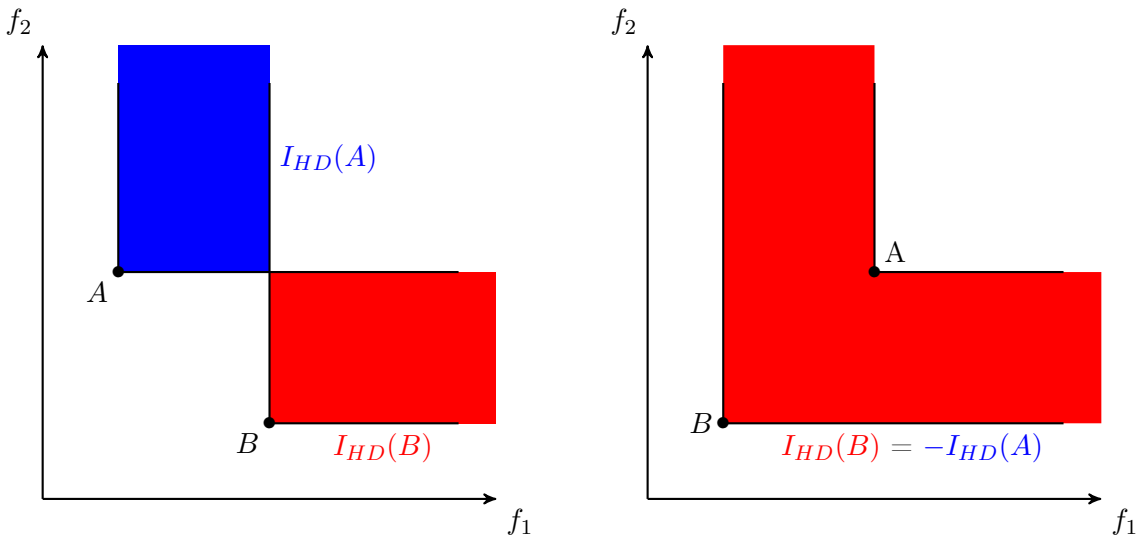


Fig. 1.8 – Illustration de l'indicateur I_{HD} (issue de [145]).

1.5 Méthodes d'optimisation multi-objectif

Nous avons discuté jusqu'à présent de la manière d'aborder la présence d'objectifs multiples sans faire de supposition sur les méthodes d'optimisation employées (sauf dans le cas des approches Pareto qui sont fortement liées aux algorithmes évolutionnistes ; ce qui est aussi le cas dans une moindre mesure des approches basées sur des indicateurs). Dans cette partie, nous commencerons par discuter de la notion de bornes primales et duales dans le cadre de l'optimisation multi-objectif. Puis, nous exposerons quelques méthodes approchées et exactes génériques. Le but ici n'est pas de présenter les méthodes *ad-hoc* pour un problème donné, ni même de faire une liste exhaustive de toutes les méthodes.

1.5.1 Bornes supérieures et inférieures

Nous commencerons cette partie par une remarque. Parler de bornes inférieures ou supérieures dans le cadre d'un problème multi-objectif n'a pas vraiment de "sens". En effet, si on a un objectif qui est une minimisation et l'autre une maximisation, cette notion d'inférieur et de supérieur devient "abstraite". Dans ce document, plutôt que de parler de borne primate et duale cependant, et en accord avec le fait que nous avons défini le problème général comme la minimisation du vecteur objectif, à moins d'une indication contraire, la borne inférieure fait référence à une relaxation et la borne supérieure à un ensemble réalisable. De ce fait, une borne supérieure n'est qu'un ensemble potentiellement non-dominé, c'est-à-dire un ensemble de points dans l'espace des solutions qui ne se dominent pas deux à deux et qui correspondent à au moins une solution réalisable.

Une première borne inférieure est le point *idéal* défini par $y^I = (y_1^I, y_2^I, \ldots, y_n^I)$ avec $y_k^I = \min_{x \in D} f_k(x)$. De manière évidente, il s'agit bien ici d'une borne inférieure puisque toute solution est nécessairement dominée (au pire faiblement pour les points extrêmes) par ce point. Il est possible aussi de définir une relaxation $\bar{y}^I$ de ce point en définissant sa coordonnée pour l'objectif

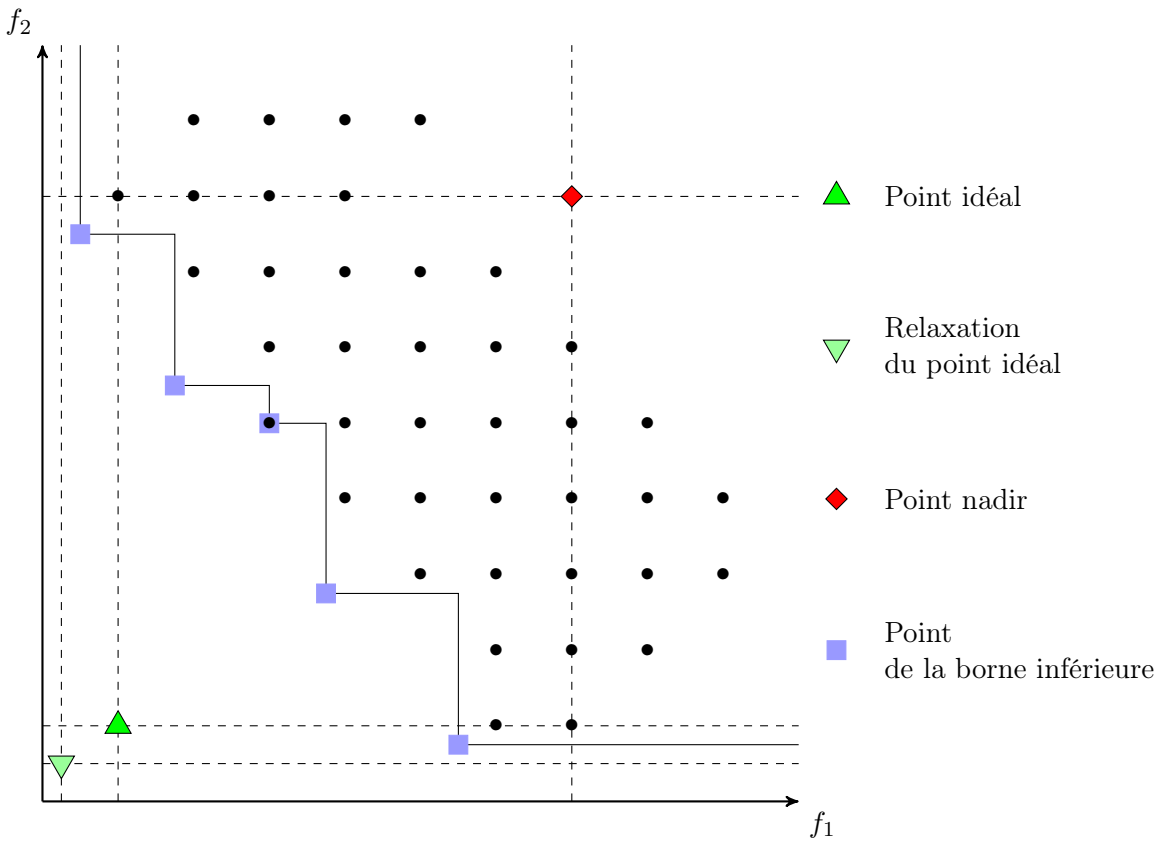


Fig. 1.9 – Point idéal et relaxation, point nadir, borne inférieure.

k comme une borne inférieure sur la fonction objectif f_k. On associe souvent au point idéal le point *nadir* $y^N = (y_1^N, y_2^N), \ldots, y_n^N$ où $y_k^N = \max_{x \in \mathcal{P}} f(x)$ avec $\mathcal{P}$ l'ensemble efficace. Il est à remarquer que le point nadir n'est pas une borne "supérieure" pour $\mathcal{Y}$ car il peut exister des points atteignables dominés par ce point. En fait, ces deux points délimitent la zone de l'espace des objectifs dans laquelle l'ensemble non-dominé se trouve. Ces concepts sont illustrés dans la figure 1.9.

La figure 1.9 illustre l'utilisation de la relaxation du point idéal comme borne inférieure. Il apparaît de cet exemple que le point idéal ou sa relaxation est une piètre borne inférieure car elle peut être très loin d'une solution réalisable et ne fournir que peu d'information sur l'ensemble non-dominé. Il est donc nécessaire d'affiner cette définition de borne inférieure. Une manière de le faire est de ne pas la définir comme un unique point de l'espace des objectifs mais comme un ensemble de points de l'espace des objectifs tel que toute solution réalisable est dominée ou correspond à un de ces points. Ce principe a été introduit par Villarreal et Karwan [137] et c'est celui que nous exploiterons par la suite dans les parties dédiées à l'utilisation d'algorithmes de séparations et coupes et de génération de colonnes.

Il est possible de définir d'autres bornes inférieures. Sourd et Spanjaard [123] définissent la borne inférieure comme une fonction qui sépare l'espace des objectifs en deux de telle sorte que toutes les solutions réalisables se trouvent dans un des deux hyperplans ainsi définis. Du fait qu'il

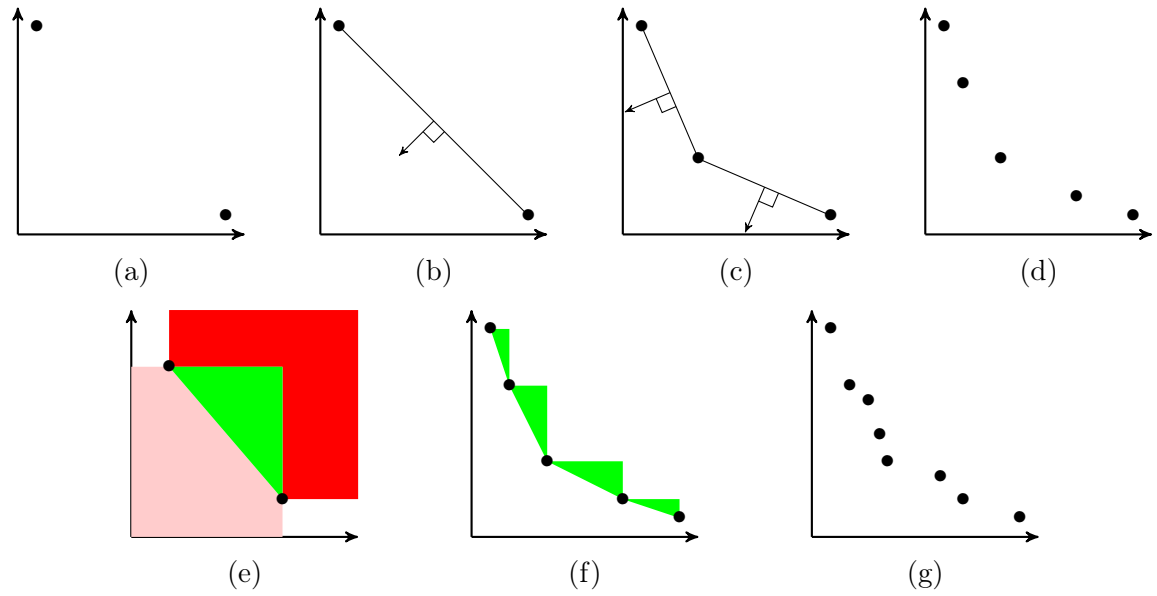


Fig. 1.10 – Illustration de la méthode en deux phases.

n'est pas possible en général de calculer exactement cette fonction, ils l'approximent en calculant des segments via des agrégations.

Dans ce document, nous nous intéressons au calcul de bornes inférieures pour des problèmes linéaires en nombres entiers via l'utilisation de l'algorithme de génération de colonnes. Ehrgott et Tind [47] proposent une première analyse de l'application de la génération de colonnes à des problèmes multi-objectif. Par la suite, si on trouve des applications de génération de colonnes à des problèmes multi-objectif, il s'agit d'applications directes de la méthode standard à des problèmes obtenus par des méthodes de scalarisation comme par exemple dans [77, 118]. Nous nous intéresserons aussi à la proposition de mécanismes qui prennent en compte la nature multi-objectif du problème directement au sein de la méthode de génération de colonnes, notamment en proposant des stratégies de recherche de solutions du sous-problème de coût réduit négatif.

1.5.2 Méthodes exactes

L'une des premières références en méthodes exactes pour l'optimisation multi-objectif est la méthode en deux phases d'Ulungu et Teghem [134]. Dans un premier temps, cette méthode pour les problèmes bi-objectif résout une succession de problèmes mono-objectif obtenus en effectuant une agrégation linéaire des deux objectifs. La méthode commence par trouver les points extrêmes (étape (a) dans la figure 1.10), puis fixe les poids pour définir une direction de recherche par rapport au plan formé par ces deux points (étape (b) dans la figure 1.10). La méthode est ensuite itérée (étape (c) dans la figure 1.10) via une recherche dichotomique jusqu'à ce qu'aucun nouveau point ne soit trouvé. On obtient alors toutes les solutions Pareto optimales supportées (étape (d) dans la figure 1.10). La méthode ne fait aucune hypothèse a priori sur la méthode utilisée pour résoudre les problèmes mono-objectif. Dans le cas de l'étude d'Ulungu et Teghem, le problème considéré était le problème d'affectation bi-objectif. L'agrégation résultait donc en

un problème d'affectation qui est polynomial. Le but de la seconde phase est de rechercher les solutions non-supportées via une énumération. La recherche est accélérée par l'utilisation des solutions trouvées lors de la première phase pour définir des "zones de recherche". Si on considère deux solutions supportées adjacentes, elles définissent 3 zones comme le montre le cas (e) de la figure 1.10. La zone verte est celle susceptible de contenir des solutions efficaces non-supportées, la zone rouge est dominée par les deux solutions efficaces et peut donc être évitée et la zone rose ne peut pas contenir de solution réalisable. Si on considère l'ensemble des solutions supportées trouvées, on peut limiter la recherche à la zone verte du cas (f) de la figure 1.10. A la fin de la seconde phase, l'ensemble des points non-dominés est trouvé (étape (g) dans la figure 1.10). Des améliorations de cette méthode, comme une extension à des problèmes avec trois objectifs, ont par la suite été proposées [46, 110, 111, 112]. D'après Ehrgott et Gandibleux [44], la plupart des méthodes exactes multi-objectif utilisent comme la méthode deux phases des sommes pondérées.

D'après la même étude, la seconde approche la plus populaire pour l'application de méthodes exactes est la méthode ϵ-contrainte. Sur un problème bi-objectif, le problème est de borner au départ un objectif par une valeur ϵ_0 qui est respectée par toutes les solutions. On obtient alors une solution Pareto optimale. On borne alors l'objectif par ϵ_1 qui est la valeur pour l'objectif directement inférieure à celle de la solution trouvée. On procède ainsi itérativement en fixant à l'itération i la valeur ϵ_i de telle manière à exclure la solution Pareto optimale trouvée à l'étape précédente sans exclure d'autres solutions Pareto optimales non trouvées. L'approche est illustrée dans la figure 1.11. Par exemple, une application à un problème de tournée couvrante a été faite dans [75]. Bérubé et al. [15] ont proposé une généralisation pour les problèmes à deux objectifs. Ils l'ont notamment appliquée à un problème de voyageur de commerce avec profits en se basant sur un algorithme de séparations et coupes. Laumanns et al [87] ont étudié une manière d'appliquer des méthodes ϵ-contraintes à des problèmes comportant plus de deux objectifs. D'autres méthodes itératives existent comme Parallel Partitioning Method (PPM) [93, 41] qui peut être vue comme une combinaison de la méthode deux phases et de la méthode ϵ-contrainte.

Le point commun entre ces approches est qu'elles ne sont que des manières d'itérer intelligemment des méthodes exactes en évitant la redondance de calcul, en mettant en commun des éléments de recherche ou en identifiant les éléments qui peuvent être effectués en parallèle. C'est un héritage de la méthode deux phases. Or, dans l'approche originale, il s'agissait d'itérer une résolution d'un problème d'affectation donc polynomiale. Cette tactique n'est pas nécessairement adaptée lorsqu'il s'agit d'itérer des méthodes sur les problèmes obtenus lorsque ces derniers sont NP-difficiles. Il est donc intéressant de préférer des méthodes qui travaillent sur l'ensemble du front Pareto à tout moment de la recherche. Il existe aussi des méthodes exactes qui ne sont pas la génération successive des solutions efficaces, mais qui retournent à la fin de leur exécution l'ensemble Pareto optimal. On peut penser à des méthodes de programmation dynamique [25], notamment en recherche de plus court chemin multi-objectif [98, 21]. Dans le cadre des plus courts chemins, l'idée est de modifier le principe d'optimalité de la programmation dynamique en un principe d'optimalité "faible" qui dit qu'un chemin optimal doit être composé de sous-chemins qui peuvent faire partie d'un chemin optimal [25, 66]. Cela se fait via une modification de la gestion des labels ; les différents objectifs formant des labels eux-mêmes. Dans ce document, nous allons nous intéresser plus particulièrement à des méthodes de recherche arborescente. Les premières utilisations d'algorithmes de séparations et évaluations sont celles de Botran et Riviera [20] et Kiziltan et Yucaoglu [78]. D'autre part, l'utilisation d'un algorithme de séparations et coupes comme méthode d'énumération pour la seconde phase de la méthode deux phases a été

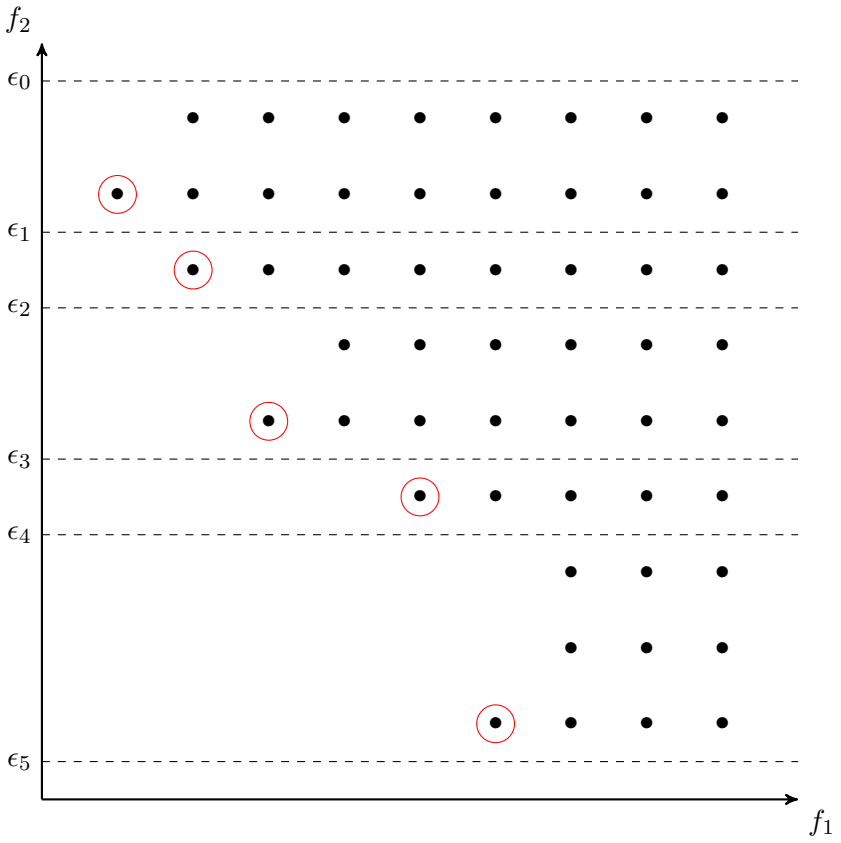


Fig. 1.11 – Illustration de l'utilisation de la méthode ϵ-contrainte.

explorée par Visée et al. [139].

Plus récemment, dans le cadre de l'étude de problèmes linéaires en nombres entiers purs, Ehrgott et Gandibleux [45] ont exploré le rôle de l'exclusion de sous-problèmes dans les algorithmes de séparations et évaluations dans un contexte bi-objectif. Sourd et Spanjaard [123] ont proposé un algorithme de séparations et évaluations qui calcule une approximation d'une fonction qui sépare l'espace des objectifs entre la zone contenant des points réalisables et celle n'en contenant pas. Cette approximation se calcule en résolvant des agrégations du problème considéré et les auteurs supposent donc que le problème agrégé est polynomial comme c'est le cas du problème d'arbre couvrant bi-objectif étudié. Dans le cadre de la programmation linéaire en nombres entiers mixtes, Mavrotas et Diakoulaki [99] ont proposé une méthode utilisant une borne inférieure relativement simple composée du point idéal. Ils l'ont par la suite modifiée pour affiner le calcul de la borne inférieure et accélérer la procédure [100]. Cette méthode a été révisée et améliorée ensuite au niveau de la représentation de l'ensemble efficace et des stratégies de branchement par Vincent et al. [138]. D'autres références sur le sujet peuvent être trouvées dans [4] et [106].

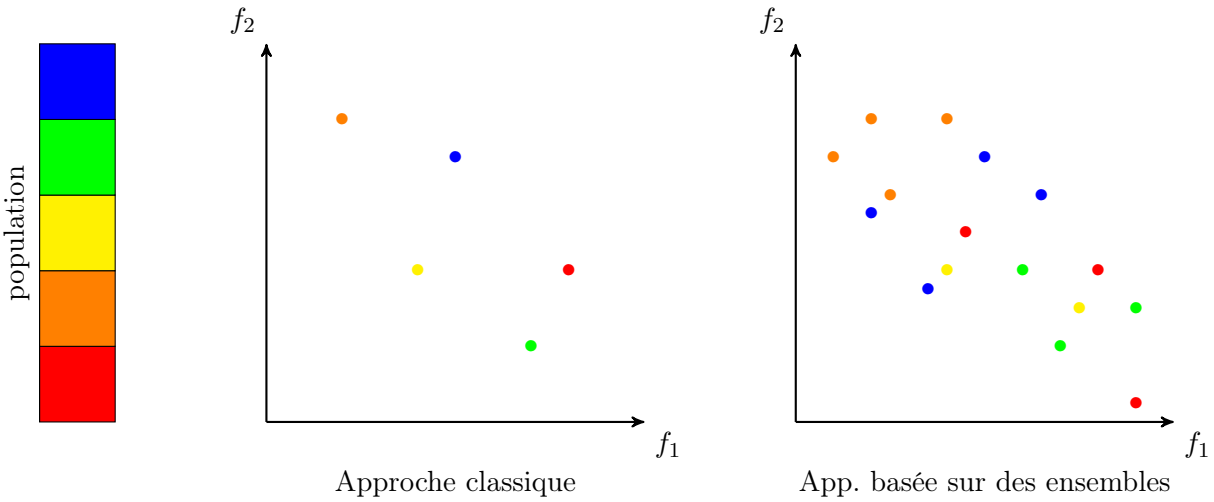


Fig. 1.12 – Différences entre les approches classiques et l'optimisation basée sur des ensembles.

1.5.3 Méthodes heuristiques

Si l'état de l'art des méthodes exactes pour l'optimisation multi-objectif est assez limité, cela n'est clairement pas le cas des méthodes heuristiques. Parmi elles, on trouve des méthodes classiques appliquées à des problèmes modifiés via des approches scalaires par exemple ou des heuristiques ad-hoc. Il y a aussi une vaste littérature de métaheuristiques *génériques* pour les problèmes multi-objectif. Au premier rang de celles-ci, on trouve les algorithmes évolutionnistes multi-objectif [36] basés sur les approches Pareto qui ont déjà été évoquées précédemment. Différentes notions ont été définies au fur et à mesure pour améliorer ces méthodes comme la notion d'élitisme visant à favoriser les solutions non-dominées pour accélérer la recherche [88, 146] ou des mécanismes pour l'optimisation dans un cadre "robuste" [37]. D'autres avancées se trouvent dans l'utilisation de nouveaux mécanismes pour classer les solutions comme les approches basées sur des indicateurs. Les méthodes évolutionnistes basées sur les populations dominent le domaine des métaheuristiques multi-objectif et les méthodes basées sur des notions de voisinage sont moins nombreuses. Dans ce cadre, on peut citer en exemple Target Aiming Pareto Search (TAPaS) [73] ou Indicator Based Multiobjective Local Search (IBMOLS) [10]. Une autre notion émergente est l'optimisation basée sur des ensembles [147]. Le principe est qu'il est difficile de donner un score à une solution dans un contexte multi-objectif car cela dépend des autres solutions pour pouvoir évaluer l'intensification et la diversification. Dans l'optimisation basée sur des ensembles, la méthode ne manipule pas des solutions, par exemple un croisement entre deux solutions, mais des ensembles de solutions qui sont considérés comme des entités. Cette notion appliquée à une population d'un algorithme génétique est illustrée dans la figure 1.12.

Chapitre 2

Contributions méthodologiques

Ce chapitre est dédié aux méthodes pour l'optimisation combinatoire multi-objectif. Dans les sections 2.1 et 2.2, nous expliquerons le type de méthodes que l'on souhaite développer et la stratégie qui nous semble pertinente dans le cadre de l'étude d'un problème multi-objectif. Puis, trois méthodes correspondantes aux trois phases proposées seront présentées. La section 2.3 porte sur une famille d'algorithmes génétiques étendant les algorithmes génétiques à clefs aléatoires aux problèmes multi-objectif via l'introduction de décodeurs multi-objectif. La proposition d'un algorithme générique de séparations et coupes répondant aux critères que nous avons fixés est abordée dans la section 2.4. Enfin, la section 2.5 porte sur la proposition de mécanismes pour l'algorithme de génération de colonnes pour une classe de problèmes bi-objectif.

2.1 Méthodes *anytime* multi-objectif

2.1.1 Principe

Le but dans ce manuscrit est de proposer des méthodes multi-objectif *anytime*. Le terme *anytime* fait référence au principe en intelligence artificielle introduit par Dean et Boddy [35] qui stipule qu'il faut être capable d'arrêter une méthode à n'importe quel instant et que la méthode doit être en mesure de retourner une réponse. Dans un contexte multi-objectif, il va de soi que la méthode doit répondre à certains critères. Les notions d'intensification et diversification présentées dans le chapitre précédent ne reflètent pas complètement ce que l'on recherche. Il s'agit en effet de qualité du résultat attendu à la fin de la recherche. Ici, ce qui est recherché est le comportement de la méthode tout au long de la recherche. Retourner une unique solution n'est par exemple pas envisageable, il faut renvoyer un ensemble potentiellement non-dominé qui répond à deux principes expliqués ci-dessous de représentativité (qui est liée à l'intensification) et d'uniformité (liée à la diversification). Pour atteindre ces buts, il est nécessaire de définir des algorithmes qui influent sur l'ensemble de l'approximation à tout instant. Un exemple type est celui des algorithmes évolutionnistes multi-objectif qui atteignent ce résultat via l'utilisation d'une population. Il y a cependant un troisième point qui doit être pris en compte et qui est un critère d'efficacité. Il est moins évident à mettre en place dans le cadre des algorithmes évolutionnistes du fait de leur nature stochastique. Cependant, cette efficacité est plus facile à prendre en compte via des approches scalaires où on dispose d'un plus grand pouvoir de guidage des recherches. Dans les méthodes proposées dans ce document, cela se fait en couplant ces méthodes scalaires à des

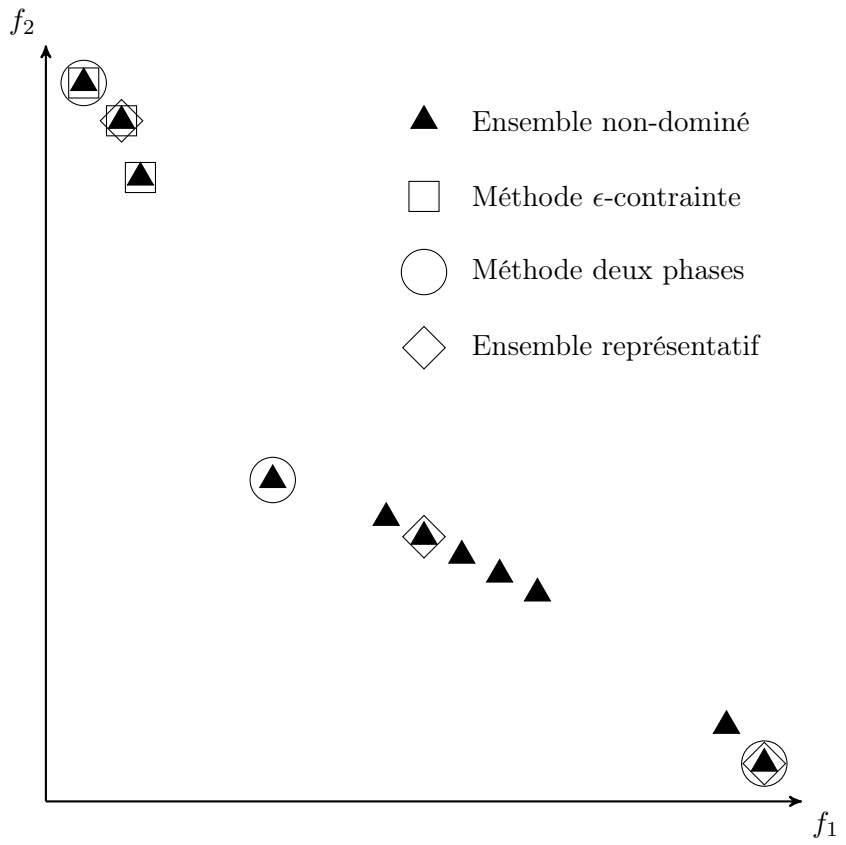


Fig. 2.1 – Illustration de la représentativité.

techniques de programmation mathématique. Cependant, chaque méthode a ses avantages et ses défauts et elles peuvent être employées à plusieurs étapes de la recherche. C'est ce qui est exposé dans la section 2.2. Avant cela, nous allons illustrer les différents critères exposés en montrant ce qui est souhaitable et ce qui ne l'est pas vis-à-vis de certaines approches.

2.1.2 Représentativité

Lorsque l'on arrête la méthode, il faut avoir une information la plus représentative de l'ensemble Pareto optimal. Il faut par exemple avoir une idée de l'étendue entre les points extrêmes, des zones de l'espace où il y a des solutions dans le cas où le front est constitué de plusieurs groupes clairement distincts de points. Ceci est lié à la notion de diversification exposée dans le chapitre précédent. Ce principe est illustré dans la figure 2.1. Les triangles représentent les points non-dominés. Supposons qu'il faille une unité de temps pour générer un de ces points et que l'on dispose de trois unités de temps pour appliquer une méthode. Nous allons regarder le résultat de deux méthodes classiques : l'approche ϵ-contrainte et la recherche dichotomique de la première phase de la méthode deux phases. La méthode ϵ-contrainte renverrait les solutions correspondant aux carrés. Il est clair que cela n'est pas intéressant dans une étude multi-objectif vu que l'on ne dispose d'aucune information sur l'étendue de l'ensemble efficace et que l'on ignore toute

information sur les solutions au "centre" qui devraient être celles qui sont les plus intéressantes pour le décideur (sinon il y a une préférence claire pour un des objectifs). Dans le cas de la méthode deux phases, les solutions trouvées seraient celles représentées par un triangle dans la figure 2.1. Si on a bien ici une idée de l'étendue de l'ensemble Pareto optimal, la solution représentant l'ensemble des solutions du centre n'est pas nécessairement la plus pertinente. Ce type de recherche dichotomique permet cependant clairement d'avoir une idée plus précise de la nature de l'ensemble efficace qu'une itération comme la méthode ϵ-contrainte. Cependant, nous verrons qu'elle ne permet pas nécessairement de répondre au second critère, l'uniformité de convergence. Un ensemble représentatif serait celui représenté par des losanges dans la figure. C'est en quelque sorte, le résultat que l'on obtiendrait si on appliquait un algorithme de clustering sur l'ensemble efficace pour ne garder que trois solutions.

2.1.3 Uniformité

Le second point auquel doit répondre une méthode est l'uniformité de convergence. De par la nature des objectifs définissant le problème, il se peut que l'un des objectifs soit plus simple à résoudre que les autres. Considérons par exemple le problème du voyageur de commerce avec profits [49]. Le problème est défini sur un graphe non-orienté valué où chaque sommet est associé à un profit. Il n'est pas nécessaire de visiter tous les sommets mais il faut qu'un sommet soit visité par la tournée pour engendrer le profit qui lui est associé. Une solution au problème est donc un cycle Hamiltonien sur un sous-ensemble de sommets commençant et terminant en un sommet particulier représentant un dépôt. Ainsi, on définit naturellement deux objectifs contradictoires : i) minimiser la longueur de la tournée ; ii) maximiser le profit engendré par la tournée. Considéré isolément, le premier objectif est trivial puisqu'une solution optimale immédiate est celle constituée uniquement du dépôt et qui est donc de coût nul et de profit nul. Le second objectif est plus complexe puisqu'il s'agit de résoudre un problème de voyageur de commerce sur l'ensemble des sommets, le profit étant maximum lorsque tous les sommets sont visités. D'autre part, les solutions proches des extrémités du front sont plus faciles à obtenir puisque d'un côté il s'agit de visiter un faible nombre de sommets et de l'autre résoudre des problèmes de voyageur de commerce sur des ensembles proches de l'ensemble complet de sommets. Les solutions de plus fort compromis représentent alors un plus grand challenge puisque la combinatoire est plus élevée. Il s'agit en effet d'une espèce de problème de bin packing combiné pour chaque sac possible à la combinatoire d'un problème de voyageur de commerce.

Si on note f_1 le premier objectif et f_2 le second objectif et que l'on considère comment se comporteraient les approches d'agrégation et ϵ-contrainte si elles doivent résoudre à chaque fois le problème obtenu à l'optimalité. Dans le cadre d'une agrégation couplée à une recherche dichotomique, il faut d'abord optimiser $\min f_1 - 0f_2$ qui correspond donc à un problème du voyageur de commerce qui est NP-difficile donc, en général, sans garantie sur la vitesse de convergence. Ensuite, la méthode résoudra le problème optimisant $\min 0f_1 + f_2$ qui est trivial. Puis, la méthode résoudra le problème minimisant $\lambda f_1 + (1 - \lambda) f_2$ en fonction de la direction fournie par les deux points extrêmes. On obtient alors un problème de tournée profitable [40] dont la combinatoire est plus élevée que celle d'un problème de voyageur de commerce ; ce dernier étant un cas particulier du problème de tournée profitable. Puis lors de la recherche dichotomique, il y aura un déséquilibre entre la recherche "à gauche" plus simple que celle "à droite". Ainsi dans un temps limité, selon l'ordre dans lequel la recherche est effectuée, on aura soit une bonne

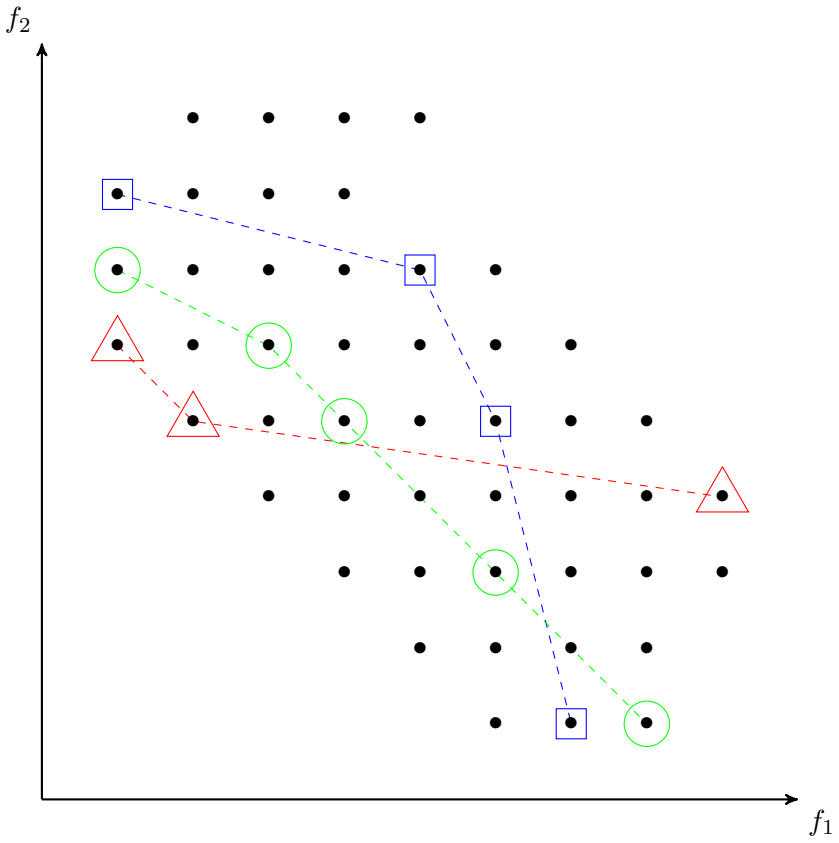


Fig. 2.2 – Uniformité de la convergence.

convergence sur des solutions faciles à obtenir soit un "enlisement" sur des cas plus complexes et nécessitant plus de calculs. Pour la méthode ϵ-contrainte, le déséquilibre est encore plus flagrant. En effet, si on contraint le second objectif, la méthode convergera très vite sur les premières solutions avant de "ralentir" sur les solutions de fort compromis qui correspondront au problème du voyageur de commerce avec collecte de prix [8]. Si on contraint le premier objectif, la méthode risque de trouver encore moins de solutions puisque les premiers problèmes résolus seront alors un problème de voyageur sélectif [76, 86] (à l'exception de la première solution qui correspond à un problème du voyageur de commerce).

Dans la figure 2.2, la méthode retournant les solutions représentées par des triangles rouges n'est pas intéressante car elle ne converge que sur un seul objectif sans prendre en compte le second objectif. La méthode retournant les solutions représentées par des carrés bleus n'est pas non plus adéquate car elle se focalise sur les solutions extrêmes au détriment des solutions de compromis. La dernière méthode garde une progression uniforme au détriment d'une convergence un peu moins poussée que les autres sur certaines zones de l'espace des objectifs et est préférable aux deux autres méthodes.

Il est donc intéressant de définir des méthodes qui font converger l'ensemble du front de manière uniforme en rendant cela le plus transparent et naturel. Dans ce document, nous exposerons deux méthodes qui se veulent uniformes dans la convergence. La première est un algorithme de

séparations et coupes qui garantit l'uniformité en calculant une relaxation du front à chaque étape via un algorithme qui est polynomial ou pseudo-polynomial. La seconde méthode est basée sur la génération de colonnes avec la proposition de stratégies pour la recherche de colonnes qui influent sur l'ensemble des solutions constituant l'approximation.

2.1.4 Efficacité

La dernière chose à considérer lors du développement d'une méthode est un souci d'efficacité. Il s'agit d'éviter de faire des calculs inutiles. On peut considérer comme des calculs inutiles l'exploration de solutions qui seront nécessairement dominées par des solutions déjà trouvées ou l'exploration multiple d'un même ensemble de solutions. On peut trouver dans la littérature plusieurs stratégies possibles pour aller dans ce sens. Par exemple, dans Target Aiming Pareto Search [73], l'espace des objectifs est divisé en zones par rapport à un ensemble potentiellement non-dominé pour définir des buts guidant des recherches locales pour qu'elles explorent des zones différentes. Bérubé et al. [15] itèrent via une méthode ϵ-contrainte un algorithme de séparations et coupes mais ils réutilisent les coupes déjà générées et les solutions déjà trouvées pour accélérer la recherche. Enfin, la seconde phase de la méthode en deux phases réutilise l'information de la première phase pour ne pas explorer des solutions déjà trouvées. Cependant, il s'agit d'itération de méthodes, qui si elles exploitent des résultats obtenus initialement, repartent plus ou moins de zéro à chaque fois et peuvent explorer tout de même des solutions communes.

Une stratégie plus efficace serait de factoriser la recherche et de ne pas repartir à zéro à chaque fois. C'est l'avantage de méthodes comme les algorithmes évolutionnistes, qui via l'utilisation d'une population, peuvent influer sur les directions de la recherche en fonction des solutions trouvées et des solutions actuellement dans la population. Pour illustrer ce que nous essayons de faire, considérons le problème dont l'ensemble des solutions est donné dans la figure 2.3. On suppose que l'on dispose d'une recherche locale pour résoudre le problème. Le voisinage d'une solution est les huit solutions (au maximum) qui l'entourent directement. La méthode démarre au point S. Si on applique une recherche dichotomique comme celle de la première phase de la méthode en deux phases, il lui faudra 7 itérations pour trouver le point A. Puis repartant du point S, il lui faudra 7 itérations pour trouver le point B et enfin 6 itérations pour trouver le point C. Il faut donc un total de 20 itérations pour trouver l'ensemble non-dominé. Une meilleure stratégie consisterait à ne faire qu'une seule recherche depuis S vers le point D. Puis à partir du point D, de lancer 3 recherches pour obtenir les points A, B et C. Le nombre d'itérations étant alors de 5 pour trouver le point D (chemin bleu dans la figure), 2 pour le point A (chemin rouge), 2 pour le point B (chemin vert) et 1 itération pour le point C (chemin orange), soit un total de 10 itérations. Ainsi la seconde méthode sera terminée alors que la première méthode n'aura pas encore trouvé les deux points extrêmes. Dans les méthodes de séparations et coupes et de génération de colonnes proposées dans ce document, ce but est atteint en travaillant à chaque itération sur le même modèle mathématique pour la génération de plusieurs solutions et en prenant des décisions qui influent le plus sur le modèle en fonction des solutions qui sont générées.

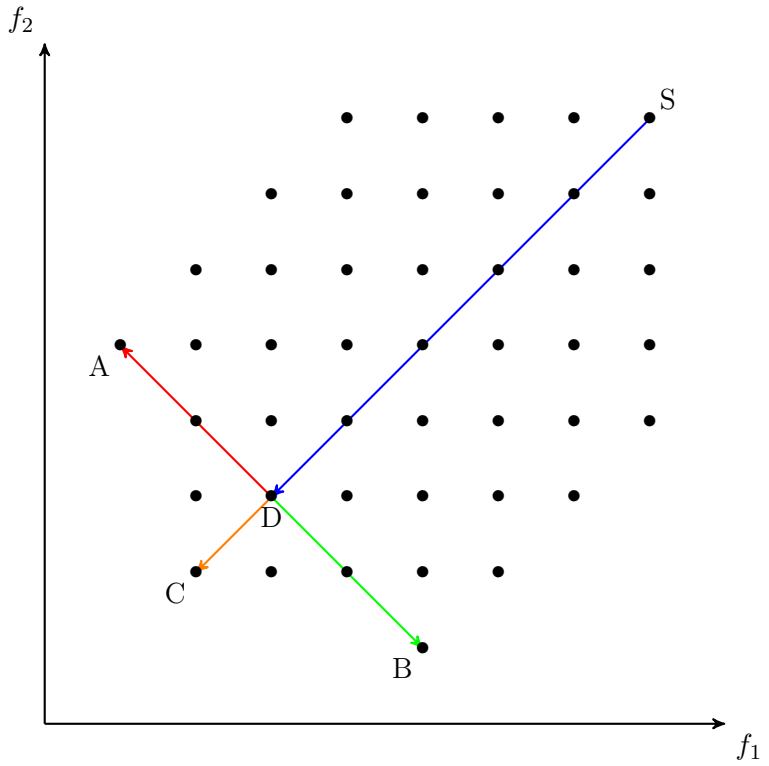


Fig. 2.3 – Efficacité.

2.2 Une étude en trois phases

Pour un problème d'optimisation standard, une approche habituelle est de définir une borne inférieure, d'utiliser celle-ci pour définir une méthode exacte et, enfin, développer des méthodes heuristiques pour aborder des problèmes de grande taille. Dans le cadre multi-objectif, on peut être tenté d'employer une approche inverse. Commencer d'abord par développer des méthodes heuristiques pour étudier le comportement des objectifs entre eux et limiter l'espace de recherche, c'est ce qui est appelé *exploration* ci-dessous. Puis, on peut utiliser des méthodes de recherche plus coûteuses et intensives pour obtenir l'ensemble non-dominé ou une très bonne approximation. Pour pallier aux coûts des méthodes employées, il est intéressant d'utiliser les informations fournies lors de la première phase pour limiter et guider la recherche. On nommera cette seconde phase *spécialisation*. Enfin, l'une des difficultés rencontrées en optimisation multi-objectifBRKGA est que les problèmes étudiés n'ont pas nécessairement beaucoup de résultats connus. Il n'est donc pas toujours évident de valider la qualité des résultats. Ainsi, dans une troisième phase de *validation*, il est intéressant d'utiliser des méthodes dédiées aux problèmes multi-objectif pour calculer des bornes inférieures de bonne qualité. Nous allons discuter plus en détail chacune des phases de cette approche, qui est illustrée dans la figure 2.4.

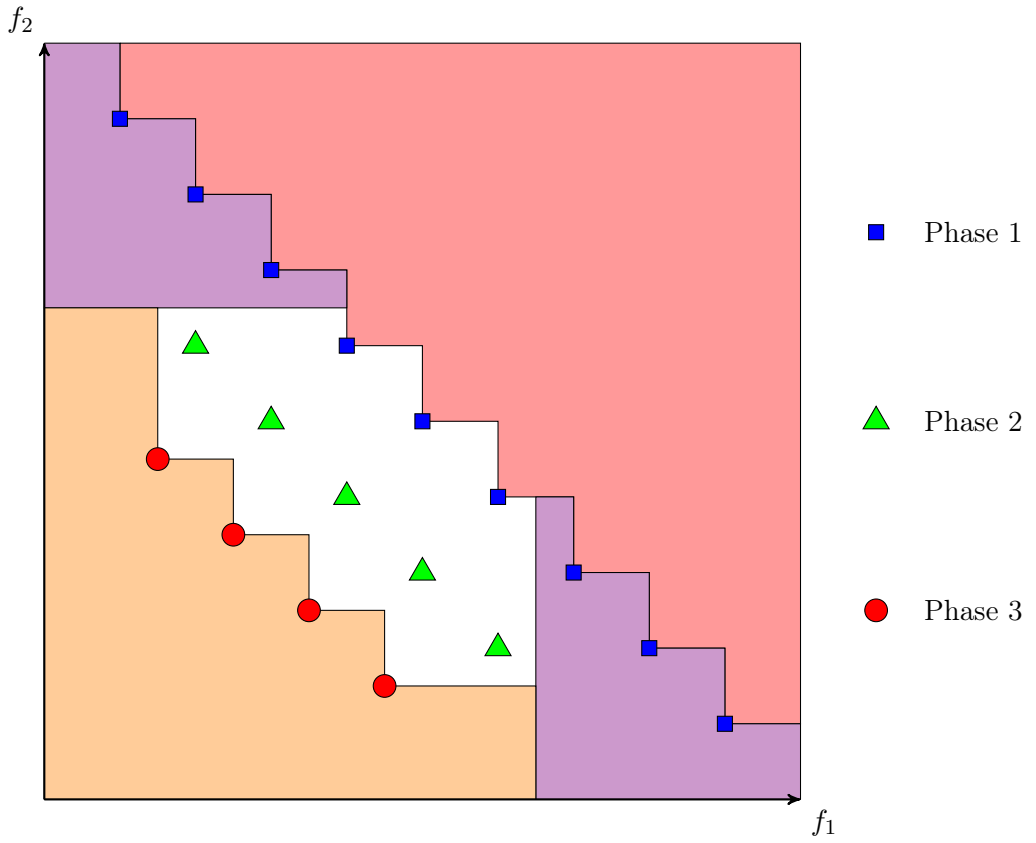


Fig. 2.4 – Une étude en trois phases.

2.2.1 Exploration

Dans cette phase, le but est de connaître l'intéraction entre les objectifs. Par exemple, sont-ils fortement conflictuels ou légèrement corrélés ? Un autre but est d'obtenir rapidement des bornes supérieures pour le problème. Il est donc nécessaire ici de définir des méthodes qui explorent de manière la plus diversifiée possible l'espace des objectifs. Elles doivent donc être capables de chercher parmi un grand nombre de solutions. Les algorithmes évolutionnistes multi-objectif sont bien adaptés à cette tâche. Ils demandent en effet peu de connaissances sur le problème ou l'intéraction entre les objectifs pour être définis. De plus, leur complexité en termes de coût d'exécution est assez peu sensible au nombre de solutions que contient l'approximation. Cependant, la définition d'opérateurs peut être compliquée si on ne veut pas favoriser un objectif par rapport à un autre. Des méthodes qui peuvent être utilisées dans cette phase et qui sont envisagées dans ce document sont les algorithmes génétiques à clefs aléatoires biaisés multi-objectif (2.3) et l'utilisation de la génération de colonnes (2.5) comme heuristique.

2.2.2 Spécialisation

Une méthode exploratrice et diversificatrice à la fois peut être difficile à définir. D'autre part, des méthodes de recherche intensive, notamment des recherches basées sur des énumérations implicites, peuvent ne pas être possibles à mettre en place si un trop grand nombre de solutions doit être envisagé. Ainsi le but est d'utiliser les résultats obtenus par la première méthode pour affiner l'espace de recherche. Si on considère dans la figure 2.4, les solutions représentées par des carrés bleus retournées durant la première phase, on peut définir plusieurs zones. Premièrement, ces solutions sont une borne supérieure, ce qui permet d'exclure la zone rouge dans une méthode de recherche arborescente par exemple. D'autre part, si le décideur est vraiment intéressé par les solutions de compromis, il est inutile de se focaliser sur les zones violettes qui correspondent peu ou prou à l'optimisation d'un objectif. L'exclusion de ces trois zones est alors pertinente. Selon le problème envisagé et la méthode utilisée dans la seconde phase, cela peut se faire via la modification des données, l'ajout de contraintes ou l'utilisation de mécanismes de dominance. Via la limitation de la recherche et la focalisation sur un sous-ensemble de l'ensemble efficace, on peut envisager l'utilisation de méthodes de recherche plus intensives et complexes. Un exemple proposé dans ce document est un algorithme de séparations et coupes (2.4) qui peut être utilisé comme méthode exacte ou comme heuristique en tronquant la recherche.

2.2.3 Validation

Il arrive souvent dans la littérature sur l'optimisation multi-objectif que, lorsqu'un problème est étudié, il n'y a pas de résultats exacts pour ce dernier. Il est donc uniquement possible de comparer des heuristiques entre elles. Voire que le problème n'ait jamais été envisagé auparavant. Il est alors difficile d'évaluer de manière absolue la qualité d'un résultat ou d'une méthode. Il faut donc mettre au point des méthodes pour calculer des bornes inférieures de bonne qualité. Du fait de la complexité du calcul d'une telle borne dans un espace multi-dimensionnel, on peut concevoir des méthodes relativement coûteuses, qui ne pourront pas nécessairement être exploitées dans des méthodes de recherche d'une borne supérieure. Il s'agit ici de définir des outils d'analyse et de validation en amont. Dans le présent document, on considérera le calcul de bornes inférieures qui sont des ensembles de points dans l'espace des objectifs tel que toute solution réalisable est dominée par un de ces points. Cette borne inférieure peut être calculée uniquement par rapport à la zone délimitée dans la phase de spécialisation. Ainsi dans la figure 2.4, la borne inférieure représentée par les disques rouges domine nécessairement toutes les solutions réalisables après avoir exclu les zones rouges et violettes mais pas nécessairement toutes les solutions réalisables du problème original. Dans cet exemple, les solutions extrêmes trouvées lors de la phase d'exploration ne sont pas dominées par la borne inférieure. Nous exposerons un mécanisme de calcul de bornes inférieures basé sur la génération de colonnes dans la section 2.5.

2.3 Algorithmes génétiques à clefs aléatoires biaisés multi-objectif

2.3.1 Problématique

Si on regarde la plupart des travaux portant sur l'application des algorithmes évolutionnaires sur des problèmes multi-objectif, les recherches se focalisent sur la définition de méthodes d'évaluation de qualité des solutions, de mécanismes de diversification ... Lorsque les méthodes

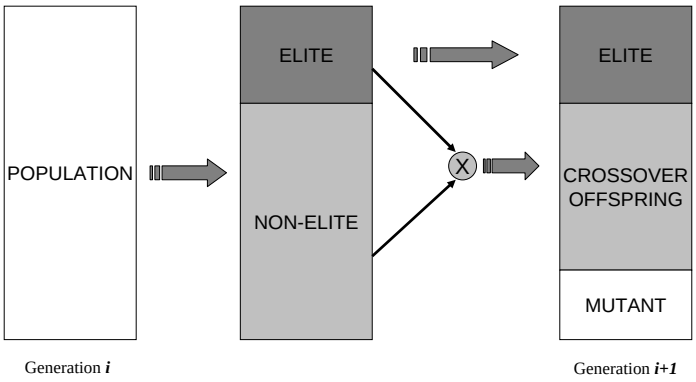


Fig. 2.5 – Gestion de la population dans BRKGA.

sont testées ou appliquées sur des problèmes difficiles ou complexes tel que des problèmes de tournées ou des problèmes d'ordonnancement, les opérateurs génétiques sont issus de la littérature standard. Or, ces opérateurs sont souvent définis pour optimiser un objectif particulier qui est alors favorisé par rapport aux autres. La solution visant à utiliser plusieurs opérateurs associés à différents objectifs est un peu moins mauvaise mais elle a tout de même tendance à favoriser la recherche vers les solutions extrêmes de l'ensemble efficace au détriment des solutions de plus fort compromis. La définition d'opérateurs de croisement qui prennent en compte plusieurs objectifs simultanément n'est pas simple. Pour répondre à cela, une adaptation des algorithmes génétiques à clefs aléatoires biaisés est proposée et son application à une large classe de problèmes de tournées de véhicules est présentée ci-dessous.

2.3.2 Algorithmes génétiques à clefs aléatoires

Les algorithmes génétiques à clefs aléatoires (RKGA) ont été introduits par Bean [12] et améliorés par Gonçalves et Resende dans le cadre des algorithmes génétiques à clefs aléatoires biaisés (RKGA) [59]. Ces méthodes ont été appliquées avec succès à de nombreux problèmes [59]. Ils ne travaillent pas directement sur une représentation spécifique à un problème mais sur une représentation générique. Un chromosome est un vecteur indexé sur un ensemble de clefs $\mathcal{K}$ de valeur dans $[0, 1]$. Par exemple, pour un problème de voyageur de commerce sur un ensemble de sommets $V = \{0, 1, 2, 3, 4, 5\}$, on peut poser $\mathcal{K} = V$. Ainsi une solution du problème aura la forme suivante :

$\mathcal{K}$	0	1	2	3	4	5
Val.	0.1	0.4	0.2	0.9	0.5	0.6

Il faut aussi définir un décodeur, c'est-à-dire une procédure de $[0, 1]^{|\mathcal{K}|} \to x \in \Omega$, qui à un vecteur associe une solution. Ce décodeur doit être déterministe : un vecteur-clef retourne toujours la même solution. Il s'agit du seul élément de la méthode spécifique au problème traité. Pour le problème du voyageur de commerce, on peut par exemple simplement visiter les sommets dans l'ordre croissant des clefs. Dans l'exemple ci-dessus, on obtient alors la solution $(0, 2, 1, 4, 5, 3, 0)$.

Une génération de l'algorithme est alors constituée des quatre phases suivantes illustrées dans la figure 2.5 :

1. **Décodage :** le décodeur est appliqué sur les vecteurs qui ont été générés lors de la précédente génération. Une partie des vecteurs est héritée de la précédente génération et il n'est donc pas nécessaire de les décoder à nouveau. Une partie de la population formée de solutions dites "élites" constituées des solutions ayant les meilleures qualités est directement reconduite dans la population de la génération suivante.
2. **Tri :** les solutions sont triées par rapport à leur qualité de la meilleure à la moins bonne. Une partie de la population formée de solutions dites "élites" constituées des solutions ayant les meilleures qualités est directement reconduite dans la population de la génération suivante.
3. **Mutation :** un pourcentage de la population, constitué des plus mauvaises solutions est remplacés par des solutions régénérées aléatoirement.
4. **Crossover :** le reste de la population est complétée en appliquant un opérateur de croisement entre deux solutions. Une des solutions est choisie aléatoirement parmi les solutions élites, l'autre parmi les solutions qui ne sont pas élites mais qui n'ont pas non plus été remplacées par la mutation. L'opérateur de croisement effectue un tirage aléatoire pour chaque clef. Selon la valeur tirée, le rejeton reçoit la valeur de la clef associée à l'un ou l'autre parent. Le tirage est biaisé pour favoriser le parent provenant de la partie élite de la population.

2.3.3 Application aux problèmes multi-objectif

Cette méthode répond bien au souhait de simplicité et de généralité des opérateurs génétiques vis-à-vis des objectifs. Pour l'adapter à un problème multi-objectif, il est nécessaire de se focaliser uniquement sur le décodeur qui est souvent un problème plus simple que le problème initial. Le principe de l'algorithme de base est de décoder un vecteur-clef d'une manière unique en une solution. Si on fait cela, on continue à influencer le décodage vers un objectif. En effet, pour une clef donnée, souvent les opérateurs essaient de générer une solution qui est optimisée par l'objectif, le décodeur étant alors une heuristique résolvant un cas particulier de l'instance.

Pour éviter cet écueil, nous utilisons ici une notion de décodeur multi-objectif. On n'associe plus à un vecteur-clef une solution unique mais un ensemble de solutions. Formellement, un décodeur est donc une fonction qui, à un vecteur-clef $v \in [0,1]^{|\mathcal{K}|}$, associe un ensemble de solutions $\mathcal{P} = \{x \in \Omega : \nexists x, y \in \mathcal{P}, x \preceq y\}$. Le mécanisme de décodage doit toujours être déterministe et ad-hoc. En cela, on peut considérer que l'on est dans une approche basée sur les ensembles. L'avantage est qu'ici, un ensemble est représenté de manière compacte par un vecteur de clefs qui est simple à manipuler. Le décodeur, à l'instar de la méthode mono-objectif, étant l'élément le plus complexe mais correspondant alors à un problème plus simple que le problème global. Dans le chapitre 5, ce principe est appliqué à un problème d'observation spatiale. Nous allons illustrer ci-dessous l'application de ce type de méthodes aux problèmes de tournées de véhicules multi-objectif.

2.3.4 Application aux problèmes de tournées de véhicules

Pour adapter la méthode aux problèmes de tournées multi-objectif, les choix à faire sont au nombre de deux : i) comment décoder un vecteur de clefs ; ii) comment attribuer un score à la solution obtenue du point de vue multi-objectif. Le décodage est effectué ainsi. Dans un

TABLE 2.1 – Analyse de la sensibilité pour le nombre de labels et d'itérations.

	# laE51-05ebels	1000 it.		2500 it.		5000 it.	
E51-05e	5	9.94e+05	32	9.97e+05	81	1.00e+06	160
E51-05e	10	1.00e+06	62	1.01e+06	157	1.01e+06	299
E51-05e	$+\infty$	1.01e+06	167	1.01e+06	423	1.01e+06	842
E76-10e	5	1.98e+06	66	1.98e+06	167	1.99e+06	338
E76-10e	10	1.98e+06	122	1.99e+06	305	1.99e+06	620
E101-08e	$+\infty$	2.00e+06	621	2.00e+06	1577	2.01e+06	3378
E101-08e	5	6.05e+06	206	6.06e+06	519	6.12e+06	1069
E101-08e	10	6.10e+06	402	6.11e+06	1011	6.13e+06	2009
E101-08e	$+\infty$	6.15e+06	3937	6.17e+06	9972	6.18e+06	19618
E121-07c	5	1.37e+07	345	1.38e+07	867	1.39e+07	1752
E121-07c	10	1.38e+07	667	1.38e+07	1678	1.39e+07	3394
E121-07c	$+\infty$	1.39e+07	10087	1.39e+07	26001	1.40e+07	53631
E101-10c	5	5.29e+06	163	5.30e+06	410	5.34e+06	816
E101-10c	10	5.31e+06	324	5.31e+06	813	5.33e+06	1618
E101-10c	$+\infty$	5.36e+06	3197	5.37e+06	8181	5.37e+06	16033

premier temps, on construit une tournée géante sur les sommets en suivant le décodeur décrit pour le problème du voyageur de commerce. Puis, on applique une procédure de *split* définie à l'origine par Prins pour le problème de tournées de véhicules avec capacité [109] et étendue par la suite à plusieurs variantes du problème de tournées de véhicules. Le découpage d'une tournée géante en sous-tournées revient donc à résoudre un problème de plus court chemin élémentaire multi-objectif avec contraintes de ressources. Il s'agit donc d'un problème NP-difficile dans le cas général mais il est possible d'adapter des algorithmes de programmation dynamique existants [50] ou d'utiliser des algorithmes de programmation dynamique pour certains cas particuliers [21]. D'autre part, le graphe obtenu par la procédure de *split* est un graphe dirigé acyclique qui a donc une structure particulière. Il est aussi possible de ne pas garder ou de pas étendre l'ensemble des labels associé à un sommet et qui peuvent potentiellement mener à une solution efficace. On peut donc limiter la taille de la liste des labels associés à un sommet en ne gardant ou en n'étendant que les sommets qui contribuent le plus à l'hypervolume. Le résultat du décodeur est l'ensemble des labels non-dominés associé au sommet représentant le dernier nœud visité dans le graphe dirigé acyclique. On obtient donc un ensemble de solutions potentiellement efficaces et on peut donc donner comme score au vecteur l'hypervolume de cet ensemble.

Cette stratégie a été implémentée pour le problème de tournées de véhicules cumulatif avec contrainte de capacité (CCVRP) en supprimant la contrainte imposant d'utiliser un nombre fixe de véhicules et en le remplaçant par un second objectif visant à minimiser le nombre de véhicules utilisés. Le tableau 2.1 reporte les valeurs de l'hypervolume pour les ensembles potentiellement efficaces obtenus en faisant varier le nombre d'itérations et le nombre de labels sur 4 instances classiques pour le problème de tournées de véhicules avec capacité. Il apparaît que limiter le nombre de labels a un impact moindre sur la qualité des résultats que le nombre de générations par exemple. A l'inverse, l'impact du nombre de labels est beaucoup plus important sur le temps

de calcul. Comme le but est ici de définir une méthode rapide et simple à mettre en oeuvre dans le cadre de la première étape d'un problème multi-objectif, cette concession est acceptable. Il est à noter qu'il s'agit là d'une implémentation préliminaire et que de nombreuses choses restent à optimiser. Cependant, les résultats sont encourageants et il est nécessaire de continuer à explorer cette voie.

En résumé, ce mécanisme permet de définir une méthode générique qui n'est pas biaisée par les objectifs. En effet, le seul élément spécifique au problème prend en compte l'ensemble des objectifs. Dans les faits, la méthode n'utilise que des mécanismes éprouvés et simples (ou relativement simples) à mettre en place : algorithmes génétiques avec vecteurs clefs, hypervolume, programmation dynamique pour les problèmes de plus court chemin ... La complexité de ce problème se retrouve d'ailleurs dans ce dernier aspect mais il s'agit d'un problème "plus simple" que le problème original et il est possible d'utiliser un grand nombre de résultats déjà existants. Ce travail est en cours et de nombreux points sont à explorer : différentes stratégies d'extension de labels (clustering, mécanismes d'accélération ...), des tests sur d'autres problèmes (en fait le CCVRP est un peu compliqué car il est toujours possible d'avoir autant de labels pour un sommet que de clients, ce qui le rend difficile) et des comparaisons avec d'autres approches comme NSGA II ou l'itération de méthodes mono-objectif.

2.4 Algorithme de séparations et coupes multi-objectif

2.4.1 Principe de l'approche

Le principe de cette approche est de définir une procédure de séparations et coupes pour les problèmes linéaires en nombres entiers multi-objectif. La méthode est applicable à tout problème pour lequel on peut définir la borne inférieure comme un ensemble de points dans l'espace des objectifs tel que cet ensemble soit calculable en un temps polynomial (ou pseudo-polynomial). Comme déjà remarqué, les itérations que l'on rencontre dans la méthode deux phases ou une approche ϵ-contrainte ne portent pas sur un problème NP-difficile mais sur un problème polynomial (ici un programme linéaire). L'idée est que l'on veut assurer que la complexité du problème ne se retrouve pas dans le calcul unique d'un point de l'ensemble Pareto optimal mais dans une procédure qui renvoie au terme de son exécution l'ensemble non-dominé. D'autre part, la procédure est faite de telle sorte qu'à chaque étape les décisions que l'on prend influent sur l'ensemble que l'on est en train de calculer pour pouvoir arrêter la résolution avant que la preuve de l'optimalité du résultat soit établie dans le cadre d'une procédure heuristique. Finalement, la méthode est définie pour dévier le moins possible de l'algorithme de séparations et coupes standard. Le schéma général est reporté dans l'algorithme 1. Ce dernier suit le déroulement classique, les différences apparaissant dans les notions de bornes inférieures et supérieures, leur comparaison ($\preceq$), leur comparaison et l'impact sur l'élagage et les stratégies de branchement. Ces points sont détaillés ci-dessous.

2.4.2 Bornes inférieures et supérieures

Les bornes supérieures et inférieures correspondent chacune à un ensemble de points dans l'espace des objectifs. Une borne inférieure lb est un ensemble de points dans l'espace des objectifs de telle sorte que chaque solution réalisable soit correspond à un de ces points, soit est dominée par au moins un de ces points. On suppose que chacun des points de la borne inférieure

Algorithme 1 Algorithme de séparations et coupes multi-objectif (MOB&C)

Etape 1 (racine de l'arbre)
Générer une borne supérieure initiale ub.
Définir le premier sous-problème.
Insérer le sous-problème dans une liste Λ.

Etape 2 (critère d'arrêt)
Si $\Lambda = \emptyset$ alors STOP, sinon choisir un sous-problème de Λ et le retirer de Λ.

Etape 3 (résolution du sous-problème)
Résoudre le sous-problème pour obtenir une borne inférieure lb.

Etape 4 (génération de contraintes)
Si des solutions entières ont été trouvées, tenter de les insérer dans ub.
si $ub \preceq lb$. **alors**
  Aller à l'étape 2.
sinon
  si des contraintes violées sont identifiées pour des solutions fractionnaires de la borne inférieure **alors**
    Les ajouter au modèle et aller à l'étape 3.
  sinon
    Aller à l'étape 5.
  fin si
fin si

Etape 5 (branchement)
Brancher sur *des* variables, introduire des nouveaux sous-problèmes dans λ. Aller à l'étape 2.

correspond à une solution, non nécessairement réalisable et que ces solutions forment l'ensemble $S_{lb} = \{x : F(x) \in lb\}$. La borne supérieure ($ub$) est un ensemble de points dans l'espace des objectifs qui correspondent à des solutions réalisables qui ne se dominent pas deux à deux. On définit $S_{ub} = \{x \in D : F(x) \in ub\}$ comme l'ensemble des solutions correspondant à un point dans la borne supérieure. Par la suite, nous considérerons que S_{lb} peut être divisé en deux sous-ensembles par rapport à S_{ub} : $S_{lb}^{\succ} = \{s \in S_{lb} : \exists y \in S_{ub}, y \preceq s\}$ et $S_{lb}^{\prec} = \{s \in S_{lb} : \nexists y \in S_{ub}, y \preceq s\}$. Il est à noter que les solutions de $S_{lb}^{\prec}$ sont nécessairement fractionnaires car on suppose que les solutions entières non réalisables sont éliminées par l'utilisation de plans sécants.

Le calcul de la borne inférieure doit rester polynomial ou au pire pseudo-polynomial. De ce fait, deux buts doivent être considérés lors de la définition de la borne inférieure : i) le nombre de points dans lb doit rester polynomial ou pseudo-polynomial par rapport à la taille du problème et ii) le calcul d'un point de lb doit rester polynomial ou pseudo-polynomial. Les principes de la méthode restent corrects si ces conditions ne sont pas respectées mais elle deviendra vite inapplicable. La méthode de calcul de la borne inférieure est clairement problème dépendante (des éléments sont exposés ci-dessous). La borne supérieure est gérée via une archive de solutions potentiellement efficaces constituées de solutions trouvées par MOB&C. L'archive est mise à jour

à chaque fois qu'une solution réalisable entière est identifiée.

2.4.3 Calcul de la borne inférieure

Pour les problèmes linéaires en nombres entiers multi-objectif, la borne inférieure peut être définie par un ensemble Φ de sous-problèmes obtenus à partir de la relaxation linéaire et d'une méthode scalaire. Dans les faits, la taille de Φ peut ne pas être bornée. Par exemple, si on utilise une agrégation linéaire, le nombre de combinaisons possibles de poids est infini mais seulement un sous-ensemble peut être résolu à l'instar de la première phase de la méthode à deux phases. Donc, une borne inférieure définie par un ensemble de problèmes Φ nécessite le calcul d'un sous-ensemble $\bar{\Phi} \subseteq \Phi$. Si $|\bar{\Phi}|$ est polynomiale ou pseudo-polynomiale, le but recherché pour un calcul efficace de la borne inférieure est atteint. Chaque solution $s \in S_{lb}$ peut être associée à un sous-ensemble $\Phi_s \subseteq \Phi$. Un algorithme pour un sous-problème $\phi \in \Phi_s$ retourne s ou une solution s' tel que $F(s) = F(s')$ (on ne cherche qu'une solution pour chaque point dans lb mais il s'agit d'un détail d'implémentation et cela peut être généralisé).

Comme nous résolvons des relaxations linéaires, les points obtenus dans lb peuvent être des solutions fractionnaires et des plans sécants peuvent être recherchés pour accélérer la recherche. Nous ne proposons pas ici de politique générale d'application d'un algorithme de plans sécants dans un cadre multi-objectif. Cependant expérimentalement, il apparaît qu'il peut être avantageux de calculer des plans sécants à chaque fois qu'un sous-problème de Φ est résolu avant de passer à un autre sous-problème. En effet, la coupe générée peut avoir un impact sur d'autres sous-problèmes que celui considéré et il est donc inutile de résoudre un de ces sous-problèmes avant d'avoir inclus la coupe. Cela est illustré sur le problème du voyageur de commerce avec labels dans le chapitre 3. Cependant, il ne s'agit là que de considérations expérimentales et le sujet mérite une exploration plus poussée.

2.4.4 Elagage et élagage partiel

L'élagage d'un nœud de l'arbre de recherche peut être effectué si l'une des trois conditions suivantes est satisfaite ; i) infaisabilité : $lb = \emptyset$; ii) optimalité : $\forall x \in S_{lb}, x \in D$; iii) dominance : $ub \preceq lb$. La dominance entre deux ensembles S_1 et S_2 est définie ainsi : $S_1 \preceq S_2 \Leftrightarrow \forall x \in S_2, x \in S_1$ ou $\exists y \in S_1, y \preceq x$. La relation de dominance entre les bornes inférieures et supérieures est illustrée dans la figure 2.6.

Dans le cas (1), la borne inférieure domine complètement la borne supérieure et la recherche doit donc être continuée. Inversement, dans le cas (2), la borne inférieure est complètement dominée par la borne supérieure et la recherche peut donc être stoppée. Le cas (3) est un cas propre à l'optimisation multi-objectif. Ici les deux ensembles sont incomparables, c'est-à-dire que ni la borne inférieure ni la borne supérieure ne domine complètement l'autre. Il faut donc continuer la recherche. Cependant, il ne peut pas y avoir de solution efficace dans la zone grisée. Il faut donc éviter de continuer à calculer la borne inférieure dans les sous-problèmes associés à cette zone. C'est ce qui est pris en compte via l'élagage partiel décrit ci-dessous.

Une zone de l'espace des objectifs nécessairement dominée pour le sous-arbre courant peut être construite à partir de $S_{lb}^{\succ}$. Cette zone est composée des points dans l'espace des objectifs dominés par l'image dans l'espace des objectifs des solutions de $S_{lb}^{\succ}$. De ce fait, les points de cette zone ne doivent pas être générés car il n'est pas nécessaire d'affiner la borne inférieure dans cette zone. Eviter la génération de points pour la borne inférieure dans cette zone est fortement dépendant du

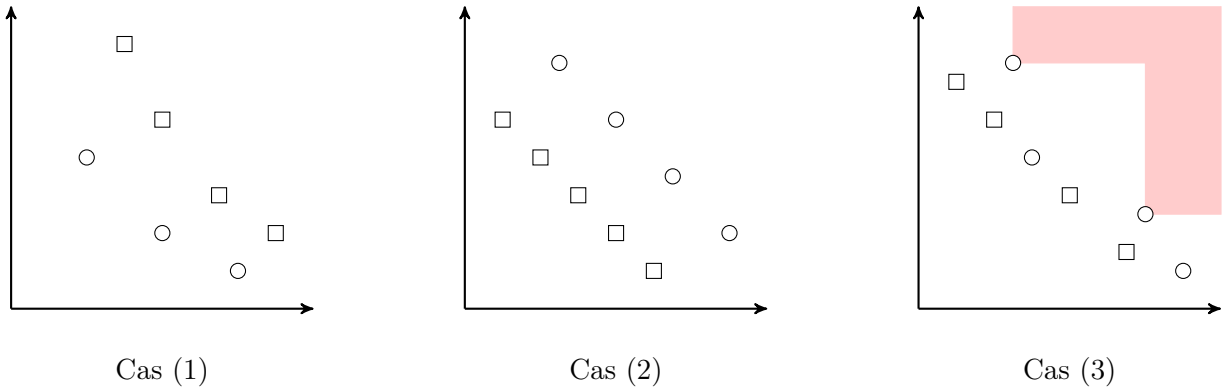


Fig. 2.6 – Relations de dominance possibles entre la borne inférieure (cercle) et la borne supérieure (carré).

problème et de la méthode de calcul de la borne inférieure. Par exemple, il est possible d'ajouter des contraintes ou d'utiliser des mécanismes ad-hoc comme une gestion d'une liste de valeurs interdites pour un des objectifs. C'est l'approche qui est utilisée pour l'application de la méthode au problème du voyageur de commerce avec labels multiples décrite dans le chapitre XX.

2.4.5 Branchement parallèle

A l'instar des recherches arborescentes classiques, différentes stratégies peuvent être utilisées pour sélectionner la variable sur laquelle brancher. Il est à noter que seules les solutions dans $S_{lb}^{\prec}$ ont besoin d'être considérées pour le branchement. Comme la borne inférieure n'est pas constituée d'une unique solution, une difficulté non présente dans la méthode classique est à prendre en compte. En effet, il se peut qu'il n'y ait pas une variable qui soit fractionnaire pour toutes les solutions de $S_{lb}^{\prec}$. De plus, une variable peut être fractionnaire pour différentes solutions mais avec des valeurs différentes et il est alors nécessaire d'exclure des intervalles différents. Il n'y a donc pas forcément une unique variable qui soit fractionnaire et valable pour toutes les solutions de $S_{lb}^{\prec}$.

Pour illustrer les problèmes rencontrés, supposons que l'on décide de brancher sur une variable χ en excluant un intervalle $]\alpha, \beta[$ avec $\alpha \in \mathbb{N}$ et $\beta = \alpha + 1$. Soit χ^s la valeur de χ dans la solution $s \in S_{lb}^{\prec}$. On peut diviser $S_{lb}^{\prec}$ en trois sous-ensembles : i) $S_{lb}^{\leq \chi} = \{s \in S_{lb}^{\prec} | \chi^s \leq \alpha\}$; ii) $S_{lb}^{= \chi} = \{s \in S_{lb}^{\prec} | \alpha < \chi^s < \beta\}$; iii) $S_{lb}^{\geq \chi} = \{s \in S_{lb}^{\prec} | \chi^s \geq \beta\}$. Lorsque $\chi \leq \alpha$ est imposé, les solutions dans $S_{lb}^{\leq \chi}$ ne sont pas impactées. De plus $\chi \leq \alpha$ peut ne pas donner une contrainte efficace car χ^s peut déjà être entière pour certaines solutions $s \in S_{lb}^{\geq \chi}$. Un raisonnement similaire peut être tenu lorsque l'on impose $\chi \geq \beta$.

Il serait plus efficace si toutes les solutions étaient impactées dans le sous-problème résultant du branchement car on aurait une mise-à-jour de l'ensemble de l'approximation du front à chaque fois en accord avec notre volonté de définir une méthode *anytime*. Cela est atteint via un mécanisme de *branchement parallèle*. Dans ce but, il est nécessaire de gérer séparément des choix de branchement pour chaque problème de Φ. De manière plus formelle, k couples distincts $(\chi_\kappa,]\alpha_\kappa, \beta_\kappa[)$ $(\kappa = 1, \ldots, k)$ doivent être sélectionnés de telle sorte que (i) $\bigcup_k S_{lb}^{= \chi_k} = S_{lb}^{\prec}$; (ii) $\bigcap_k S_{lb}^{= \chi_k} = \emptyset$. Ainsi chaque solution fractionnaire est affectée par une et une seule règle de

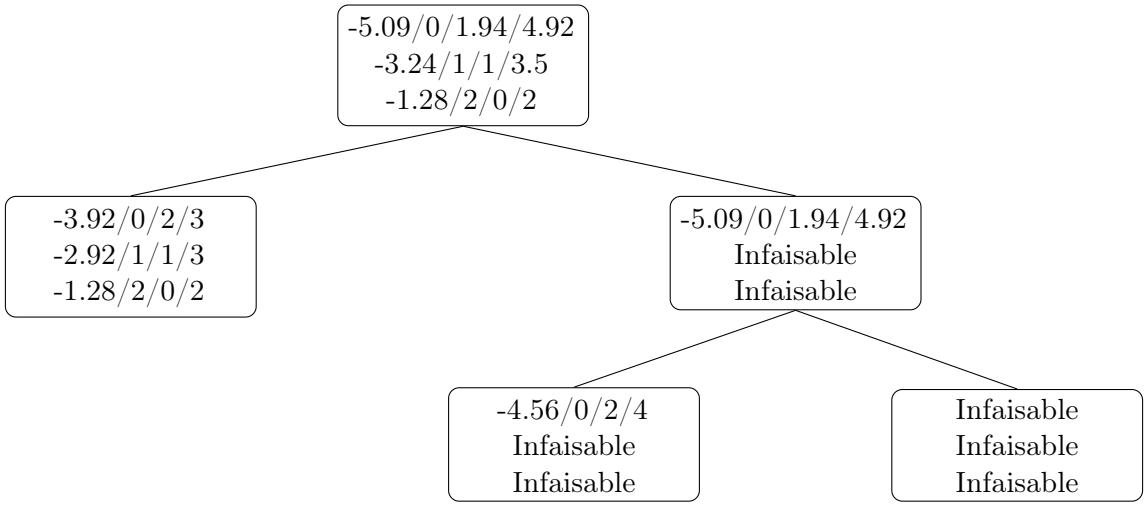


Fig. 2.7 – Arbre de recherche standard. La première ligne (resp. la seconde et la troisième ligne) donne la valeur des deux objectifs et de x_1 et x_2 pour s_0 (resp. s_1 et s_2).

branchement. Il apparaît expérimentalement que k doit être gardé le plus petit possible. Cela se comprend par le fait que l'on souhaite avoir une recherche la plus unifiée possible et non pas un grand nombre de recherches qui sont effectuées en parallèle. Dans les faits, le premier sous-problème (respectivement le second sous-problème) à inclure dans la liste Λ est créé ainsi : pour tous les $\kappa \in \{1, \ldots, k\}$, pour tous les $s \in S_{lb}^{=\chi_\kappa}$ et pour tous les $\phi \in \Phi_s$, la contrainte $\chi_\kappa \leq \alpha_\kappa$ (respectivement $\chi_\kappa \geq \beta_\kappa$) est ajoutée à ϕ.

2.4.6 Exemple numérique

Soit le programme linéaire en nombres entiers multi-objectif :

$$\begin{aligned}
\text{minimiser} \quad & -1.00x_1 - 0.64x_2 && (2.1)\\
\text{minimiser} \quad & x_3 && (2.2)\\
50x_1 + 31x_2 &\leq 250 && (2.3)\\
3x_1 - 2x_2 &\geq -4 && (2.4)\\
x_1 + x_3 &\leq 2 && (2.5)\\
x_1, x_2 &\geq 0 \text{ et entières} && (2.6)\\
x_3 &\in \{0, 1, 2\}. && (2.7)
\end{aligned}$$

On définit $\Phi = \{\phi_\epsilon : \epsilon \in \{0, 1, 2\})$ où ϕ_ϵ est obtenu en relaxant linéairement le programme et en posant $x_3 = \epsilon$. On notera la solution de ϕ_ϵ par s_ϵ.

La figure 2.7 représente l'arbre de recherche si le branchement est effectué sur la variable la plus fractionnaire. On suppose que chaque problème ϕ_ϵ est résolu à chaque nœud. On branche donc sur x_2 selon sa valeur dans s_1. Le sous-problème de gauche ($x_2 \leq 3$) est alors élagué par optimalité. Dans le sous-problème de droite ($x_2 \geq 4$), on branche en fonction de x_1. L'ensemble efficace est trouvé en 5 nœuds et requiert le calcul de 15 programmes linéaires.

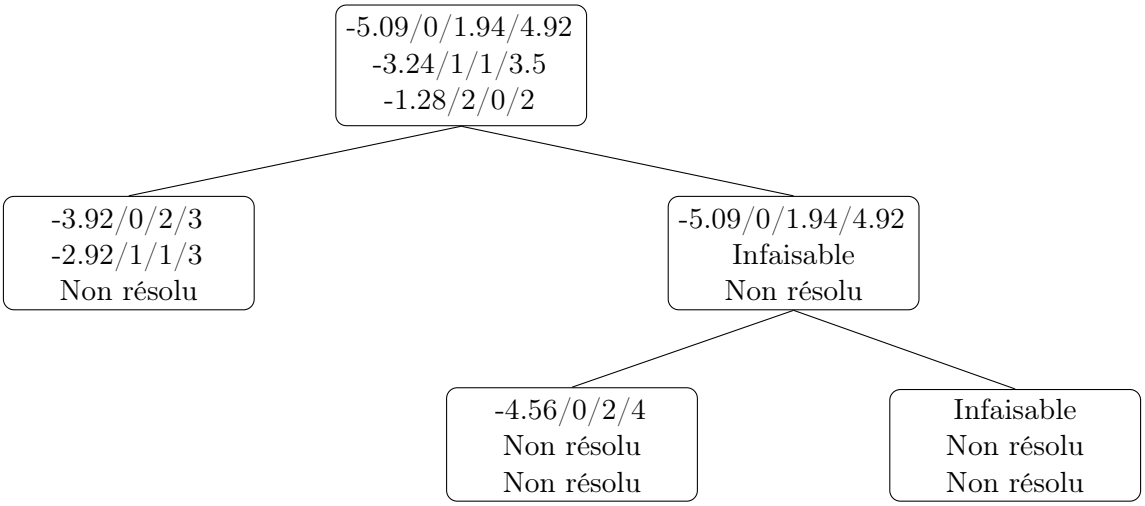


Fig. 2.8 – Arbre de recherche avec élagage partiel. La première ligne (resp. la seconde et la troisième ligne) donne la valeur des deux objectifs et de x_1 et x_2 pour s_0 (resp. s_1 et s_2).

Comme s_2 est entière à la racine de l'arbre, elle est efficace. De ce fait, brancher conduira toujours à une solution dominée ou infaisable pour ϕ_2. De la même manière, dans le sous-problème de droite au premier niveau de l'arbre, s_1 devient infaisable et il n'est pas utile de le recalculer dans les sous-problèmes suivants. L'élagage partiel peut être utilisé pour éviter ces calculs inutiles. On obtient alors l'arbre de recherche de la figure 2.8. L'ensemble efficace est toujours trouvé en 5 nœuds mais il ne nécessite que 9 résolutions de programmes linéaires.

En ce qui concerne le branchement, il apparaît dans cet exemple que lorsqu'on impose $x_2 \geq 4$ dans le sous-problème de droite au second niveau de l'arbre de recherche, ϕ_0 n'est pas affecté et sa résolution conduit à la même solution qu'à la racine de l'arbre. Cela est l'une des situations que le branchement parallèle permet d'éviter. La figure 2.9 montre l'arbre de recherche obtenu via l'application de ce mécanisme. A la racine, pour s_0, on branche sur x_2 en imposant $x_2 \leq 4$ dans un sous-problème et $x_2 \geq 5$ dans l'autre. On branche aussi sur x_2 pour s_1 mais en excluant l'intervalle $]3, 4[$. En choisissant un branchement pour chaque sous-problème, on assure que chaque calcul dans les sous-problèmes a un impact sur la borne inférieure. Par exemple, la solution optimale pour ϕ_0 est directement obtenue dans le premier sous-problème alors qu'elle est trouvée presque à la fin de la recherche dans la figure 2.8. Avec les deux mécanismes, la recherche nécessite seulement 3 nœuds et la résolution de 7 programmes linéaires.

2.5 Génération de colonnes en optimisation combinatoire multi-objectif

Les études portant sur l'utilisation de la génération de colonnes pour des problèmes multi-objectif sont rares. Une première utilisation est faite par Bard et Purmono [9] pour un problème d'emploi du temps d'infirmières. Cependant, bien que le problème soit multi-objectif, il est traité comme un problème mono-objectif. Le problème est résolu par des heuristiques. Ehrgott et Tind [47] combine la génération de colonnes et la méthode ϵ-contrainte. Ils utilisent l'analyse

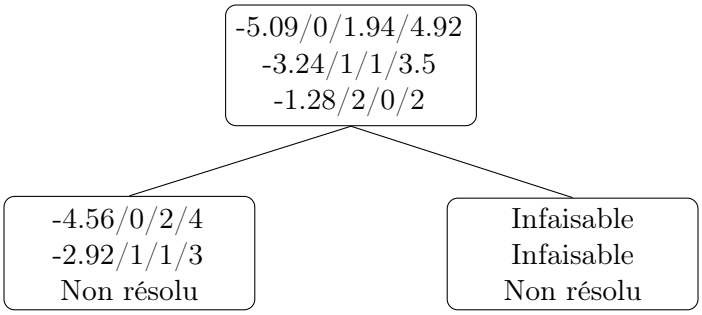


Fig. 2.9 – Arbre de recherche avec élagage partiel et branchement parallèle. La première ligne (resp. la seconde et la troisième ligne) donne la valeur des deux objectifs et de x_1 et x_2 pour s_0 (resp. s_1 et s_2).

de sensibilité pour générer des solutions proches de solutions trouvées. Khanafer et al. [77] proposent une méta-heuristique basée sur la génération de colonnes pour une variante bi-objectif du problème de bin packing. Salari et Unkelbach [118] traite un problème multi-objectif pour la planification de radiothérapies. Les auteurs utilisent une somme pondérée. Du fait du grand nombre de colonnes nécessaires pour construire une solution, ils utilisent la stratégie suivante. L'ensemble Pareto est construit en fonction des colonnes présentes dans le problème maître restreint. Puis, un sous-problème est défini pour rechercher des colonnes qui améliorent au mieux la frontière courante dans son ensemble. Du fait de la difficulté du problème, la méthode finale est une heuristique basée sur la génération de colonnes.

2.5.1 Problèmes considérés

Dans le cadre de ce document nous allons nous centraliser sur des problèmes dont la modélisation en programme linéaire en nombres entiers peut prendre une forme particulière. Cependant, certains mécanismes proposés peuvent être appliqués à des problèmes plus génériques. Ici, on considère les problèmes de la forme suivante :

$$\begin{array}{rcll} \text{minimiser} & & \sum_{k\in\overline{\Omega}} c_k\lambda_k & \\ \text{minimiser} & & \lambda_{\max} & \\ \sum_{k\in\overline{\Omega}} a_{ik}\lambda_k & \geq & b_i & (i \in I) \\ \lambda_{\max} & \geq & \rho_k\lambda_k & (k \in \overline{\Omega}) \\ \lambda_k & \in & \{0,1\} & (k \in \overline{\Omega}) \end{array}$$

L'ensemble $\overline{\Omega}$ représente l'ensemble des colonnes possibles. On considère que le vecteur c est entier et que les valeurs possibles de ρ_k sont aussi entières et qu'il y en a un nombre fini compris dans l'intervalle $[\rho_{\min}, \rho_{\max}]$ et on suppose les valeurs classées par ordre croissant ; on a donc $i < j \Rightarrow \rho_i \leq \rho_j$.

Pour chaque colonne k, ρ_k est une valeur associée et on doit choisir les colonnes de telle sorte que l'on optimise un premier objectif global liant les différentes colonnes choisies et un second objectif qui force à prendre les colonnes les moins coûteuses. Par exemple, pour un problème de tournées de véhicules, on peut considérer un objectif classique qui minimise le coût total de la solution tandis que le second objectif est lié à une caractéristique des tournées : longueur

maximale autorisée, qualité de service vis-à-vis des clients de la tournée ... Des exemples peuvent être facilement trouvés dans le cadre de problèmes qui composent une solution à partir de solutions qui peuvent être optimisées indépendamment. En plus des problèmes de tournées, on peut citer par exemple les problèmes d'ordonnancement sur machines parallèles, des problèmes de packing et de découpe, des problèmes de couvertures. Deux autres exemples seront illustrés dans ce document : le problème de bin parking et le problème de tournées couvrantes avec plusieurs véhicules.

Clairement, ce genre de modèle se prête bien à une résolution par une approche ϵ-contrainte en contraignant le second objectif. Par contre, on n'est pas obligé ici d'ajouter une contrainte spécifique dans le modèle qui peut détériorer la qualité de la relaxation linéaire pour le calcul d'une borne inférieure. Dans l'approche, le paramètre ϵ peut prendre un nombre fini de valeurs qui correspondent aux valeurs possibles de ρ_k. On notera ces valeurs ϵ_i $(i = 1, \ldots, p)$ et on les suppose classées par valeur croissante : $e_i < e_j \Leftrightarrow i < j$. Les valeurs forment une séquence σ.

L'idée est ici de modifier les ensembles de colonnes pour y inclure l'information sur le second objectif. On change donc l'ensemble $\overline{\Omega}$ pour un ensemble Ω où une colonne $k \in \Omega$ est définie par un tripletUBPP (c_k, a, ρ_k). Dans certains cas, comme le problème de bin packing ci-dessous, on a $\overline{\Omega} = \Omega$. C'est-à-dire que les choix à faire dans le sous-problème sont les mêmes et la valeur ρ_k est fixée par ces choix. Dans d'autres cas, comme le problème de tournées couvrantes avec plusieurs véhicules, on a $\overline{\Omega} \subseteq \Omega$ car la valeur ρ_k peut être un élément de décision par rapport au choix. Comme dans une approche classique de génération de colonnes, à une itération donnée, on travaillera sur un problème maître restreint défini par l'ensemble Ω_1 qui sera noté PMR(Ω_1). Le modèle s'écrit alors :

$$\begin{array}{rcll} \text{minimiser} & & \sum_{k\in\Omega_1} c_k\lambda_k & \\ \sum_{\lambda_k\in\Omega_1} a_i k & \geq & b_i & (i \in I) \\ 0 \leq & \lambda_k & 1 & (k \in \Omega_1) \end{array}$$

Lorsque l'on considérera le problème maître restreint par rapport à une valeur ϵ_i bornant le second objectif, on considère le problème maître restreint induit par Ω_1^i qui sera noté PMR(Ω_1^i) avec $\Omega_1^i = \{k \in \Omega_1 : \rho_k \leq \epsilon_i\}$. Il s'agit donc uniquement de "retirer" des colonnes sans avoir à modifier le problème de manière drastique.

2.5.2 Avantages

Cette approche offre de multiples avantages. D'abord le problème maître et le sous-problème sont des problèmes mono-objectif qui ne demandent pas de mécanismes particuliers à l'optimisation multi-objectif pour être résolus. On peut les résoudre facilement par des méthodes ϵ-contraintes [15] sans craindre un affaiblissement de la relaxation linéaire. D'autant plus que l'on est souvent proche du modèle pour une version mono-objectif du problème. Ainsi les méthodes de génération de colonnes pour la version mono-objectif peuvent facilement être utilisées dans le cadre multi-objectif. D'autre part, un avantage de travailler sur un modèle unique est que le sous-problème est le même pour toutes les valeurs de ϵ. Si les valeurs duales sont sensiblement identiques pour des valeurs proches de ϵ, il est probable qu'une solution du sous-problème de coût réduit négatif le soit aussi pour d'autres solutions du problème maître restreint associées à d'autres valeurs ϵ. C'est cet avantage que l'on exploitera par la suite dans la définition de mécanismes de recherche de colonnes de coût réduit négatif.

2.5.3 Un exemple : le problème de bin packing non-contraint

Pour illustrer les points précédents et certaines problématiques, nous allons utiliser une généralisation du problème de bin packing où l'on supprime la contrainte sur la taille maximale d'une bin appelée problème de bin packing non-contraint (UBPP). A la place, on cherchera à minimiser en plus du nombre de bins, la taille maximale des bins utilisées. On a donc un ensemble d'objets I avec pour chaque objet $o_i \in I$ une taille w_i. En utilisant la modélisation proposée dans [135], on obtient le modèle suivant. L'ensemble Ω est constitué de toutes les combinaisons possibles de I. Une bin b_k est alors associée à un sous-ensemble I_k de I. On définit alors $\rho_k = \sum_{o_i \in I_k} w_i$ et $a_{ik} = 1$ si $o_i \in I_k$, 0 sinon. En associant une variable binaire $\lambda_k = 1$ si et seulement si b_k est utilisée, le UBPP peut se modéliser :

$$
\begin{array}{rcll}
\text{minimiser} & & \sum_{b_k \in \Omega} \lambda_k & \\
\text{minimiser} & & \rho_{\max} & \\
\sum_{b_k \in \Omega} a_{ik} & = & 1 & (o_i \in I) \\
\rho_k \lambda_k & \leq & \rho_{\max} & (b_k \in \Omega) \\
\lambda_k & \in & \{0,1\} & (b_k \in \Omega)
\end{array}
$$

Pour résoudre le problème, on travaillera donc sur le problème maître restreint relâché induit par un sous-ensemble de colonnes Ω_1 suivant :

$$
\begin{array}{rcll}
\text{minimiser} & & \sum_{b_k \in \Omega_1} \lambda_k & \\
\sum_{b_k \in \Omega} a_{ik} & = & 1 & (o_i \in I) \\
\lambda_k & \geq & 0 & (b_k \in \Omega_1)
\end{array}
$$

Il s'agit du même modèle que celui proposé dans [135] et il est donc possible de réutiliser les résultats de ce dernier pour le résoudre.

2.5.4 Résultats de résolution basique

On a résolu le UBPP en utilisant les deux approches les plus basiques. Tout d'abord, une méthode ϵ-contrainte classique. On résout le problème pour la plus grande valeur de ϵ, puis on fixe la valeur pour être juste en-dessous de celle du second objectif obtenu à l'optimum et on itère. Sur la figure 2.10 qui correspond à une instance de petite taille, on obtient l'ensemble formé des deux points rouges. La seconde approche consiste à appliquer un algorithme de génération de colonnes pour chaque valeur possible de ϵ. On obtient la borne inférieure représentée par les points bleus dans la figure 2.10.

Dans le premier cas, on a une borne inférieure dégénérée. Si des valeurs de ϵ intermédiaires avaient été explorées, on aurait pu éviter cela ou du moins affiner la borne inférieure. Dans le second cas, on obtient la "meilleure" borne inférieure que l'on pourrait avoir pour ce problème avec cette modélisation. Mais le grand nombre de points fait en sorte qu'il faut résoudre un grand nombre de sous-problèmes. Dans le cadre du UBPP, où la résolution d'un sous-problème est triviale, cela demande un effort de calcul important et il faut donc des mécanismes pour accélérer la recherche sur des problèmes où le calcul d'un sous-problème est plus coûteux. Dans ce document, on se focalise sur des stratégies de recherche de colonnes de coût réduit négatif où la recherche est "factorisée" pour permettre d'impacter plusieurs sous-problèmes via une unique recherche. D'autres mécanismes sont aussi en cours d'exploration mais ne seront pas présentés ici.

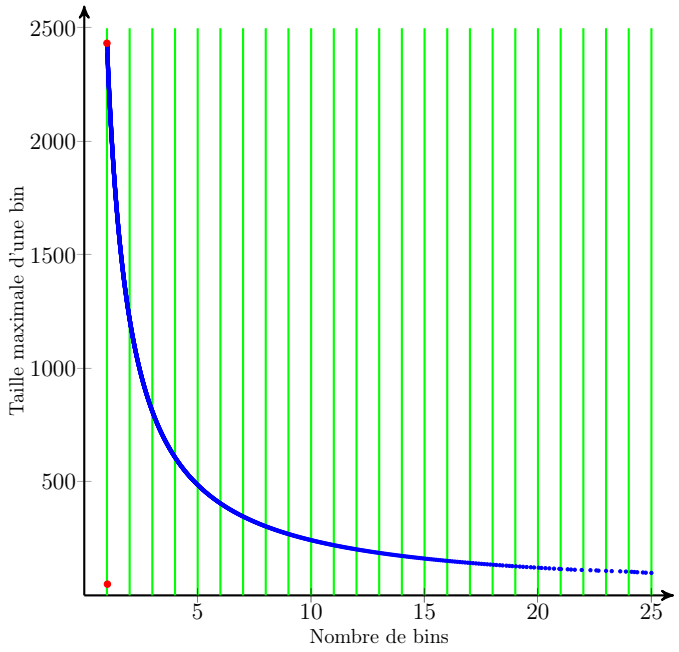


Fig. 2.10 – Bornes inférieures pour une instance du UBPP.

2.5.5 Stratégie de recherche de colonnes

La question est dans quel ordre faut-il rechercher les colonnes et cela a-t'il une influence sur l'efficacité de la méthode ? Faut-il résoudre complètement le problème induit par une valeur de ϵ avant de passer au suivant ? Ou faut-il résoudre le problème maître restreint pour plusieurs valeurs de ϵ et chercher des colonnes pour les solutions obtenues avant d'itérer ? Il faut aussi se demander comment on peut exploiter le résultat d'une résolution d'un sous-problème pour influer sur l'ensemble du front. En effet, la partie "coûteuse" de la méthode est cette résolution et il est intéressant d'en faire le moins possible. Une manière générique de définir la recherche est la suivante :

1 : **repéter**
2 : Sélectionner un sous-ensemble de valeurs $\overline{\sigma} \subseteq \sigma$
3 : **tant que** $\overline{\sigma} \neq \emptyset$ **faire**
4 : Choisir une valeur ϵ dans $\overline{\sigma}$ et la retirer de la liste
5 : Appliquer l'algorithme de génération de colonnes sur le sous-problème associé à ϵ avec une certaine intensité
6 : Mettre à jour $\overline{\sigma}$ et σ en fonction des colonnes trouvées
7 : **fin tant que**
8 : Utiliser les colonnes trouvées pour tenter de générer de nouvelles colonnes pour les autres sous-problèmes en exploitant les valeurs duales associées si elles ont été calculées
9 : Retirer les valeurs de σ pour lesquelles il y a convergence
10 : **jusqu'à** Il n'y ait plus de colonnes de coût réduit négatif

L'idée est qu'à chaque itération on peut choisir plusieurs valeurs pour ϵ et considérer les problèmes maîtres restreints associés à chacune. De la même manière, pour une valeur de ϵ, on peut résoudre le problème associé par génération de colonnes avec une certaine "intensité" variant d'un pas de l'algorithme de génération de colonnes à une convergence complète. A chaque fois que l'on trouve des colonnes de coût réduit négatif, il est possible de vérifier si elles ne sont pas aussi valides et de coût réduit négatif pour d'autres valeurs de ϵ ou de les utiliser pour générer rapidement des colonnes pour d'autres valeurs en exploitant ou non l'information des variables duales lorsque celle-ci est disponible (résolution de la relaxation du problème maître).

Nous allons présenter trois approches parmi celles étudiées. La première, la recherche point-par-point, est la méthode classique. La seconde est une amélioration permettant de générer à moindre coût des colonnes à partir de celles trouvées. Par contre, la résolution se fait toujours de la même manière. La dernière stratégie, l'approche "un pour tous et tous pour un", permet une recherche sur l'ensemble du front via l'utilisation d'heuristiques ad-hoc pour la génération rapide de colonnes. Ces stratégies seront testées et comparées sur le problème de tournées couvrantes avec plusieurs véhicules dans le chapitre 4.

Recherche point-par-point (PPS)

La première approche correspond à l'application directe d'une méthode ϵ-contrainte. Pour chaque valeur de ϵ envisagée, on laisse la génération de colonnes converger et on calcule la prochaine valeur de ϵ en fonction de la valeur trouvée. La méthode est résumée dans l'algorithme 2. Cette approche peut être appliquée à tous les problèmes. Elle ne suppose rien sur la forme du

Algorithme 2 Recherche point-par-point (PPS)

[t]

1 : Set $\varepsilon = \varepsilon_{\max}$.
2 : **tant que** $\varepsilon \geq \varepsilon_{\min}$ **faire**
3 : Résoudre le PMR à l'optimalité par génération de colonnes
4 : Stocker la solution
5 : Fixer ε à la prochaine valeur dans σ strictement inférieure à la valeur du second objectif
6 : **fin tant que**

problème maître et la relation entre le second objectif et des valeurs associées aux colonnes.

Recherche point-par-point améliorée (IPPS)

Dans la recherche point-par-point améliorée (Algorithme 3), l'idée est de rechercher des colonnes pour un sous-problème à partir de celles trouvées pour la valeur ϵ courante à l'aide d'heuristiques rapides. Cependant, les recherches heuristiques n'utilisent aucune information sur les sous-problèmes pour les autres valeurs de ϵ. Elles utilisent la structure d'un problème pour tenter d'améliorer le second objectif (qui est lié uniquement à la colonne) tout en conservant le fait que la colonne reste de coût réduit négatif.

Algorithme 3 Recherche point-par-point améliorée (IPPS)

Set $i \leftarrow 1$, $\varepsilon^i \leftarrow \infty$, and $lb \leftarrow \emptyset$.
tant que LRMP is feasible **faire**
  Solve LRMP once to obtain a vector of dual values π^i.
  Let $\bar{c}^i$ be the optimum, $\bar{\lambda}^i$ the optimal solution, and compute $\bar{\sigma}^i = \max_{k\in\Omega} \sigma_k \bar{\lambda}^i$.
  Solve $s_i = (\bar{c}^i, \bar{\sigma}^i, \pi^i, \varepsilon^i)$ and let $\Lambda = \{\bar{\lambda}^j : j = 1, \ldots, |\Lambda|\}$ be the set of columns obtained.
  si $|\Lambda| \neq 0$ **alors**
    pour $j \leftarrow 1$ to $|\Lambda|$ **faire**
      Apply heuristics to quickly generate other relevant columns from $\bar{\lambda}^j$.
    fin pour
  sinon
    $lb \leftarrow lb \cup \{(\bar{c}^1, \bar{\sigma}^1)\}$.
    Set $\bar{\lambda}_k \leftarrow 0$ for all k such that $\sigma_k \geq \bar{\sigma}^i$.
    Set $\varepsilon^{i+1} \leftarrow \bar{\sigma}^i - 1$.
    Set $i \leftarrow i + 1$.
  fin si
fin tant que

Algorithme 4 Un pour tous, tous pour un (SOGA)

repéter
  Générer un ensemble de points pour différentes valeurs non convergées ε dans σ
  Soit Λ l'ensemble des variables duales associées à ces différents points
  Résoudre le sous-problème associé pour un des points et ajouter les colonnes trouvées au PMR
  Modifier les colonnes trouvées pour obtenir des colonnes pour les autres points en utilisant les informations sur les variables duales stockées dans Λ
  Stocker les solutions qui ont convergé
jusqu'à Λ soit vide

Approche "un pour tous, tous pour un" (SOGA)

Une dernière approche est appelée "un pour tous, tous pour un" (SOGA) (Algorithme 4). La méthode commence par résoudre le problème maître restreint pour différentes valeurs de ϵ et sauvegarde les valeurs duales associées à la valeur de ϵ qui les a données. La méthode choisit alors un des points et résout le sous-problème associé. On utilise alors la solution trouvée et les autres valeurs duales pour chercher heuristiquement des colonnes de coût réduit négatif pour les autres valeurs de ϵ. A la différence de la recherche point-par-point améliorée, cette méthode utilise explicitement l'information duale des autres points pour générer des colonnes à partir de celle trouvée par la résolution d'un sous-problème donné ; ainsi il est plus probable qu'elle trouve des colonnes affectant les autres valeurs de ϵ. Différentes stratégies peuvent être utilisées pour choisir le point à résoudre.

Deuxième partie

Application au transport terrestre et à la logistique

Chapitre 3

Problème du voyageur de commerce avec labels

Dans mes travaux, une famille de problèmes étudiée est celle de la recherche de séquences avec la prise en compte de labels ou de familles. Dans ces problèmes, on associe à certains éléments du problème (les sommets ou les arêtes du graphe, les tâches d'un problème d'ordonnancement ...) un label. Il est alors possible de définir des objectifs sur ces labels. Typiquement, on cherchera à minimiser le nombre de labels différents utilisés ou le nombre de fois que l'on change de labels dans la séquence. Il est possible aussi de pondérer les labels et les changements de labels. Un exemple de ce type de problème que j'ai étudié est un problème d'ordonnancement à une machine avec temps de setup et minimisation du retard pour lequel un algorithme de branch-and-bound couplé à une approche ϵ-contrainte est proposé [6]. Dans ce chapitre, le problème du voyageur de commerce avec labels est défini (3.1). Après une analyse polyèdrale (3.2), un algorithme de branch-and-cut est proposé pour la résolution des variantes mono-objectif du problème (3.3). Ensuite, une implémentation de MOB&C pour la version bi-objectif est présentée dans la section 3.4 et les méthodes sont comparées dans la section 3.5.

3.1 Définition et modélisation

Le problème du voyageur de commerce avec labels (MLTSP) est une extension du problème du cycle Hamiltonien avec minimisation des labels (MLHCP) (aussi appelé problème du voyageur de commerce coloré) [26, 142]. Le MLTSP est défini sur un graphe non orienté valué $G = (V, E)$ avec V l'ensemble des sommets et E l'ensemble des arêtes. Un ensemble L de labels (parfois appelés *couleurs* [142]) est aussi défini. Chaque arête $e \in E$ possède un coût c_e et un label $\delta(e) \in L$. Le but est de déterminer un cycle Hamiltonien qui minimise la longueur de la tournée et le nombre de labels utilisés. La figure 3.1 est ainsi optimale en nombre de labels utilisés. Par contre, il est possible de faire mieux en longueur mais cela induira l'utilisation d'un label supplémentaire.

Le MLTSP peut être formulé comme une extension du modèle pour le problème du voyageur de commerce de Dantzig et al. [33]. Des variables binaires x_e ($e \in E$) et u_k ($k \in L$) sont définies. La variable x_e vaut 1 si et seulement si e est utilisée et u_k vaut 1 si et seulement si le label k est

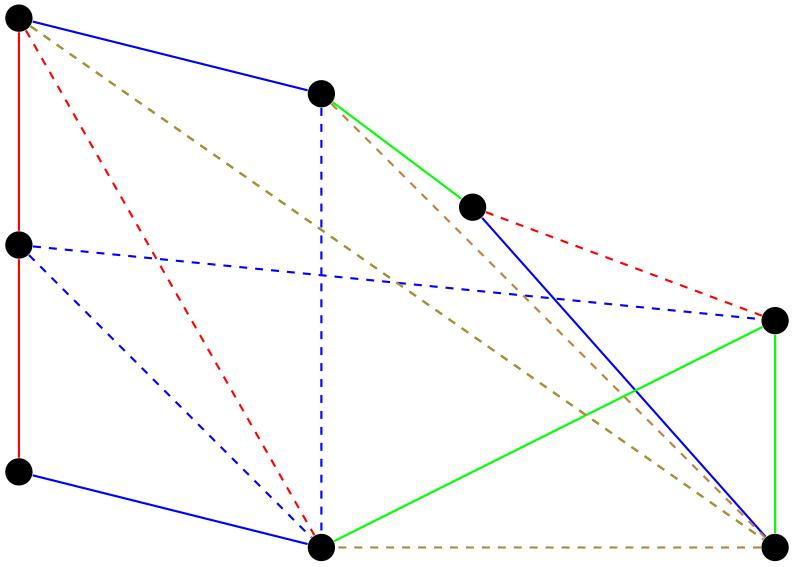


Fig. 3.1 – Une solution pour le MLTSPM.

utilisé. Le problème est alors :

$$\text{minimiser} \quad \sum_{e \in E} c_e x_e \tag{3.1}$$

$$\text{minimiser} \quad \sum_{k \in L} u_k \tag{3.2}$$

$$\sum_{e \in \omega(\{i\})} x_e = 2 \qquad (i \in V) \tag{3.3}$$

$$\sum_{e \in \omega(S)} x_e \geq 2 \qquad (S \subset V, 3 \leq |S| \leq |V| - 3) \tag{3.4}$$

$$x_e \leq u_{\delta(e)} \qquad (e \in E) \tag{3.5}$$

$$x_e \in \{0,1\} \qquad (e \in E) \tag{3.6}$$

$$u_k \in \{0,1\} \qquad (k \in L), \tag{3.7}$$

avec $\omega(S) = \{e = (i,j) \in E | i \in S, j \in V \setminus S\}$.

L'objectif (3.1) minimise la longueur du cycle et l'objectif (3.2) le nombre de labels utilisés. Les contraintes (3.3) sont des contraintes de degré et les contraintes (3.4) des contraintes de connectivité. Les contraintes (3.5) assurent que si on utilise une arête $e \in E$ alors le label $\delta(e)$ est aussi utilisé. Les contraintes (3.6) et (3.7) sont des contraintes binaires.

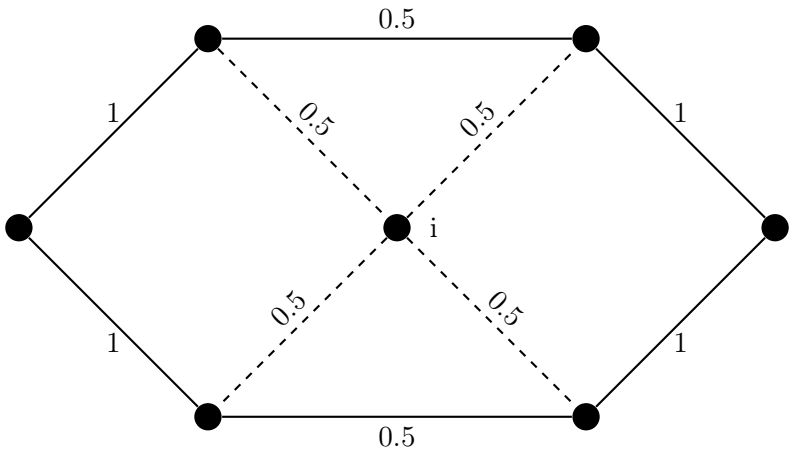


Fig. 3.2 – Une contrainte (3.9) est violée pour i et le label représenté par des tirets.

Il a été démontré [70] que les contraintes ci-dessous sont des inégalités valides :

$$u_k \leq \sum_{e \in \zeta(k)} x_e \qquad (k \in L) \tag{3.8}$$

$$\sum_{e \in \omega(\{i\}) \cap \zeta(k)} x_e \leq 2u_k \qquad (i \in V, k \in L, 3 \leq |\omega(\{i\}) \cap \zeta(k)| \leq |V| - 3) \tag{3.9}$$

$$\sum_{e \in E(S) \cap \zeta(k)} x_e \leq (|S| - 1)u_k \quad (k \in L,\ S = \{i, j, k\} \subset V,\ \forall e \in S \times S, u_{\delta(e)} = k) \tag{3.10}$$

$$\sum_{k \in L} \gamma_k(S) u_k \geq 2 \qquad (S \subset V, 3 \leq |S| \leq |V| - 3) \tag{3.11}$$

avec

$$\gamma_k(S) = \begin{cases} |\omega(S) \cap \zeta(k)| & \text{if } |\omega(S) \cap \zeta(k)| \leq 1 \\ 2 & \text{sinon.} \end{cases}$$

Les contraintes (3.8) signifient que si un label k est utilisé, alors au moins l'une des arêtes possédant ce label doit être utilisée. Les contraintes (3.9) sont une conséquence des contraintes de degré (3.3). Une solution de la relaxation linéaire respectant les contraintes de degré mais pas les contraintes (3.9) est donnée dans la figure 3.2. Il est aussi possible de dériver des contraintes d'élimination de sous-tours les contraintes 3.11. La figure 3.3 représente une solution relâchée pour laquelle les contraintes d'élimination de sous-tours sont respectées alors que les nouvelles inégalités ne le sont pas.

On a aussi les relations suivantes entre les solutions du MLTSP et celles du problème du voyageur de commerce (TSP) [70].

Lemme 1. *1. Une solution pour le TSP correspond à au moins une solution pour le MLTSP.*

2. Si les contraintes (3.8) sont ajoutées au modèle, il existe alors une bijection entre les solutions du TSP et les solutions du MLTSP.

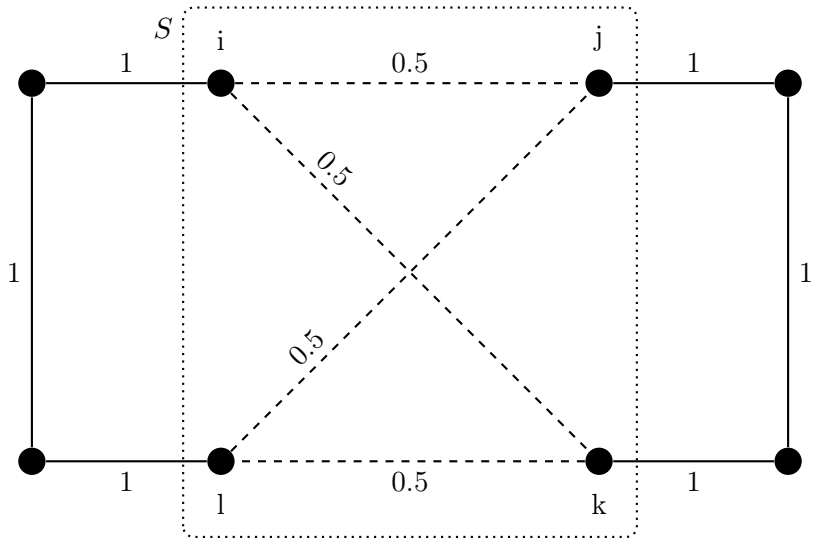


Fig. 3.3 – Une contrainte (3.11) est violée pour S et le label représenté par des tirets.

3.2 Analyse polyèdrale

Une analyse polyèdrale a aussi été effectuée. Les résultats sont énoncés ci-dessous. Les preuves se trouvent dans [70]. Dans cette partie, on note n le nombre de sommets, p le nombre de labels, et $\zeta(k) \subseteq E$ est l'ensemble des arêtes possédant le label k. Les enveloppes convexes de l'ensemble des solutions réalisables du TSP et du MLTSPM sont respectivement notées $\mathcal{C}_{\text{TSP}}$ et $\mathcal{C}_{\text{MLTSP}}$, c'est-à-dire :

$$\mathcal{C}_{\text{MLHCP}} = \text{conv}\{(x,u) \in \mathbb{R}^{n(n-1)/2} \times \mathbb{R}^p : (x,u) \text{ satisfait } (3.3) - (3.7)\}$$

et

$$\mathcal{C}_{\text{TSP}} = \text{conv}\{x \in \mathbb{R}^{n(n-1)/2} : x \text{ satisfait } (3.3),\ (3.4) \text{ et } (3.6)\}.$$

Théorème 1. *Si $F = \{y \in \mathcal{C}_{TSP} : \pi y = \pi_0\}$ est une face propre de $\mathcal{C}_{TSP}$ et $\pi x \leq \pi_0$ est une inégalité valide pour le MLTSP alors $\bar{F} = \{(x,u) \in \mathcal{C}_{MLTSP} : \pi x = \pi_0\}$ est une face propre de $\mathcal{C}_{MLTSP}$*

Théorème 2. *Soit $k \in L$ et $i \in V$, si $|3 \leq |\omega(\{i\}) \cap \zeta(k)| \leq n-3$, et si on suppose qu'il existe $e \in \zeta(k) \setminus \omega(\{i\})$ tel que pour $e', e'' \in \omega(\{i\}) \setminus \zeta(\{k\}$, $\{e, e', e''\}$ n'est pas un cycle, alors la face $F^{ki}_{3.9} = \{(x,u) \in \mathcal{C}_{MLTSP} : \sum_{e \in \omega(\{i\}) \cap \zeta(k)} x_e - 2u_k = 0\}$ définie par la contrainte (3.9) pour k et i est une face propre de $\mathcal{C}_{MLTSP}$.*

Théorème 3. *Soit $k \in L$ et $S \subset V$, si $3 \leq |S| \leq n-3$ et $\forall i \in S, |\omega(\{i\}) \cap \zeta(k)| \geq |S|/2$, alors la face $F^{kS}_{3.11} = \{(x,u) \in \mathcal{C}_{MLTSP} : \sum_{e \in E(S) \cap \zeta(k)} x_e - (|S|-1)u_k = 0\}$ définie par la contrainte (3.11) pour k et S est une face propre de $\mathcal{C}_{MLTSP}$.*

Lemme 2. *Pour $n \geq 3$, dim $\mathcal{C}_{MLHCP}$ = n(n-3)/2 + p sous la condition que $\forall i \in V, \forall k \in L, |\omega(\{i\}) \setminus \zeta(k)| \geq n/2$.*

Dans le reste de cette partie, on définit pour tous les $k \in L$, $\mathcal{C}^k_{\text{MLTSP}} = \mathcal{C}_{\text{MLTSP}} \cap \{(x,u) \in \mathbb{R}^{n(n-1)/2} \times \mathbb{R}^p : u_k = 0\}$.

Théorème 4. *Si l'inégalité*

$$\sum_{e \in \bar{E}} \alpha_e x_e \leq \beta \tag{3.12}$$

avec $\bar{E} \subseteq E$ définit une facette de $\mathcal{C}_{TSP}$ et $\forall k \in L, \mathcal{C}^k_{MLTSP} \neq \emptyset$, alors

$$\sum_{e \in \bar{E}} \alpha_e x_e \leq \beta - \sum_{k \in \bar{L}} \theta_k (1 - u_k) \tag{3.13}$$

est valide pour $\mathcal{C}_{MLTSP}$ pour tous $\theta_k \leq \bar{\theta_k}$ avec $\bar{L} = \{k \in L : \bar{E} \cap \zeta(k) \neq \emptyset\} = \{k_1, k_2, \ldots, k_q\}$ et $\bar{\theta}_{k_i} = \beta - \max(\sum_{e \in \bar{E}} \alpha_e x_e + \sum_{j=1}^{i-1} \theta_{k_j} 1 - u_{k_j} : (x,u) \in \mathcal{C}^{k_i}_{MLTSP})$. De plus, si $\theta_k = \bar{\theta}_k, \forall k \in L$, alors (3.13) est une facette de $\mathcal{C}_{MLTSP}$.

Théorème 5. *Les inégalités $x_e \geq 0$ pour $e = (n,m) \in E$ définissent des facettes de $\mathcal{C}_{MLTSP}$ pour tout $n \geq 4$ et si $\forall i \in V, |\omega(\{i\}) \backslash \zeta(\delta(e))| \geq n/2$ et $\forall k \in L \backslash \{\delta(e)\}$, les conditions suivantes tiennent : $\forall i \in V \setminus \{n,m\}, |\omega(\{i\}) \setminus \zeta(k)| \geq n/2$, $|\omega(\{n\}) \setminus \zeta(k)| - 1 \geq n/2$, et $|\omega(\{m\}) \setminus \zeta(k)| - 1 \geq n/2$.*

Théorème 6. *Les inégalités $x_e \leq u_{\delta(e)}$ pour $e \in E$ définissent des facettes de $\mathcal{C}_{MLTSP}$ pour tout $n \geq 4$ sous les conditions : $\forall i \in V, \forall k \in L, |\omega(\{i\}) \setminus \zeta(k)| \geq n/2$.*

Théorème 7. *Les inégalités $u_k \leq 1$ for $k \in L$ définissent des facettes de $\mathcal{C}_{MLTSP}$ pour tout $n \geq 3$ sous les conditions : $\forall i \in V, \forall k \in L, |\omega(\{i\}) \setminus \zeta(k)| \geq n/2$.*

3.3 Résolutions mono-objectif

Un algorithme de séparations et coupes a été proposé pour trois versions mono-objectif du MLTSP. La première version correspond au problème de recherche de cycle Hamiltonien avec minimisation du nombre de labels (PCHML) tel que défini par Cerulli et al. [26] et Xiong et al. [142]. Il s'agit d'une version où les arêtes ne sont pas valuées et où on cherche uniquement à minimiser le nombre de labels utilisés. Les deux autres problèmes sont obtenus en considérant l'un ou l'autre des objectifs comme une contrainte. Le premier, le problème de recherche de cycle Hamiltonien avec minimisation du nombre de labels et contrainte sur la longueur (PCHMLCL), est obtenu en retirant l'objectif (3.1) et en ajoutant la contrainte suivante :

$$\sum_{e \in E} c_e x_e \leq \gamma \tag{3.14}$$

avec γ la longueur maximum de cycle autorisée. Il s'agit donc de minimiser le nombre de labels en respectant une limite sur la longueur du cycle obtenu. L'autre problème, le problème du voyageur de commerce avec contrainte de labels (LCTSP), est l'inverse de celui-ci : on minimise la longueur du cycle en limitant le nombre de labels maximum que l'on peut utiliser. L'algorithme de séparations et coupes est le même pour les trois problèmes et il suit la procédure classique. Des détails sur son implémentation, notamment les contraintes utilisées et la recherche de coupes, peuvent être trouvés dans [70]. Il a été appliqué sur des instances spécifiques pour le PCHML et sur des instances modifiées de la TSPLIB pour les deux autres problèmes. Les résultats montrent

que l'algorithme est capable de résoudre la plupart des instances proposées en moins d'une heure. D'autre part, une heuristique basée sur une relaxation combinatoire du problème a aussi été proposée et elle obtient de bons résultats en des temps de calculs compétitifs. Pour une présentation et une analyse plus complète, le lecteur est invité à lire [70].

3.4 Résolution bi-objectif

La version bi-objectif a été résolue par une implémentation de MOB&C. Cette partie sert donc aussi à illustrer l'utilisation de cet algorithme dans un cas concret. Cependant, nous nous concentrerons sur l'implémentation des mécanismes de MOB&C et non sur les détails liés au problème en lui-même comme les coupes utilisées, leur recherche ou le modèle. D'autre part, à l'instar des versions mono-objectif, une heuristique basée sur une relaxation combinatoire et une approche ϵ-contrainte a aussi été définie pour le calcul d'une borne supérieure initiale. Ces points sont décrits en détail dans l'article [71] et le lecteur est invité à s'y référer. Nous allons expliciter ici la définition de la borne inférieure pour le MLTSPM, le mécanisme d'élagage partiel, la stratégie de branchement parallèle et la prise en compte de la génération de coupes.

3.4.1 Définition de la borne inférieure

A la racine de l'arbre de recherche, l'ensemble de problèmes définissant la borne inférieure est $\Phi = \{\phi_\epsilon : \epsilon \in \{1, \ldots, |L|\}\}$. Le problème ϕ_ϵ est défini par l'objectif (3.1) et les contraintes (3.3)(3.5)(3.8)-(3.10), la relaxation linéaire des contraintes (3.6) et (3.7) et la contrainte :

$$\sum_{k \in L} u_k \leq \epsilon. \tag{3.15}$$

Le calcul de cette borne est bien polynomial. Dans la suite de cette section, nous allons considérer que les solutions $s_i = (u^i, x^i) \in S_{lb}$ sont triées : si $1 \leq i < j \leq |S_{lb}|$ alors $\sum_{k \in L} u_k^i < \sum_{k \in L} u_k^j$. D'autre part, on a nécessairement $\sum_{e \in E} c_e x_e^i > \sum_{e \in E} c_e x_e^j$. Nous allons maintenant expliquer l'implémentation des mécanismes d'élagage partiel et de branchement parallèle, ainsi que la politique de génération de coupes. Puis, il sera montré que seul un sous-ensemble de Φ a besoin d'être calculé pour générer la borne inférieure dans le but d'éviter des calculs inutiles.

3.4.2 Elagage partiel

Une liste Λ_{pruned} est maintenue. Cette liste contient les valeurs de ϵ pour lesquelles soit la solution de ϕ_ϵ sera nécessairement dominée par la borne supérieure soit ϕ_ϵ sera infaisable. A la racine, Λ_{pruned} est vide. Elle est mise à jour en fonction du calcul de la borne inférieure et passée aux sous-problèmes éventuellement générés. Elle est restaurée lors de la remontée. La mise à jour est faite de la manière suivante :

- $\Lambda_{pruned} \leftarrow \Lambda_{pruned} \cup \{1, \ldots, \lfloor \sum_{k \in L} u_k^1 \rfloor\}$.
- Si $s_{|S_{lb}|} \in S_{lb}^{\succ}$, $\Lambda_{pruned} \leftarrow \Lambda_{pruned} \cup \{\lceil \sum_{k \in L} u_k^{|S_{lb}|} \rceil, \ldots, |L|\}$
- $\forall 1 \leq i < |S_{lb}|$, si $s_i \in S_{lb}^{\succ}$, alors $\Lambda_{pruned} \leftarrow \Lambda_{pruned} \cup \{\lceil \sum_{k \in L} u_k^i \rceil, \ldots, \lfloor \sum_{k \in L} u_k^{i+1} \rfloor\}$.

3.4.3 Branchement parallèle

Le branchement parallèle est géré par une matrice B de taille $|L| \times (|L| + |E|)$. $B[r, y]$ indique pour ϕ_r si la variable y est fixée à 0 ou 1, ou est libre. Au départ de la recherche, $B[k, y]$ est fixé à libre pour tout $k \in \{1, \ldots, |L|\}$ et pour toutes les variables y. La matrice est mise à jour de la manière suivante après le calcul de la borne inférieure. On commence par choisir la première variable sur laquelle brancher. Soit $k = \arg\max_{e \in E}(|S_{lb}^{=u_k}|)$. Si $|S_{lb}^{=u_k}| = 0$, cela signifie que toutes les variables u sont entières et on considère alors les variables x : $e = \arg\max_{e \in E}(|S_{lb}^{=x_E}|)$. Si $|S_{lb}^{=u_k}| > 0$, soit $v = u_k$, sinon $v = x_e$.

On fixe alors $B[r, v] = 0$ dans un sous-problème et $B[r, v] = 1$ dans l'autre pour chaque solution $s_i \in S_{lb}^{=u_k}$ et chaque r dans $\{\lceil \sum_{k \in L} u_k^i \rceil, \ldots, \lfloor \sum_{k \in L} u_k^{i+1} \rfloor\}$ pour lesquels $B[r, v]$ est libre. Si $s_{|S_{lb}|} \in S_{lb}^{=v}$, on sélectionne r dans $\{\lceil \sum_{k \in L} u_k^i \rceil, \ldots, |L|\}$. Si $S_{lb}^{\prec} \setminus S_{lb}^{=v} = \emptyset$, alors une variable de branchement a été choisie pour chaque solution fractionnaire non dominée et on procède au sous-problème suivant. Sinon, on fixe $S_{lb}^{\prec} \leftarrow S_{lb}^{\prec} \setminus S_{lb}^{=v}$ et le processus est réitéré.

3.4.4 Génération de coupes

L'ordre de génération des coupes peut avoir une influence sur la performance en temps de calcul de la méthode. On peut en effet considérer plusieurs stratégies (simples ici, des approches plus complexes peuvent faire l'objet d'une étude complémentaire). Ainsi, lors de la génération de la borne inférieure, on peut d'abord générer toutes les solutions fractionnaires puis ensuite rechercher des coupes pour ces points indépendamment. Cependant, deux solutions s_{ϵ_1} et s_{ϵ_2} "voisines" dans l'ensemble correspondant à des sous-problèmes ϕ_{ϵ_1} et ϕ_{ϵ_2} avec $\epsilon_1 > \epsilon_2$ ont des chances de ne pas être trop éloignées en termes de valeurs des variables. De ce fait des coupes générées pour s_{ϵ_1} peuvent être valides pour s_{ϵ_2}. La solution s_{ϵ_2} devient donc obsolète et la recherche de coupes pour s_{ϵ_2} n'a plus de raison d'être. Ce genre de stratégie peut s'envisager cependant dans un cadre où les points de la borne inférieure et les recherches de coupes se font en parallèle. Ce n'est pas le cas ici et on a donc opté pour une approche qui cherche les coupes au fur et à mesure que les points de la borne inférieure sont générés pour pouvoir les exploiter dans la suite du calcul de la borne inférieure. Pour un ϵ donné, on résout une relaxation du modèle et on recherche des coupes violées (principalement des contraintes d'élimination de sous-tours qui ont été retirées au départ) pour la solution du sous-problème si elle est fractionnaire et non dominée par la borne supérieure. Les contraintes ainsi générées sont donc ajoutées à tous les $\phi \in \Phi$. Cela se fait naturellement car il n'y a qu'un seul problème linéaire qui est géré et qui est commun à tous les sous-problèmes. Le changement d'un sous-problème à l'autre se faisant simplement via la modification du membre droit de la contrainte induite par l'approche ϵ-contrainte.

3.4.5 Calcul efficace de la borne inférieure

Dans de nombreux cas, comme par exemple pour le TSPML, il n'est pas nécessaire de calculer tous les $\phi \in \Phi$ pour avoir une borne inférieure valide car plusieurs sous-problèmes peuvent mener à la même solution. Il est nécessaire d'éviter ces calculs sous peine que la méthode ne soit pas efficace en termes de temps de calcul. Dans une approche par ϵ-contrainte comme c'est le cas ici, si pour un sous-problème ϕ_{ϵ_0} on obtient une solution s pour laquelle $\sum_{k \in L} u_k^s = \epsilon_1 \leq \epsilon - 1$ alors les résolutions des sous-problèmes $\phi_\epsilon \in [\epsilon_1, \epsilon_0 - 1]$ fourniront le même point (mais pas nécessairement la même solution). Ces calculs doivent être évités. Les valeurs pertinentes pour ϵ

Algorithme 5 Calcul de la borne inférieure.

ub est la borne supérieure
Λ_{pruned} est la liste d'élagage partiel
Fixer *élagué* $\leftarrow$ VRAI

Fixer $\epsilon \leftarrow \alpha$ avec $\alpha = \max\{\alpha \in \mathbb{N} | \alpha \notin \Lambda_{\text{pruned}} \text{ et } \alpha \leq |C|\}$.
tant que $\epsilon > 0$ **faire**
  Résoudre ϕ_ϵ (Algorithme (6))
  Soit o^* le nombre de labels utilisés dans la solution optimale, ou 0 s'il n'y a pas de solution.
  Fixer $\epsilon \leftarrow \alpha$ avec $\alpha = \max\{\alpha \in \mathbb{N} | \alpha \notin \Lambda_{\text{pruned}} \text{ et } \alpha \leq o^*\}$.
fin tant que

sont déterminées en appliquant l'algorithme 5. Cet algorithme maintient aussi la liste Λ_{pruned} et la borne supérieure. Il renvoie aussi le fait si le nœud courant doit être élagué ou non.

3.5 Résultats expérimentaux

La méthode a été testée sur des instances générées aléatoirement. Elle a été comparée à une approche ϵ-contrainte classique utilisant l'algorithme de séparations et coupes pour le LCTSP. Cette méthode a été implémentée de manière la plus efficace possible. Par exemple, les différentes itérations ne sont pas considérées comme complètement indépendantes mais exploitent les solutions déjà trouvées ainsi que les coupes déjà identifiées. Les instances sont définies par $|L| = 20, 30, 40, 50$ et $|V| = 20, 30, 40, 50$. Pour chaque valeur de $|V|$, cinq graphes différents sont générés et pour chaque combinaison de $|L|$ et $|V|$, cinq affectations différentes des labels sont créées. Les principaux résultats sont reportés ci-dessous ; pour une analyse plus approfondie, le lecteur est invité à consulter [71].

Les résultats sont reportés dans le tableau 3.1. Chaque ligne reporte la moyenne sur 25 instances. Pour chaque taille d'instance, on reporte la taille moyenne de l'ensemble efficace (#Par), le nombre de sommets de l'arbre de recherche (#Nœuds), le nombre de coupes générées (#Coupes) et le temps en secondes (Secondes). Pour MOB&C, on reporte également le temps moyen pour trouver l'ensemble efficace sans en prouver l'optimalité (Secondes*). Pour la méthode ϵ-contrainte, le nombre des noeuds est la somme des nombres de noeuds explorés dans tous les algorithmes de plans sécants. De manière générale, MOB&C apparaît plus rapide que l'approche ϵ-contrainte et explore au total un nombre plus faible de nœuds dans l'arbre de recherche. Il y a une corrélation ici mais il faut noter que l'écart dans le temps de calcul n'est pas proportionnel à celui en termes de nombre de nœuds ; le travail effectué par MOB&C pour un nœud étant plus important. Cependant, cela montre bien qu'il y a une *factorisation* de la recherche qui est perdue par la méthode ϵ-contrainte qui repart à chaque fois de zéro pour le branchement. Finalement, l'avantage de MOB&C est lié à l'utilisation du branchement parallèle et de l'élagage partiel. Autrement, on perdrait toute avance sauf sur l'aspect *anytime* de la méthode. En effet, si on fixe un temps de calcul maximum, la méthode ϵ-contrainte n'aura généré qu'une partie de l'ensemble tandis que MOB&C aura effectué des recherches sur l'ensemble du front.

Ce point est exploré dans le tableau 3.2 qui reporte les résultats obtenus quand on n'explore que $25\%, 50\%$ ou 75% des nœuds de l'arbre de recherche nécessaires pour prouver l'optimalité.

Algorithme 6 Calcul d'un sous-problème et génération de contraintes.

Fixer *continuer* ← VRAI
tant que *continuer* est VRAI **faire**
  continuer ← FAUX
  Résoudre ϕ_ϵ.
  Soit o^* le nombre de labels utilisés dans la solution optimale, ou 0 s'il n'y a pas de solution.
  si une solution est trouvée **alors**
    si la solution est réalisable et entière ou si elle est dominée par*ub* **alors**
      si la solution est réalisable et entière **alors**
        Essayer de l'ajouter dans *ub* et mettre à jour *ub* si nécessaire
      fin si
      $\Lambda_{\text{pruned}} \leftarrow \Lambda_{\text{pruned}} \cup \{\lceil o^* \rceil, \ldots, \epsilon\}$
    sinon
      si des contraintes violées sont identifiées **alors**
        Les ajouter au modèle
        continuer ← VRAI
      sinon
        élagué ← FAUX
      fin si
    fin si
  sinon
    $\Lambda_{\text{pruned}} \leftarrow \Lambda_{\text{pruned}} \cup \{1, \ldots, \epsilon\}$
  fin si
fin tant que

En plus du nombre de solutions efficaces (#Par), on reporte la taille de l'ensemble de solutions potentiellement efficaces (Size) et le pourcentage de solutions efficaces trouvées. La colonne "Gap" est calculée ainsi pour chaque solution $r = (u^r x^r)$ dans l'approximation A, on identifie la solution (u^{r*}, x^{r*}) dans l'ensemble efficace tel que $\sum_{k \in L} u_k^{r*} - \sum_{k \in L} y_k^r$ soit positif et aussi petit que possible. On calcule alors $g_r = \sum_{e \in E} c_e x_e^r / \sum_{e \in E} x_e x_e^{r*}$ et "Gap" est alors égale à $\sum_{r \in A} g_r / |A|$. Il s'agit donc de l'écart moyen en termes de longueur de tournées entre une solution de l'approximation et la solution la plus proche de l'ensemble efficace qui respecte la même contrainte sur le nombre de labels. Ces résultats montrent que la méthode est rapidement capable de trouver une partie significative de l'ensemble efficace tout en étant proche pour les solutions qui ne sont pas efficaces en un temps raisonnable. On peut donc envisager de l'utiliser comme heuristique pour générer une approximation de l'ensemble efficace, ce qui ne pourrait être le cas pour la méthode ϵ-contrainte. En effet pour cette dernière, il faudrait estimer à l'avance le nombre de solutions que l'on cherche à trouver et limiter la recherche de chaque exécution de l'algorithme de séparations et coupes à un temps égal au temps total imparti divisé par cette estimation. De plus, le choix des ϵ sera influencé par le fait que la solution retournée ne soit qu'une approximation et on pourra obtenir des solutions dominées par les solutions suivantes éventuellement.

TABLE 3.1 – Résumé des résultats pour l'algorithme de séparations et coupes.

			MOB&C				ϵCM		
$\|L\|$	$\|V\|$	#Par	#Nœuds	#Coupes	Secondes	Secondes*	#Nœuds	#Coupes	Secondes
20	20	10.3	227.6	56.2	1.4	1.0	561.2	59.2	1.4
20	30	13.9	452.2	130.9	8.8	6.6	1225.0	128.8	9.5
20	40	15.9	839.2	231.3	34.6	25.0	2206.4	232.0	33.6
20	50	17.4	1509.6	346.3	96.1	69.8	3248.2	324.8	100.3
30	20	12.4	418.3	87.0	2.9	2.1	1320.6	100.8	3.8
30	30	16.4	1006.8	188.8	26.3	19.3	3463.2	207.0	31.7
30	40	18.8	2552.0	388.2	177.1	126.8	6745.8	384.9	170.9
30	50	21.7	4735.3	588.2	431.0	286.9	12920.3	582.9	606.5
40	20	12.1	606.8	98.9	4.2	3.1	1571.0	107.6	5.0
40	30	17.8	1913.0	258.9	58.7	42.7	5806.0	273.4	67.2
40	40	21.7	4406.6	548.2	503.0	349.8	17462.0	578.3	665.8
40	50	26.6	15360.6	926.0	1845.9	1374.5	45306.6	1037.9	3334.5
50	20	12.4	718.9	102.8	4.4	3.4	2296.6	116.1	6.8
50	30	18.8	3248.3	355.0	144.0	110.2	12687.6	428.1	224.9
50	40	23.9	8722.7	738.3	1374.4	1097.7	36339.4	797.5	1636.9
50	50	27.7	20680.3	1204.9	4094.0	2902.5	74336.6	1307.5	5938.4

TABLE 3.2 – Résumé des résultats pour l'algorithme de séparations et coupes tronqué.

			25%			50%			75%			
$\|L\|$	$\|V\|$	#Par	Size	%	Gap	Size	%	Gap	Size	%	Gap	Secondes
20	20	10.3	10.0	60.2	1.010	10.3	72.8	1.006	10.3	88.3	1.002	0.9
20	30	13.9	13.8	56.8	1.007	13.9	71.9	1.004	13.9	90.6	1.001	5.1
20	40	15.9	15.6	47.8	1.008	15.8	65.4	1.005	15.8	83.6	1.002	17.0
20	50	17.4	17.0	46.6	1.006	17.3	60.9	1.004	17.3	79.9	1.002	48.2
30	20	12.4	12.0	62.1	1.009	12.2	76.6	1.004	12.3	92.7	1.001	1.9
30	30	16.4	16.2	49.4	1.008	16.4	67.7	1.004	16.4	89.6	1.001	14.3
30	40	18.8	18.4	38.3	1.010	18.7	51.6	1.007	18.7	75.5	1.002	78.0
30	50	21.7	21.0	40.6	1.009	21.4	55.3	1.005	21.6	82.5	1.002	188.8
40	20	12.1	11.5	58.7	1.011	11.9	76.0	1.005	12.0	87.6	1.002	2.7
40	30	17.8	17.3	41.6	1.010	17.7	62.9	1.005	17.8	83.7	1.002	30.0
40	40	21.7	21.2	31.3	1.011	21.6	43.8	1.007	21.7	80.2	1.002	200.3
40	50	26.6	25.4	34.2	1.009	26.2	51.9	1.006	26.4	71.8	1.003	708.0
50	20	12.4	11.8	59.7	1.011	12.2	69.4	1.009	12.3	84.7	1.004	2.9
50	30	18.8	18.3	41.0	1.012	18.6	63.8	1.005	18.6	86.2	1.002	75.4
50	40	23.9	23.1	34.3	1.011	23.9	51.9	1.005	23.8	82.0	1.002	601.8
50	50	27.7	26.5	24.5	1.012	27.2	40.8	1.007	27.6	69.7	1.003	1679.9

Chapitre 4

Problèmes de tournées couvrantes

Une seconde famille de problèmes de tournées qui est envisagée est celle composée par les problèmes où il n'est pas nécessaire de visiter tous les clients. Ces problèmes sont par nature bi-objectif même s'ils ne sont pas systématiquement exprimés comme cela. Un premier objectif vise à minimiser le coût de la ou les tournées effectuées tandis que le second objectif tend à minimiser une pénalité engendrée par les clients qui ne sont pas visités. Ce second objectif prend la forme d'une perte par rapport à la totalité des profits possibles dans les problèmes avec profits [49, 69] par exemple ou par un coût d'affectation dans le cadre du ring star problem [84, 95]. Dans ce chapitre, on s'intéresse aux problèmes de tournées couvrantes. Dans les versions mono-objectif, le second objectif est pris en compte via la définition d'un paramètre du problème. Dans une généralisation bi-objectif, ce paramètre est remplacé par un objectif [75]. Chaque sommet non couvert est affecté au sommet visité le plus proche. L'objectif est de minimiser la plus grande distance d'affectation. Ce chapitre s'intéresse aux variantes où l'on ne recherche pas uniquement une tournée mais un ensemble de tournées. Le chapitre est organisé de la manière suivante. Le problème est défini et des modèles mathématiques sont donnés dans la section 4.1. Un algorithme de branch-and-price pour la version mono-objectif est décrit dans la section 4.2. Enfin, une implémentation des mécanismes de génération de colonnes exposés dans le chapitre 2 est faite pour la version bi-objectif du problème dans la section 4.3.

4.1 Définition du problème de tournées couvrantes avec véhicules multiples

Le problème de tournées couvrantes avec véhicules multiples (mCTP) est une extension du problème de la tournée couvrante (CTP) [57]. Il a été introduit et résolu par heuristiques par Hachicha et al. [63]. Par la suite et très récemment, Tricoire et al. [132] ont proposé une étude d'une variante bi-objectif stochastique du problème, tandis que Hà et al. [61] ont proposé un algorithme de séparations et coupes et une métaheuristique pour un cas particulier.

Le problème est défini sur un graphe non-dirigé pondéré $G = (V \cup W, E)$. L'ensemble $V = \{v_1, \ldots, v_n\}$ est l'ensemble des sommets qui peuvent être visités et $W = \{w_1, \ldots, w_m\}$ contient les sommets qui doivent être couverts par un sommet de V. La notion de couverture n'est pas la même si on considère le problème de manière mono-objectif ou multi-objectif. Ce point sera abordé par la suite. L'ensemble E est l'ensemble des arêtes entre les sommets de W. On définit la

fonction de distance c soit en associant c_e au coût de l'arête $e \in E$, soit en associant une distance c_{ij} entre un sommet de w_i de W et un sommet v_j de V. Un sous-ensemble $T = \{v_1, \ldots, v_{|T|}\}$, avec $|T| \geq 1$, de sommets qu'il est nécessaire de visiter, est défini. Le but est de trouver une collection de tournées sur un sous-ensemble de V tout en assurant une "couverture" pour les ensembles de V. Des contraintes additionnelles sont aussi considérées : il n'est pas possible de visiter plus de p sommets dans une tournée et la longueur d'une tournée ne doit pas dépasser une valeur q.

Dans le problème mono-objectif tel que défini dans [57, 63], la notion de couverture est une donnée du problème. Chaque sommet $w_i \in W$ est associé à un sous-ensemble $\delta_i = \{v_j \in V \setminus \{v_1\} : c_{ij} \leq c\} \subset V$. La couverture est définie par la constante c qui représente un rayon de couverture. Pour chaque $w_i \in W$, il doit y avoir au moins un sommet de δ_i visité par l'une des tournées. Dans une généralisation bi-objectif du problème, on supprime la valeur c. De ce fait, toute collection de tournées vérifiant les contraintes sur les tournées est réalisable et on ajoute un second objectif qui minimise l'écart maximum entre un sommet de W et le sommet visité de V le plus proche.

4.1.1 Modélisation

Le problème est modélisé par un programme linéaire en nombres entiers basé sur des chemins. Pour la version bi-objectif, deux modèles sont fournis. Dans le premier, la valeur du second objectif associé à la couverture est explicitement définie et calculée par le modèle, tandis que dans le second modèle, le deuxième objectif est pris en compte implicitement via la définition des variables. Dans tous les modèles, les notations suivantes seront les mêmes. Soit Ω l'ensemble des tournées réalisables. Pour chaque route $r_k \in \Omega$, soit c_k la longueur de la tournée. Si le sommet $v_i \in V$ est visité par r_k, on fixe le paramètre b_{ik} à 1, 0 sinon. On définit des variables binaires égales à 1 si et seulement si r_k est utilisée.

4.1.2 Modèle pour la version mono-objectif

Pour chaque sommet $w_i \in W$ et pour chaque tournée $r_k \in \Omega$, on fixe le paramètre a_{ik} à 1 s'il y a au moins un sommet de δ_i visité par r_k, 0 sinon. D'autre part, pour chaque $v_i \in V \setminus \{v_1\}$, on fixe le paramètre γ_i à 1 s'il est nécessaire de visiter v_i. Autrement, ce paramètre est égal à 0. Le modèle s'écrit alors :

$$\text{minimiser} \quad \sum_{r_k \in \Omega} c_k \theta_k \tag{4.1}$$

$$\sum_{r_k \in \Omega} a_{ik} \theta_k \geq 1 \qquad (w_i \in W) \tag{4.2}$$

$$\sum_{r_k \in \Omega} b_{ik} \theta_k \geq \gamma_i \qquad (v_i \in V \setminus \{v_1\}) \tag{4.3}$$

$$\theta_k \geq 0 \text{ et entiers } (r_k \in \Omega) \tag{4.4}$$

L'objectif (4.1) minimise la somme des longueurs des tournées sélectionnées. Les contraintes (4.2) assurent que tous les sommets de W sont couverts, tandis que les contraintes (4.3) vérifient que l'on visite bien tous les sommets obligatoires. Les contraintes (4.4) imposent que les variables soient positives et entières. Il n'est pas nécessaire de les définir comme des variables binaires.

4.1.3 Premier modèle pour la version bi-objectif

Pour ce modèle, pour chaque pair $w_i \in W$ et $v_j \in V \setminus \{v_1\}$, on définit une variable binaire z_{ij} égale à 1 si et seulement si on décide de couvrir w_i par v_j. On ajoute aussi la variable $Cov_{\max}$ qui contient la valeur de la plus grande distance d'affectation définie par les variables z. On peut alors le modéliser par le modèle suivant noté BOMCTP$_1$.

$$\text{minimiser} \quad \sum_{r_k \in \Omega} c_k \theta_k \tag{4.5}$$

$$\text{minimiser} \quad Cov_{\max} \tag{4.6}$$

$$Cov_{\max} - d_{ij} z_{ij} \geq 0 \quad (w_i \in W, v_j \in V \setminus \{v_0\}), \tag{4.7}$$

$$\sum_{v_j \in V \setminus \{v_0\}} z_{ij} \geq 1 \quad (w_i \in W), \tag{4.8}$$

$$\sum_{r_k \in \Omega} a_{jk} \theta_k - z_{ij} \geq 0 \quad (w_i \in W, v_j \in V \setminus \{v_0\}), \tag{4.9}$$

$$\sum_{r_k \in \Omega} a_{jk} \theta_k \geq 1 \quad (v_j \in T \setminus \{v_0\}), \tag{4.10}$$

$$Cov_{\max} \geq 0, \tag{4.11}$$

$$z_{ij} \in \{0, 1\} \quad (w_i \in W, v_j \in V \setminus \{v_0\}), \tag{4.12}$$

$$\theta_k \in \mathbb{N} \quad (r_k \in \Omega). \tag{4.13}$$

Le premier objectif minimise la somme des longueurs des tournées et le second la plus grande distance entre un sommet de W et un sommet visité de V. Les contraintes (4.7) fixent la couverture à la plus grande valeur entre un sommet w_i et le sommet v_j par lequel il est considéré couvert. Les contraintes (4.16) vérifient que tous les sommets de W sont couverts, tandis que les contraintes (4.9) vérifient que si un sommet de V est utilisé pour couvrir un sommet de W alors il appartient à une tournée sélectionnée. Les contraintes (4.10) assurent que tous les sommets qui doivent être couverts, sauf le dépôt, sont visités par une tournée. Les autres contraintes sont des contraintes de signe et d'intégrité sur les variables.

4.1.4 Second modèle pour la version bi-objectif

Le second modèle pour la version bi-objectif redéfinit la notion d'une colonne. L'ensemble Ω est l'ensemble de toutes les colonnes et une colonne $\omega_k \in \Omega$ ne correspond plus seulement à une route r_k mais à un triplet (r_k, ϕ_k, ρ_k) avec $\phi_k \subseteq W$ l'ensemble des sommets de W que l'on déclare couverts par r_k. On note ρ_k la plus grande distance entre un sommet de ϕ_k et le plus proche sommet de r_k. La variable θ_k est égale à 1 si on choisit la colonne ω_k, elle est fixée à 0 sinon. Si on note $\Gamma_{\max}$ la couverture induite par les colonnes sélectionnées, on obtient alors le modèle BMCTP$_2$ suivant :

$$\text{Minimiser} \quad \sum_{\omega_k \in \Omega} c_k \theta_k \tag{4.14}$$

$$\text{Minimiser} \quad \Gamma_{\max} \tag{4.15}$$

$$\sum_{\omega_k \in \Omega} a_{ik} \theta_k \geq 1 \qquad (w_i \in W)\,, \tag{4.16}$$

$$\sum_{\omega_k \in \Omega} b_{ik} \theta_k \geq 1 \qquad (v_i \in T \setminus \{v_0\})\,, \tag{4.17}$$

$$\Gamma_{\max} \geq \rho_k \theta_k \qquad (\omega_k \in \Omega)\,, \tag{4.18}$$

$$\theta_k \in \{0,1\} \qquad (\omega_k \in \Omega)\,. \tag{4.19}$$

Les deux objectifs sont les mêmes que précédemment. Les contraintes (4.16) vérifient que chaque sommet $w_i \in W$ est couvert par au moins une tournée et de même les contraintes (4.17) assurent que tous les sommets de T autre que 0 sont visités par une tournée. Les contraintes (4.18) fixent la valeur de la couverture nécessaire pour couvrir les sommets de W en fonction des colonnes choisies et les contraintes (4.19) définissent les variables comme étant binaires.

Ce problème se situe dans la lignée des problèmes que nous avons définis pour l'application de la génération de colonnes. Il est donc possible de supprimer le second objectif qui porte sur la borne des différents éléments choisis. Dans le cadre de l'utilisation du modèle dans une approche par ϵ-contrainte, on associe le modèle $\text{BOMCTP}_2^{\epsilon}$ au problème obtenu en bornant le second objectif par ϵ :

$$\text{Minimiser} \quad \sum_{\omega_k \in \Omega_\epsilon} c_k \theta_k \tag{4.20}$$

$$\sum_{\omega_k \in \Omega_\epsilon} a_{ik} \theta_k \geq 1 \qquad (w_i \in W)\,, \tag{4.21}$$

$$\sum_{\omega_k \in \Omega_\epsilon} b_{ik} \theta_k \geq 1 \qquad (v_i \in T \setminus \{v_0\})\,, \tag{4.22}$$

$$\theta_k \geq 0 \text{ et entière} \qquad (\omega_k \in \Omega_\epsilon)\,. \tag{4.23}$$

avec $\Omega_\epsilon = \{\omega_k \in \Omega : \phi_k \leq \epsilon\} \subseteq \Omega$. Il est à remarquer que ce problème est très proche du modèle pour la version mono-objectif BOMCTP_1. En effet, la différence provient du fait que les a_{ik}, c'est-à-dire si un sommet de W est couvert ou non par la colonne, sont maintenant des éléments de décision dans le sous-problème. Ainsi l'ensemble des colonnes envisagées par ce modèle est plus large que l'ensemble mono-objectif et il contient les colonnes du modèle mono-objectif. Cependant, pour ne pas alourdir la notation, nous continuerons d'utiliser la notation Ω pour représenter cet ensemble. Donc, si le modèle mono-objectif fournit une borne correcte, le modèle bi-objectif fournira aussi une bonne borne pour le problème en faisant varier les ϵ.

4.2 Algorithme de branch-and-price pour la version mono-objectif

Pour évaluer la qualité de la borne inférieure, un algorithme de branch-and-price pour le problème mono-objectif a été développé.

4.2.1 Sous-problème

Soit λ_i ($w_i \in W$) (respectivement π_i ($v_i \in V \setminus \{v_1\}$) les variables duales associées aux contraintes (4.2) (respectivement aux contraintes (4.3)) dans la relaxation linéaire du problème maître restreint défini pour un sous-ensemble Ω_1 de Ω. Le sous-problème consiste dans la recherche d'un plus court chemin élémentaire avec contraintes de ressources commençant et se terminant en v_1 qui minimise l'objectif :

$$\sum_{e \in E} c_e x_e - \sum_{w_i \in W} \lambda_i a_i - \sum_{v_i \in V \setminus \{v_1\}} \pi_i b_i \tag{4.24}$$

avec x_e ($e \in E$) des variables binaires égales à 1 si et seulement si l'arête e est utilisée dans le chemin. Le chemin doit aussi respecter les deux contraintes :

$$\begin{cases} \sum_{v_i \in V \setminus \{v_1\}} b_i \leq p \\ \sum_{e \in V} c_e x_e \leq q \end{cases}$$

Il est aussi possible de considérer le sous-problème comme un problème de voyageur de commerce avec profits (PVCP) [49] en définissant le profit d'un sommet $v_i \in V \setminus \{v_1\}$ comme la somme des λ_i des sommets $w_i \in W$ qu'il peut couvrir. Cependant, une différence notable existe dans le fait que le profit d'un sommet v_i n'est pas fixe mais varie selon les sommets qui sont visités par la tournée. A notre connaissance, ce problème n'apparaît pas tel quel actuellement dans la littérature alors qu'il est d'un intérêt certain même en le considérant comme un problème à part entière et pas uniquement comme le sous-problème que nous étudions ici. Plusieurs modélisations ont été envisagées et étudiées et celle qui offre les meilleurs résultats dans le cadre d'une résolution par un algorithme de séparations et coupes est liée à la réduction du sous-problème en un Ring Star Problem (RSP) [84] développée ci-dessous.

4.2.2 Réduction polynômiale du sous-problème dans le Ring Star Problem

Le RSP est défini sur un graphe mixte $G_R = (V_R, E_R \cup A_R)$ où V_R est l'ensemble de sommets, E_R un ensemble d'arêtes et A_R un ensemble d'arcs. Un coût d'anneau c_{ij}^R est défini pour chaque arête $[v_i, v_j] \in E_R$ et un coût d'affectation d_{ij}^R est associé à chaque arc $(v_i, v_j) \in A_R$. Le but est de trouver un cycle sur un sous-ensemble V_R' de V_R tel que la somme du coût du cycle sur V_R' et du coût d'affectation des sommets qui ne sont pas sur le cycle soit minimale. Le coût d'affectation est égal à $\sum_{v_i \in V_R \setminus V_R'} \min_{(v_i,v_j) \in A_R} d_{ij}^R$.

Le sous-problème peut se réduire polynômialement en le Ring Star Problem en suivant les règles suivantes :

$$V_R \leftarrow V \cup W \tag{4.25}$$

$$E_R \leftarrow (V \times V) \tag{4.26}$$

$$A_R \leftarrow \bigcup_{w_i \in W} (\{(w_i, v_j) : v_j \in \delta_i\} \cup \{(w_i, v_1)\}) \tag{4.27}$$

$$c_e^R \leftarrow c_e \ (e \in E \cap (V \times V)) \tag{4.28}$$

$$d_a^R \leftarrow \begin{cases} -\lambda_i & \text{if } j \neq 1, \\ 0 & \text{otherwise} \end{cases} \ (a = (w_i, v_j) \in A_R) \tag{4.29}$$

La règle (4.25) indique que l'ensemble des sommets est composé de V et W. Seuls les sommets de V peuvent être visités du fait de la règle (4.26). La règle (4.27) limite l'affectation d'un sommet de W uniquement aux sommets de V qui peuvent le couvrir ou au sommet v_1 signifiant qu'il n'est pas couvert par la tournée. Le coût d'une arête pour le RSP est le même que celui dans le sous-problème (règle (4.28)). Le coût d'affectation est égal à moins la valeur de la variable duale associée à $w_i \in W$ si w_i est affecté à un sommet de δ_i, ce qui correspond à la collecte du profit associé à w_i. Ce coût est égal à 0 si w_i est affecté à v_1.

A partir du modèle proposé par Labbé et al. pour le RSP [84], il est possible de modéliser le sous-problème par le programme linéaire en nombres entiers suivant. Pour chaque arête $e \in E$, on associe une variable binaire x_e égale à 1 si et seulement si e est utilisée. On définit aussi pour chaque sommet $v_j \in V$ une variable binaire y_j égale à 1 si et seulement si v_j est visité. Les variables entières z_{ij} indiquent si un sommet $w_i \in W$ est affecté à un sommet $v_j \in \delta_i \cup \{v_1\}$. Le modèle est alors :

$$\text{minimiser} \quad \sum_{e \in E} c_e x_e - \sum_{v_j \in V \setminus \{v_1\}} \pi_j y_j - \sum_{w_i \in W} \lambda_i \sum_{v_j \in \delta_i \cup \{v_1\}} z_{ij} \tag{4.30}$$

$$\sum_{x_e \in \omega(j)} x_e = 2y_j \ (v_j \in V) \tag{4.31}$$

$$\sum_{v_j \in \delta_i \cup \{v_1\}} z_{ij} = 1 \ (w_i \in W) \tag{4.32}$$

$$\sum_{e \in \omega(S)} x_e \geq 2y_j \ (S \subset V, v_j \in V \setminus \{v_1\} : v_1 \notin S, v_j \in S\} \tag{4.33}$$

$$\sum_{e \in \omega(S)} x_e \geq 2 \sum_{v_j \in S \cap \delta_i} z_{ij} \ (S \subset V, w_i \in W : v_1 \notin S) \tag{4.34}$$

$$\sum_{v_j \in V \setminus \{v_1\}} y_j \leq p \tag{4.35}$$

$$\sum_{e \in E} c_e x_e \leq \min(q, \text{ub}) \tag{4.36}$$

$$y_1 = 1 \tag{4.37}$$

$$x_e \in \{0,1\} \ (e \in E) \tag{4.38}$$

$$y_j \in \{0,1\} \ (v_j \in V \setminus \{v_1\}) \tag{4.39}$$

$$z_{ij} \quad \text{entiers} \ (w_i \in W, v_j \in \delta_i \cup \{v_1\}) \tag{4.40}$$

L'objectif (4.30) minimise la longueur de la tournée en soustrayant les profits collectés. Les contraintes (4.31) sont des contraintes de degré qui vérifient que si un sommet $v_j \in V$ est visité alors deux arêtes adjacentes doivent être utilisées. Les contraintes (4.32) forcent chaque sommet w_i de W à être affecté à un sommet de $\delta_i \cup \{v_1\}$. Il y a deux familles de contraintes de connectivité : i) les contraintes (4.33) sont des contraintes de connectivité standard pour le problème du voyageur de commerce avec profits [49] ; ii) les contraintes (4.34) sont des contraintes introduites dans [84] qui généralisent les précédentes en assurant que si un sommet w_i est affecté à un sommet v_j alors des contraintes de connectivité relient v_j à v_1. La contrainte (4.35) borne le nombre de sommets visités et la contrainte (4.36) limite la longueur de la solution. La valeur ub est égale à une borne supérieure pour le problème complet. Cela est utile notamment lorsque $p = +\infty$ pour accélérer

la recherche dans le sous-problème puisqu'il est inutile de générer une tournée plus grande que la meilleure solution trouvée jusque là. La visite de v_1 est imposée par la contrainte (4.37). Les autres contraintes sont des contraintes d'intégrité sur les variables.

4.2.3 Algorithme de séparations et coupes pour le sous-problème

Le sous-problème est résolu par un algorithme de séparations et coupes. Initialement, on commence avec la relaxation linéaire du programme linéaire en nombres entiers présenté ci-dessus et en retirant les contraintes (4.33) et (4.34). D'autre part, les contraintes suivantes sont directement ajoutées au modèle :

$$x_e \leq y_j \ (e = [v_1, v_j] \in E) \tag{4.41}$$

À chaque nœud de l'arbre de recherche, on recherche jusque 300 contraintes violées dans l'ordre suivant. Premièrement, on recherche des contraintes violées de la forme :

$$\begin{cases} x_e \leq y_i \\ x_e \leq y_j \end{cases} \quad (v_i, v_j \in V \setminus \{v_1\} : e = [v_i, v_k] \in E) z_{ij} \leq y_j \ (w_i \in W; v_j \in \delta_i) \tag{4.42}$$

Ces contraintes sont dérivées des inégalités valides pour le RSP [84] :

$$x_{ij}^R + y_{ij}^R \leq y_{jj}^R \ (v_i, v_j \in V_R \setminus \{v_1\}) \tag{4.43}$$

Ensuite, on recherche des contraintes d'élimination de sous-tours (4.33) en identifiant des composantes connexes dans le sous-graphe $G_m = (V_m, E_m)$ défini pour une valuation $\overline{y}$ des variables y donnée par le résultat de la relaxation avec $V_m = \{v_j \in V : \overline{y} = 1\}$ et $E_m = E \cap \{e = [v_i, v_j] \in E : v_i, v_j \in V_m\}$. S'il n'y a qu'une seule composante connexe, on recherche une contrainte violée par la résolution d'un problème de coupe minimale sur G_m. Finalement, des contraintes (4.34) violées sont recherchées en utilisant la méthode de séparation proposée par Labbé et al. [84].

Les branchements sont effectués en priorité sur les variables y. Quand toutes les variables y sont entières, on considère les variables x. La variable la plus proche de 0.5 est sélectionnée pour le branchement et on commence par explorer la branche où la variable est fixée à 1.

4.2.4 Application de la génération de colonnes et stratégies de branchement

Le mCTP est alors résolu par un algorithme de branch-and-price. Le sous-ensemble de colonne Ω_1 initial est constitué par les tournées visitant uniquement un sommet en plus de v_1. A la fin de chaque application de la génération de colonnes, si la solution n'est pas entière, on résout le problème maître restreint en rétablissant l'intégrité des variables pour obtenir une solution entière et tenter d'améliorer la borne supérieure. Le modèle étant assez simple, cela est peu coûteux en temps de calcul par rapport à la recherche complète. Comme suggéré par Rousseau et al. [117], la relaxation du problème maître restreint est résolue par un algorithme de barrière. D'autre part, pour limiter le nombre de colonnes dans le problème maître restreint et ne pas considérer des colonnes inutiles lors de leur génération, une règle de dominance entre les colonnes est définie. Une colonne r_1 domine une colonne r_2 si et seulement si

$$\begin{cases} c_1 < c_2 \\ a_{i2} \leq a_{i1} \ (w_i \in W) \end{cases} \tag{4.44}$$

ou

$$\begin{cases} c_1 = c_2 \\ a_{i2} \leq a_{i1} \ (w_i \in W) \\ \exists w_i \in W : a_{i2} < a_{i1} \end{cases} \tag{4.45}$$

La partie branchement suit celle proposée par Boussier et al. [23] pour les problèmes de team orienteering. Comme il est habituel de le faire pour les problèmes de tournées [48], les branchements ne sont pas directement effectués sur les colonnes mais sur deux aspects des colonnes : la visite d'un sommet donné et l'utilisation d'une arête donnée. Dans un premier temps, un sommet v_j visité un nombre fractionnaire de fois est recherché. Si un tel sommet existe, deux branches sont créées. Dans la première branche, la visite de v_j est forcée en mettant $\gamma_j = 1$ dans le problème maître restreint. Dans la seconde branche, la visite de v_j est interdite en posant $\theta_k = 0$ pour toutes les colonnes visitant v_j et en imposant $y_j = 0$ dans le sous-problème.

Si aucun sommet répondant au critère précédent n'est trouvé, on considère les arêtes $e \in E$. Soit $e = [v_i, v_j] \in E$ une arête utilisée un nombre fractionnaire de fois. Deux cas peuvent alors se produire. D'un côté, si v_i ou v_j est obligé d'être visité, deux branches sont créées. Dans la première branche, e est rendue obligatoire. Au niveau du problème maître restreint, cela est fait en imposant $\theta_k = 0$ pour les colonnes r_k qui visitent v_i ou v_j mais n'utilisent pas e. Au niveau du sous-problème, on ajoute les deux contraintes suivantes au programme linéaire :

$$\begin{cases} y_i \leq x_e \\ y_j \leq x_e \end{cases} \tag{4.46}$$

La seconde branche correspond à l'interdiction de e. Cela se fait simplement en imposant $\theta_k = 0$ pour les colonnes r_k utilisant e dans le problème maître restreint et $x_e = 0$ dans le programme linéaire du sous-problème.

D'un autre côté, si ni v_j ni v_j doivent nécessairement être visités, trois branches sont créées. La première consiste en l'interdiction d'utiliser v_i et donc e. Une seconde branche force l'utilisation de v_i et e. La troisième branche force l'utilisation de v_i mais interdit d'employer e. Les choix sont pris en compte au niveau du problème maître restreint et du sous-problème comme précédemment.

4.2.5 Résultats expérimentaux

L'algorithme a été testé sur des intances générées aléatoirement. Des graphes ont été générés aléatoirement pour $|V| = 20, 30, 40, 50, 60$ et $|W| = 100, 150$. La cardinalité de T a été gardée à 1 car l'accroître tend à trivialiser le problème. En effet, on peut supprimer tous les sommets de W couverts par un sommet de T et en conséquence supprimer les sommets de V ne couvrant plus de sommets de W. On arrive alors rapidement à résoudre un problème du voyageur de commerce. Ce comportement peut être observé dans les résultats de Gendreau et al. [57]. Dans la même logique, le dépôt est considéré comme ne couvrant aucun sommet (sinon il suffit de supprimer les sommets couverts par le dépôt de W). La distance de couverture c est calculée de telle sorte que chaque sommet de W peut être couvert par au moins deux sommets de $V \setminus \{v_1\}$. Pour chaque combinaison de paramètres, 5 instances sont générées. Des tests ont été menés pour différentes combinaisons de paramètres. Ils sont reportés et analysés dans [68]. Il apparaît que la qualité de la borne inférieure est très bonne mais que les temps de calculs, notamment pour prouver l'optimalité, deviennent importants pour les instances à 50 ou 60 sommets.

4.3 Application de la génération de colonnes à la version bi-objectif

Les deux modèles proposés ont été utilisés pour tester les techniques de génération de colonnes. Dans les deux cas, des tests ont été menés pour voir l'effet de l'ordre dans lequel on génère dans les colonnes. Dans le second modèle, on utilise aussi la définition particulière pour proposer des mécanismes d'accélération, où à partir d'une colonne de coût réduit négatif, on génère rapidement d'autres colonnes en la modifiant. On regarde aussi la qualité des modèles et des stratégies de génération des colonnes dans un cadre où on utilise la méthode de manière heuristique. On ne reporte ici que les résultats obtenus pour le second modèle. Plus de détails peuvent être trouvés dans [120, 121].

4.3.1 Définitions du problème maître restreint et du sous-problème

Le problème maître restreint du modèle BOMCTP$_2^\epsilon$ est obtenu en relâchant les contraintes d'intégrité des variables et en ne considérant qu'un sous-ensemble de colonnes $\Omega_1^\epsilon \subset \Omega^\epsilon$ dans le modèle BCMCTP$_2^\epsilon$. D'autre part, on suppose que les variables sont juste positives ou nulles pour éviter les contraintes de type $\theta_k \leq 1$ qui ne servent pas. La prise en compte du second objectif est faite par la suppression des colonnes qui ne vérifient pas la limite sur la couverture.

Soit π_i ($w_i \in W$) les variables duales associées aux contraintes (4.21). On suppose pour cette étude que $T = \{v_1\}$ et on ne considère donc pas les contraintes (4.22) ici. Le dual du problème maître restreint s'écrit alors :

$$\text{Maximiser} \quad \sum_{w_i \in W} \pi_i \tag{4.47}$$

$$\text{sous-contraintes} : \quad \sum_{w_i \in W} a_{ik}\pi_i \leq c_k \qquad (k \in \Omega)\,, \tag{4.48}$$

$$\pi_i \geq 0 \qquad (w_i \in W)\,. \tag{4.49}$$

Le sous-problème consiste donc à trouver des tournées sur un sous-ensemble de V et des affectations de W à ces tournées de telle sorte que $c_k - \sum_{w_i \in W} a_{ik}\pi_i$ soit négatif, que le nombre de sommets visités et la longueur de la tournée ne dépassent pas les valeurs p et q respectivement et qu'aucune distance d'affectation d'un sommet de W à un sommet visité ne dépasse ϵ. En appliquant le même raisonnement que précédemment et en modifiant le graphe en un graphe dirigé sur un ensemble d'arcs A, on retrouve le fait que le sous-problème est un problème de plus court chemin élémentaire avec contraintes de ressources. Cependant, à l'instar du sous-problème pour la version mono-objectif, le coût des arcs est égal à celui des arêtes. La fonction objectif du sous-problème comporte un second terme qui récupère le profit associé lié à la couverture d'un sommet de W si on choisit de le couvrir par la tournée.

4.3.2 Résolution du sous-problème

Puisque $r_k \subseteq V$ et $\phi_k \subseteq W$, il faut construire une route sur un sous-ensemble de V en minimisant sa longueur tout en choisissant les sommets de W à couvrir pour engendrer le profit qui leur est associé. Le sous-problème est résolu par un algorithme de programmation dynamique, plus précisément l'algorithme *decremental state space relaxation* (DSSR) [22, 116]. Deux ressources sont considérées dans l'implémentation : le nombre de sommets dans la tournée et le fait que l'on

ne considère que les sommets de W qui sont à une distance inférieure à ϵ d'un sommet dans la tournée. Lors de l'extension d'un label, les sommets de W qui ne sont pas encore couverts et qui peuvent être couverts par l'ajout d'un nouveau sommet dans la tournée sont identifiés et le profit total qu'ils peuvent engendrer est calculé et soustrait du coût réduit du label. De cette manière, on garantit d'obtenir le coût minimal pour chaque label. Le test de dominance entre deux labels l_1 et l_2 se fait alors ainsi. Le test au niveau de la consommation des ressources correspond au test habituel. Par contre, lorsque l'on compare les deux coûts réduits, un facteur correspondant à la somme des profits des sommets couverts par l_1 mais pas encore par l_2 doit être retiré du coût de l_2. Cela assure que l'on n'élimine pas un label qui pourrait mener à une solution optimale.

4.3.3 Heuristique pour IPPS

Le sous-problème construit une colonne k en construisant ϕ_k comme l'ensemble de tous les sommets de W qui sont à moins d'une distance ϵ des sommets dans r_k. Cela facilite la recherche en minimisant le coût réduit de la colonne. Cependant, ϕ_k n'a pas besoin de contenir tous les sommets qu'il est possible de couvrir. N'importe quel sous-ensemble de W tel que la somme des profits associés est plus petite ou égale à la c_k est admissible. Ces colonnes ne seront jamais trouvées par la méthode DSSR tel qu'implémentée. L'heuristique présentée ici vise à les rechercher de manière rapide à partir des colonnes de coût réduit négatif trouvées par la méthode. Pour une colonne $k = (r_k, \phi_k, \rho_k)$, l'heuristique retire successivement le sommet de ϕ_k qui induit la valeur ρ_k. On retire si nécessaire un sommet de r_k si ce dernier ne couvre plus aucun sommet. Les colonnes générées de cette manière peuvent être valides pour d'autres valeurs de ϵ que celle qui est couramment traitée et ainsi accélérer la convergence du problème associé à ces valeurs lorsqu'elles seront envisagées. Par contre, leur génération est "aveugle" et n'utilise aucune information liée à ces valeurs. C'est ce qui est fait dans le cadre de SOGA.

4.3.4 Heuristique pour SOGA

Dans SOGA, le premier choix à faire est quelle valeur de ϵ faut-il choisir pour résoudre le sous-problème associé par DSSR ? Pour former une base de comparaison avec PPS et IPPS, on a choisi ici d'utiliser la plus grande valeur de ϵ pour laquelle on n'a pas encore convergé. L'heuristique de génération de nouvelles colonnes fonctionne de la manière suivante. Supposons que l'on ait une colonne $k_1 = (r_k^1, \phi_k^1, \rho_k^1)$ de coût réduit négatif pour le sous-problème résolu par DSSR pour le vecteur dual π_1 associé à ϵ_1. Considérons un vecteur π_2 associé à $\epsilon_2 \neq \epsilon_1$. Le principe est là aussi de modifier l'ensemble des sommets que l'on considère couverts par la tournée. On construit donc la colonne $k_2 = (r_k^2, \phi_k^2, \rho_k^2)$ ainsi. On commence par fixer $r_k^2 = r_k^1$ et $\phi_k^2 = \{w_i \in W : \exists v_j \in r_k^2, c_{ij} \leq \epsilon_2\}$. Si le coût réduit obtenu n'est pas négatif, on ne continue pas la procédure. Sinon, on calcule ρ_k^2 la couverture effective, c'est-à-dire la plus grande distance, entre un sommet de ϕ_k^2 et le sommet le plus proche de r_k^2. Finalement, on retire de r_k^2 les sommets qui ne couvrent pas seuls au moins un sommet de ϕ_k^2.

4.3.5 Résultats expérimentaux

Les méthodes ont été testées sur des instances générées aléatoirement qui suivent les procédures décrites dans [57, 65, 63]. Tous les programmes sont implémentés en C++ et les programmes linéaires sont résolus par ILOG CPLEX 12.4. Les tests ont été effectués sur un Intel Core 2 Duo

TABLE 4.1 – Temps de calcul (secondes cpu).

			|lb|				
p	$\|V\|$	$\|W\|$	moy.	dev. std.	PPS	IPPS	SOGA
5	30	60	25.0	3.2	18.03	13.88	13.00
5	30	90	20.8	4.8	15.38	14.60	12.28
5	40	80	26.0	1.9	49.50	40.46	36.68
5	40	120	27.2	4.9	126.75	94.01	86.39
5	50	100	31.6	3.0	205.79	153.87	154.51
5	50	150	29.8	3.4	392.54	286.95	268.06
8	30	60	25.4	3.1	49.76	29.69	25.46
8	30	90	22.2	4.0	31.26	23.30	18.68
8	40	80	27.0	2.9	113.41	103.59	86.81
8	40	120	29.4	4.3	511.23	503.89	326.63
8	50	100	32.2	2.8	1343.66	1012.47	821.23
8	50	150	29.8	3.8	1525.19	1186.22	1049.09

CPU E7500 avec 2 Go de mémoire. Des résultats pour plusieurs combinaisons de tailles d'instance et de valeurs de p sont reportés ci-dessous. Le tableau 4.1 reporte, en plus de la taille moyenne ($|\text{lb}|$) des ensembles de points dans la borne inférieure, les temps de calcul moyens des différentes stratégies. Il apparaît que SOGA est la meilleure et permet de gagner environ 33% de temps de calcul sur une implémentation basique d'une méthode ϵ-contrainte. Le gain est moins important par rapport à IPPS tout en restant intéressant.

Cette différence en temps de calcul s'explique par le nombre de fois que l'on résout un sous-problème complètement par DSSR. Le nombre de fois moyen pour chacune des méthodes est reporté dans le tableau 4.2. IPPS et SOGA résolvent un nombre bien moindre de sous-problèmes que PPS, ce qui illustre qu'il y a une factorisation de la recherche. Cela explique aussi en partie les différences de temps de calcul entre les méthodes, la résolution du sous-problème étant la partie la plus coûteuse. Ainsi une stratégie à la SOGA est utile pour les problèmes où la résolution du sous-problème est complexe.

Un dernier point étudié est le nombre de colonnes effectivement générées et se trouvant dans le problème maître restreint à la fin de la recherche. Les valeurs sont reportées dans le tableau 4.3. Il apparaît que la recherche aveugle de IPPS génère beaucoup plus de colonnes qu'une recherche ciblée par SOGA. Par contre, il n'y a pas de différence claire entre PPS et SOGA. Garder le nombre de colonnes générées le plus bas possible est intéressant si on souhaite utiliser la méthode comme une heuristique en rétablissant l'intégralité sur les variables pour résoudre directement par un solveur le problème maître restreint. Un exemple d'une borne inférieure et d'une borne supérieure obtenues de cette manière par SOGA pour une instance de grande taille est illustré dans la figure 4.1. On obtient quasiment l'ensemble Pareto optimal dans son ensemble. Ce résultat est représentatif des observations sur les jeux d'instance.

TABLE 4.2 – Nombre de problèmes résolus par DSSR.

			|lb|				
p	$\|V\|$	$\|W\|$	moy.	dev. std.	PPS	IPPS	SOGA
5	30	60	25.0	3.2	184.6	124.0	112.2
5	30	90	20.8	4.8	163.4	119.6	105.6
5	40	80	26.0	1.9	228.0	155.2	141.4
5	40	120	27.2	4.9	330.4	200.8	173.8
5	50	100	31.6	3.0	390.4	226.4	212.6
5	50	150	29.8	3.4	486.4	246.6	223.6
8	30	60	25.4	3.1	226.8	152.0	136.0
8	30	90	22.2	4.0	215.2	142.2	121.8
8	40	80	27.0	2.9	302.0	217.8	182.8
8	40	120	29.4	4.3	480.6	293.2	243.2
8	50	100	32.2	2.8	522.0	335.4	289.0
8	50	150	29.8	3.8	672.0	384.0	305.6

TABLE 4.3 – Nombre de colonnes générées.

			|lb|				
p	$\|V\|$	$\|W\|$	moy.	dev. std.	PPS	IPPS	SOGA
5	30	60	25.0	3.2	1198.2	1809.2	1229.4
5	30	90	20.8	4.8	1063.4	1568.8	1041.2
5	40	80	26.0	1.9	1597.4	2098.6	1609.8
5	40	120	27.2	4.9	2570.8	3388.0	2347.6
5	50	100	31.6	3.0	3035.4	3459.0	2871.2
5	50	150	29.8	3.4	4052.8	4054.2	3087.4
8	30	60	25.4	3.1	1627.0	2234.8	1657.2
8	30	90	22.2	4.0	1551.6	2010.2	1284.0
8	40	80	27.0	2.9	2283.0	2961.4	2252.6
8	40	120	29.4	4.3	3949.4	4662.8	3652.4
8	50	100	32.2	2.8	4306.4	5070.6	4392.8
8	50	150	29.8	3.8	5799.4	6004.6	4781.8

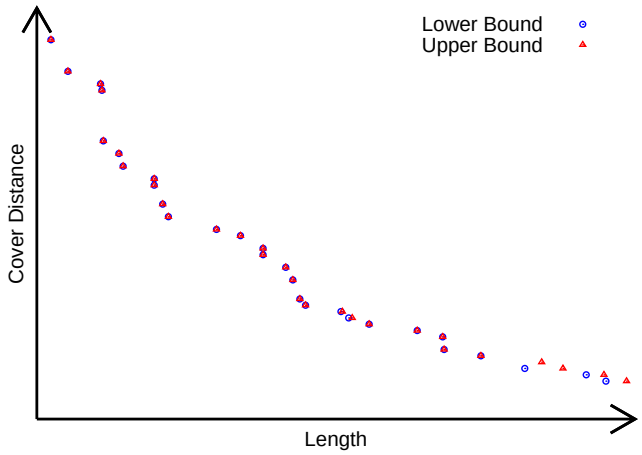


Fig. 4.1 – Borne inférieure et borne supérieure pour $|V| = 50, |W| = 150, p = 5, q = +\infty$.

Troisième partie

Application au spatial

Chapitre 5

Observation de la Terre par satellites agiles

Ce chapitre présente une application de l'optimisation multi-objectif dans le cadre du spatial. Il s'agit de la sélection et de l'ordonnancement de prises de vue par un satellite agile dans le contexte où des requêtes proviennent de plusieurs utilisateurs. On cherche donc à maximiser le profit associé aux prises de vue acquises tout en garantissant via un second objectif l'équité entre les utilisateurs. Différentes méthodes et mécanismes ont été testés pour ce problème. Pour des détails plus précis que dans ce document, le lecteur est invité à se référer à [130, 131, 129]. Ce chapitre rassemble les principales méthodes et les principaux résultats. Le problème est décrit brièvement dans 5.1. Puis, une première méthode, un algorithme génétique, est présentée dans 5.2. La section suivante porte sur une implémentation d'une méthode de recherche locale multi-objectif basée sur les indicateurs [10]. Les différentes méthodes sont comparées dans la section 5.4.

5.1 Problème et contexte

Ce chapitre traite d'un problème d'optimisation multi-objectif associé à la sélection et l'ordonnancement de prises de vue de la Terre par un satellite. La mission d'un tel satellite est de prendre des photographies de la surface de la Terre pour répondre à des demandes émanant d'utilisateurs. Les satellites passent une période de plusieurs jours pour faire un cycle d'orbite et les prises de vue doivent être acquises durant une orbite. La Terre a été vue dans son ensemble lorsque le satellite a complété un cycle complet [62]. Le satellite possède différents types d'équipements selon son utilisation (par exemple, caméra optique ou infrarouge). Ainsi, lorsque le satellite atteint la zone de visibilité d'une photographie requise, cette dernière peut être prise comme illustré dans la figure 5.1. Puis, le satellite tente de télécharger les images acquises directement à une station au sol par la suite. Si cela n'est pas possible, les données sont stockées dans une mémoire à bord de capacité limitée jusqu'à ce que le téléchargement soit possible.

Parmi les différents types de satellites, on considère ici les satellites dits "agiles". Un tel satellite est équipé d'une unique caméra mais le satellite peut tourner autour de 3 axes [91], ce qui permet au satellite de pouvoir prendre toutes les photographies avec une seule caméra. Un exemple est le PLEIADES développé par le CNES. Du fait de cette liberté de mouvement, la date de début d'une prise de vue n'est pas fixe mais elle doit avoir lieu dans un intervalle de

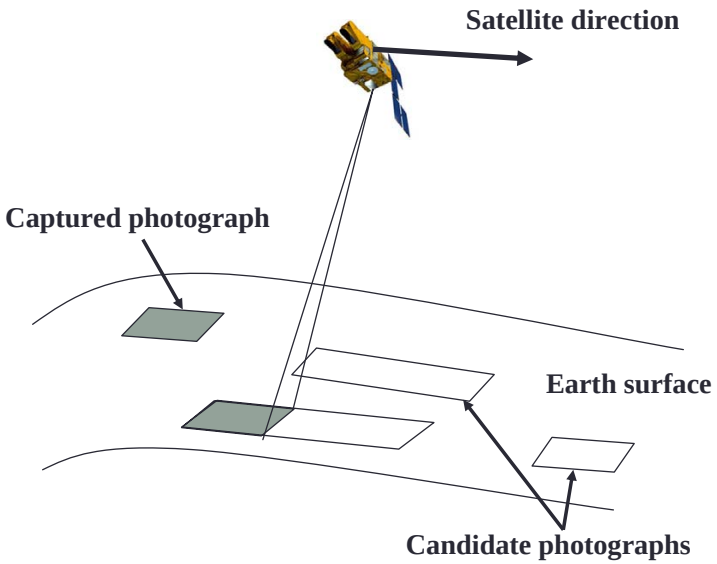


Fig. 5.1 – Prise de vue par le satellite [97].

temps appelé "fenêtre de temps". Ainsi, ce genre de satellites offre une meilleure efficacité du système complet, mais le problème de sélection et d'ordonnancement des prises de vue est plus complexe [92].

Le processus de gestion des satellites commence lorsque plusieurs utilisateurs font des requêtes à une station terrestre. Ces requêtes ne peuvent pas être affectées directement au satellite. La station doit gérer les demandes selon les limitations du satellite avant de lui transmettre une séquence d'acquisition. Si une requête est acquise par le satellite, elle engendre un "profit". Typiquement, la station essaie de maximiser le profit engendré par les acquisitions qu'elle a retenues. Il n'est pas possible d'effectuer toutes les acquisitions pour des problèmes de chevauchement des fenêtres de temps de deux requêtes ou de limitation matérielle du satellite par exemple.

Chaque requête peut être d'un des deux types suivants : mono ou stéréo. Une zone n'est prise qu'une fois pour les requêtes mono tandis que les requêtes stéréos doivent être prises deux fois en suivant le même sens de balayage mais selon un angle différent. Il y a aussi deux formes de requêtes différentes : les points et les polygones. Le point est une petite zone circulaire de moins de 10 kilomètres, tandis que le polygone est une zone de forme variable de 20 à 100 kilomètres. Les deux formes sont gérées via leur découpage en "strips". Un strip est une zone qui peut être prise en une seule passe par le satellite. Un point est considéré comme un seul strip. Un polygone est divisé en plusieurs strips. Les strips ont tous la même largeur, mais ils sont de longueurs différentes. Un exemple de découpage est donné dans la figure 5.2. Chaque strip peut être pris suivant deux directions. Chaque direction est parallèle à la longueur du strip mais dans un sens opposé comme montré dans la figure 5.3. Le strip associé à une direction d'acquisition est appelé une acquisition. Il y a donc deux acquisitions associées à chaque strip. Les intervalles de temps pour prendre chaque acquisition peuvent être calculés a priori.

Le problème est donc de sélectionner les acquisitions et de les ordonner en respectant les contraintes liées au satellite. La première contrainte est le respect des fenêtres de temps. La seconde contrainte consiste à respecter le temps nécessaire pour repositionner le satellite entre

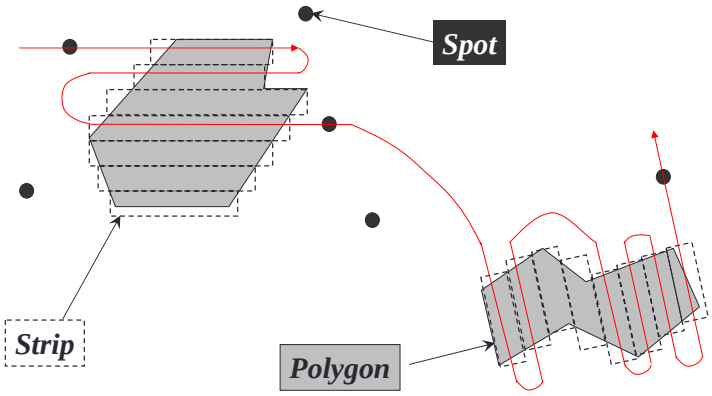


Fig. 5.2 – Exemple de découpage en strips et de spot [92].

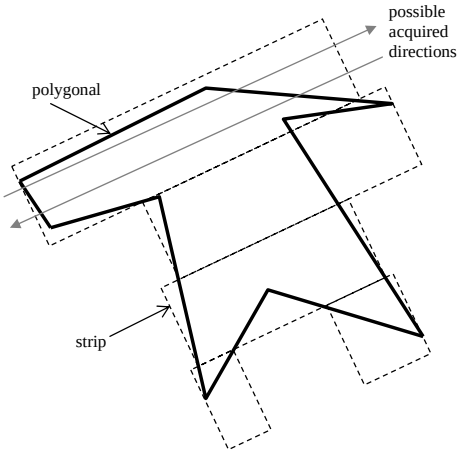


Fig. 5.3 – Découpage d'un polygone en strips et direction d'acquisition [92].

deux acquisitions planifiées. La troisième contrainte empêche de prendre les deux acquisitions associées à un même strip. La quatrième contrainte est liée aux requêtes stéréos. Elles sont représentées par quatre acquisitions (2 pour chacune des requêtes). Si l'une est sélectionnée, il faut alors aussi sélectionner l'acquisition associée à la seconde requête qui correspond à la même direction. Finalement, le satellite dispose d'une mémoire à bord limitée qu'il ne faut pas dépasser. Chaque acquisition génère un profit pré-établi. De ce fait, une solution au problème est évaluée par la somme des profits des acquisitions sélectionnées et ordonnées.

Il existe plusieurs études portant sur les satellites d'observation agiles. Lemaître et al. [92] ont proposé quatre méthodes pour une version simplifiée : un algorithme glouton, un programme dynamique, un modèle de programmation par contraintes et une recherche locale. Li et al. [94] ont eux proposé un recuit simulé et un algorithme génétique. Le challenge ROADEF 2003 [1] portait sur la gestion de mission de satellites agiles. Les données et les critères d'optimisation sont explicités dans [136]. Le vainqueur du challenge a utilisé un recuit simulé [83], tandis que les seconds ont employé une recherche Tabou [31]. L'ensemble des travaux considérait le problème comme un

1. `http://challenge.roadef.org/2003/en`

problème purement mono-objectif où seule la maximisation du profit engendré était considérée.

Ici, on considère le cas où plusieurs utilisateurs formulent les requêtes. Cela entraîne naturellement un souci d'équité entre les utilisateurs. On a donc affaire à un problème multi-objectif. D'un point de vue de l'utilisation de l'optimisation multi-objectif dans le spatial, on peut trouver plusieurs références dans la littérature [55, 140, 5]. Pour l'aspect d'équité, elle est discutée dans [90] et deux fonctions objectif liées à l'efficacité et à l'équité ont été proposées dans [11, 89]. Trois approches sont discutées. La première donne la priorité à l'équité, la seconde à l'efficacité et la dernière offre un ensemble de compromis entre les deux. Cependant les auteurs ne cherchent pas un ensemble de solutions non-dominées. Une recherche Tabou prenant en compte plusieurs satellites, plusieurs orbites et plusieurs utilisateurs est présentée dans [17]. L'équité y est prise en compte via l'utilisation de moyennes de poids pondérés issues de [143]. Dans ces travaux, l'équité est prise en compte en minimisant l'écart entre l'utilisateur ayant le plus grand profit et celui ayant le plus petit.

5.2 Algorithme génétique à clefs aléatoires biaisé

Dans cette section, nous allons présenter une implémentation d'un algorithme génétique à clefs aléatoires. Une particularité provient du fait qu'un vecteur-clef (une solution) est décodé de deux manières qui peuvent conduire à des solutions qui ne sont pas comparables entre elles d'un point de vue de la dominance de Pareto. Des mécanismes pour prendre en compte cela sont présentés.

5.2.1 Evaluation des solutions

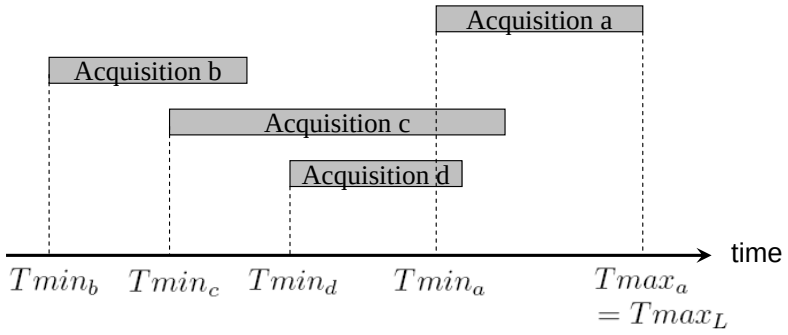
Pour évaluer et sélectionner les individus, plusieurs mécanismes issus de la littérature multi-objectif ont été employés et comparés. Il s'agit de la méthode de ranking de NSGA-II [39], la sélection par métrique $\mathcal{S}$ [16] et l'évaluation basée sur les indicateurs de IBEA [145]. Il a été établi expérimentalement [130] que c'est cette dernière approche qui offre les meilleurs résultats et nous nous limiterons à cette dernière dans ce document. La procédure donne un score entre 0 et 1 à chaque solution. Ensuite, on retire la plus mauvaise solution et on recalcule les scores jusqu'à ce que l'on ait obtenu la taille de population souhaitée. Les autres mécanismes de gestion de la population sont les mêmes que ceux de la méthode initiale et on a utilisé les valeurs préconisées par Gonçalves et Resende [59] pour les valeurs des paramètres.

5.2.2 Décodage basique

Chaque gène du vecteur-clef représente une acquisition. Ce codage est donc simple. Une première procédure de décodage est la suivante. On considère chaque acquisition selon une priorité qui est égale à la valeur de la clef du gène qui lui est associée. La solution est ensuite reconstruite de manière gloutonne en considérant en priorité les acquisitions de plus grande priorité et en écartant les acquisitions qui ne permettent pas de respecter les contraintes opérationnelles.

5.2.3 Décodage avec priorité idéale

Une deuxième procédure de décodage issue des problèmes d'ordonnancement de projets avec contraintes de ressources basée sur [101] a aussi été implémentée. La priorité d'une acquisition



Acquisition	a	b	c	d
Ideal priority	$\frac{Tmax_L - Tmin_a}{Tmax_L}$	$\frac{Tmax_L - Tmin_b}{Tmax_L}$	$\frac{Tmax_L - Tmin_c}{Tmax_L}$	$\frac{Tmax_L - Tmin_d}{Tmax_L}$

Fig. 5.4 – Exemple de calcul de priorité idéale.

est définie selon deux valeurs : la valeur du gène associé et la valeur de la priorité idéale calculée. Pour le concept de priorité idéale, la tâche qui a la date de début la plus tôt possible devrait être sélectionnée d'abord et ordonnancée au début de la séquence. Donc, on donne de cette manière une priorité plus forte aux tâches ayant les dates de début les plus tôt possibles. La priorité idéale d'une tâche j est donnée par

$$\frac{LLP_j}{LCP}, \tag{5.1}$$

où LLP_j est la longueur du plus long chemin entre le début de l'acquisition j et la fin du projet et LCP est la longueur du chemin critique du projet. On combine cette priorité à la valeur du gène en utilisant l'expression :

$$Priority_j = \frac{LLP_j}{LCP} \times \left[\frac{1 + gene_j}{2}\right]. \tag{5.2}$$

Dans [101], l'objectif est la minimisation du makespan. Ici, on la redéfinit en associant les acquisitions aux tâches et en prenant en compte nos objectifs. La priorité idéale d'une acquisition j est alors :

$$\frac{Tmax_L - Tmin_j}{Tmax_L}, \tag{5.3}$$

où $Tmax_L$ est la dernière date de début de la dernière acquisition possible et $Tmin_j$ est la date de début au plus tôt de l'acquisition j. La priorité de l'acquisition j est alors :

$$Priority_j = \frac{Tmax_L - Tmin_j}{Tmax_L} \times \left[\frac{1 + gene_j}{2}\right]. \tag{5.4}$$

Un exemple est donné dans la figure 5.4. Ici, la priorité idéale retournerait la séquence b, c, d, a.

5.2.4 Décodage hybride

Pour un vecteur-clef donné, les deux décodages ci-dessus ne vont pas donner la même solution et il est possible que les deux solutions ne soient pas comparables. Dans ce cas, il est nécessaire

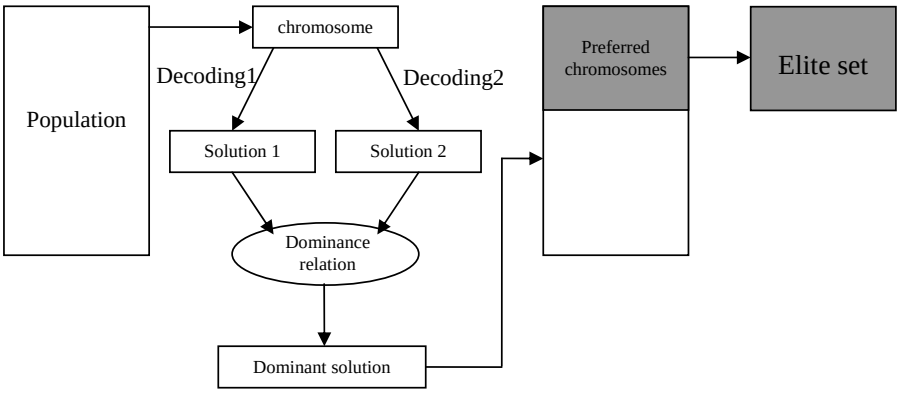


Fig. 5.5 – Gestion de l'ensemble de solutions élites lors du décodage hybride.

soit de faire un choix sur quel décodeur utiliser, soit d'utiliser les deux et de mettre en place des procédures pour cela. C'est la solution qui a été retenue ici. Dans un contexte mono-objectif, cela n'est pas nécessaire car une solution sera toujours préférée. Dans le décodeur hybride, on applique les deux décodeurs et on obtient deux solutions. Il est donc nécessaire de redéfinir les procédures pour sélectionner l'ensemble élite et l'ensemble des solutions à remplacer par les mutants. Différentes stratégies ont été envisagées et testées. Le lecteur est invité à se référer à [131] pour plus de détails. Nous présenterons ici la méthode qui offre les meilleurs résultats expérimentalement.

La procédure opère ainsi. On décode le vecteur-clef par les deux décodeurs. Si une solution domine la seconde, elle est sélectionnée pour être dans l'ensemble de solutions et la seconde solution est écartée. Si les deux solutions sont non-dominées l'une par l'autre, on garde une des deux solutions aléatoirement. Cette procédure est relativement simple mais obtient des résultats comparables en qualité avec des procédures plus complexes qui sont plus coûteuses en temps de calcul. D'autre part, les résultats sont meilleurs que si un seul des deux décodeurs est utilisé. La procédure est résumée dans la figure 5.5.

5.2.5 Vérification des contraintes durant le décodage

Durant le décodage, les acquisitions sont considérées de manière gloutonne selon leur ordre de priorité. Il faut cependant veiller à respecter les contraintes opérationnelles. La procédure de vérification pour l'insertion d'une acquisition est illustrée dans la figure 5.6. Un exemple pour deux acquisitions utilisant le décodage basique est reporté dans la figure 5.7. L'instance est formée de deux strips, soit quatre acquisitions possibles et donc 4 gènes.

5.3 Recherche locale multi-objectif basée sur des indicateurs

Une seconde méta-heuristique pour le problème a été développée. Il s'agit d'une implémentation d'une recherche locale multi-objectif basée sur les indicateurs (IBMOLS) [10]. IBMOLS est une recherche locale stéréo qui travaille sur une population. La sélection de l'individu à explorer dans la population courante est faite via un indicateur binaire. Dans le cadre de cette étude, nous utiliserons l'hypervolume comme indicateur. Lors de la première itération, un ensemble

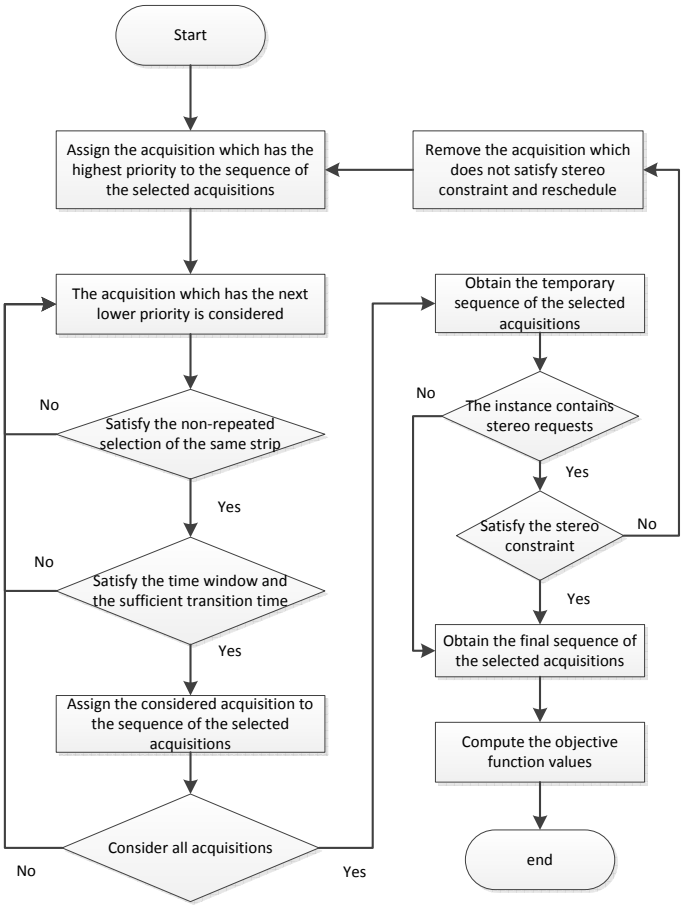


Fig. 5.6 – Vérification des contraintes et affectation des acquisitions.

potentiellement Pareto optimal (PO) vide est créé et il est mis à jour à la fin de chaque itération. Une itération de IBMOLS commence par générer une population initiale. Deux stratégies de génération de population seront présentées ici. La première est utilisée lors de la première itération, tandis que la seconde exploite les solutions déjà trouvées et sert dans les itérations suivantes. Les individus non-dominés de la population courante sont stockés dans une archive A. On applique alors l'étape de recherche locale sur la population courante. On tente de mettre à jour A avec les nouvelles solutions. Si A est modifiée, on recommence la phase de recherche locale. Sinon, l'itération courante est déclarée finie. On combine alors A et PO. Une nouvelle itération est lancée jusqu'à atteindre un critère d'arrêt (nombre d'itérations, temps de calcul ...). Nous allons maintenant décrire les implémentations faites pour notre problème. Il s'agit plus précisément des méthodes de génération de la population initiale à chaque itération et de la phase de recherche locale.

5.3.1 Génération de la population initiale - première itération

Lors de la première itération, N solutions sont générées aléatoirement. Une solution est une séquence d'acquisitions qui respecte les contraintes opérationnelles. La procédure est définie

Random-key chromosome	Acquisition0	Acquisition1	Acquisition2	Acquisition3
	0.6984	0.9939	0.6885	0.2509
Sequence of the selected acquisitions	1 | 2			
Total profit	1.04234e+007			
Maximum profit difference	0			

Fig. 5.7 – Exemple de solutions pour l'instance modifiée.

dans la figure 5.8. Les acquisitions sont prises en compte dans un ordre aléatoire. Pour chaque acquisition, on vérifie si elle peut être ajoutée dans la séquence actuelle en fonction des temps de transition et de sa fenêtre de temps. Si les deux contraintes sont satisfaites, on calcule son temps de début au plus tôt et on retire l'acquisition qui correspond au même strip mais dans le sens inverse des candidats. S'il s'agit d'une acquisition stéréo, on teste si on peut mettre la seconde acquisition dans la séquence. Si ce n'est pas le cas, on la retire. On itère jusqu'à ce que toutes les acquisitions sont soit incluses dans la séquence soit déclarées non candidates.

5.3.2 Génération de la population initiale - autres itérations

Lors des itérations suivantes, la population initiale est générée à l'aide d'un mécanisme de perturbation. Un nouvel individu est généré à partir d'une solution s de PO. On retire de s un quart des acquisitions planifiées. Le retrait d'une des deux acquisitions dans une demande stéréo entraîne le retrait de l'acquisition associée.

5.3.3 Etape de recherche locale

Cette étape commence par un individu dans la population courante P. A la différence de la méthode IBMOLS originale, on ne suit pas la stratégie de choisir la première amélioration mais on prend la meilleure amélioration dans le voisinage. La solution avec le meilleur fitness donné par un indicateur d'hypervolume binaire est sélectionnée et remplace la plus mauvaise solution dans la population courante. Quatre types de voisinage sont utilisés : i) insertion d'une acquisition mono ; ii) retrait d'une acquisition mono ; iii) insertion de deux acquisitions stéréos ; iv) retrait de deux acquisitions stéréos. La faisabilité des mouvements d'insertion est assurée en pré-calculant les dates de début au plus tard pour les acquisitions déjà ordonnées.

5.4 Résultats expérimentaux

Des instances (sous-ensemble A) du challenge ROADEF 2003 ont été modifiées pour répartir les acquisitions entre 4 utilisateurs. Dans la suite, le nom des instances est de la forme a_b_c avec a le nombre de requêtes, b le nombre de requêtes stéréos et c le nombre de strips. Les paramètres de l'algorithme génétique ont été fixés expérimentalement. Pour IBMOLS, la taille de la population a été fixée à 10 et le critère d'arrêt, le nombre d'itérations sans amélioration de PO, à 50. Les deux méthodes ont été implémentées en C++ et dix exécutions par instance ont été effectuées. Les résultats sont évalués par la mesure d'hypervolume. La comparaison entre les deux métaheuristiques est reportée ici. Les résultats sont reportés dans la figure 5.9. La

première colonne illustre les résultats de l'algorithme génétique et la seconde ceux d'IBMOLS. IBMOLS obtient les meilleures valeurs médianes pour l'hypervolume pour toutes les instances et une meilleure déviation standard pour la plupart des instances. De plus, IBMOLS est plus efficace en temps de calcul, spécialement pour les instances de grandes tailles. Dans la figure 5.10, l'amélioration des valeurs d'hypervolume par rapport aux temps de calculs pour les instances de moyennes et grandes tailles est illustrée. Les résultats montrent que IBMOLS obtient des approximations plus proches des meilleurs résultats trouvés et que la méthode converge aussi plus rapidement que l'algorithme génétique. Finalement, les meilleures approximations pour toutes les instances sont données dans la figure 5.11. Le profit total est reporté sur l'axe des abscisses et la différence de profits entre les utilisateurs sur l'axe des ordonnées.

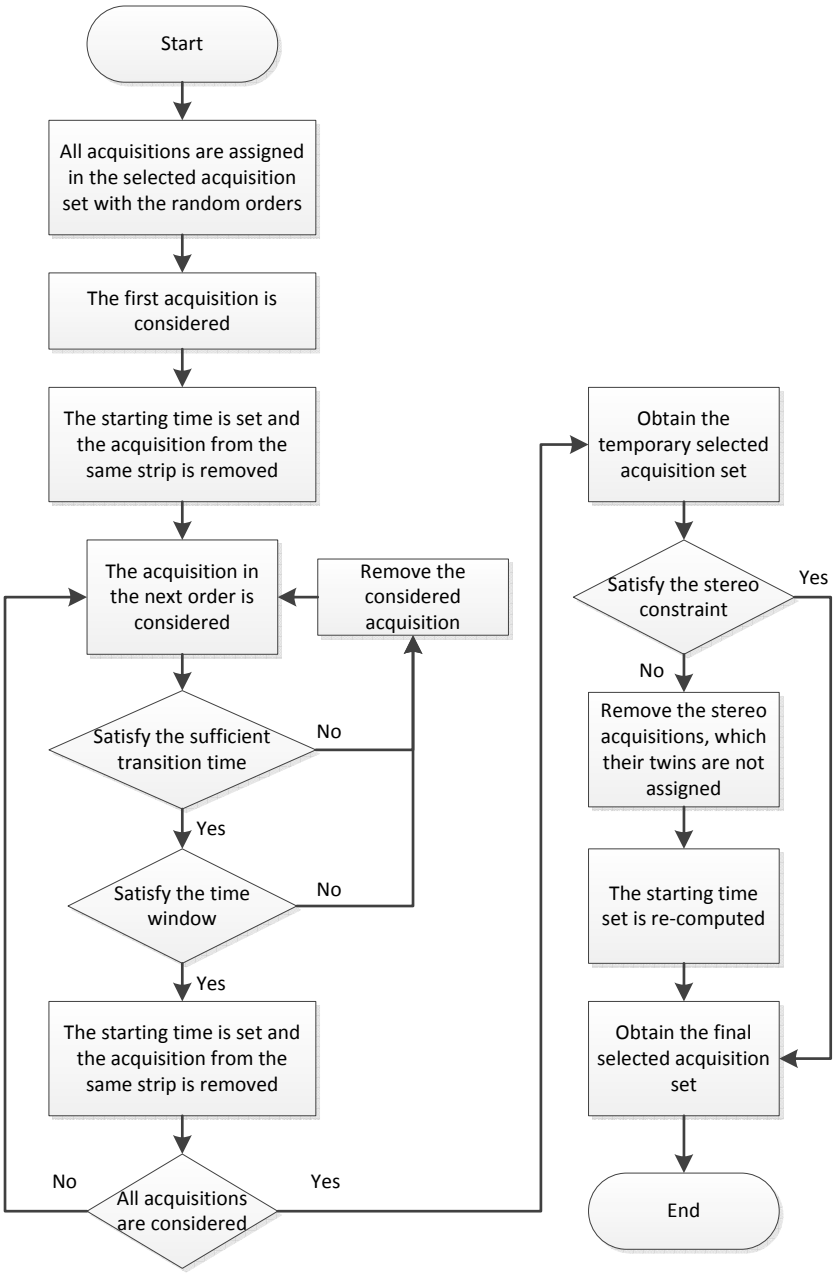


Fig. 5.8 – Génération aléatoire de la population initiale.

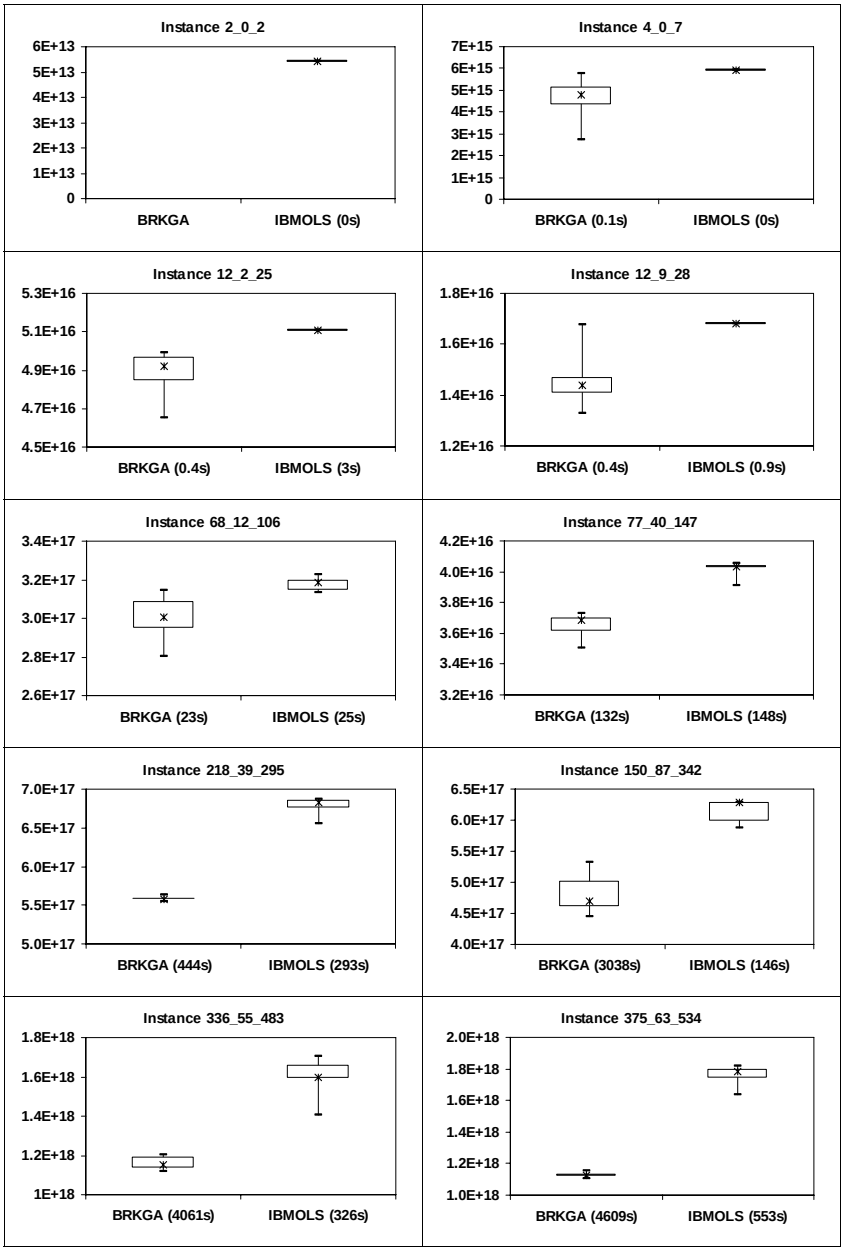


Fig. 5.9 – Comparaisons des valeurs d'hypervolume et des temps de calcul entre BRKGA et IBMOLS.

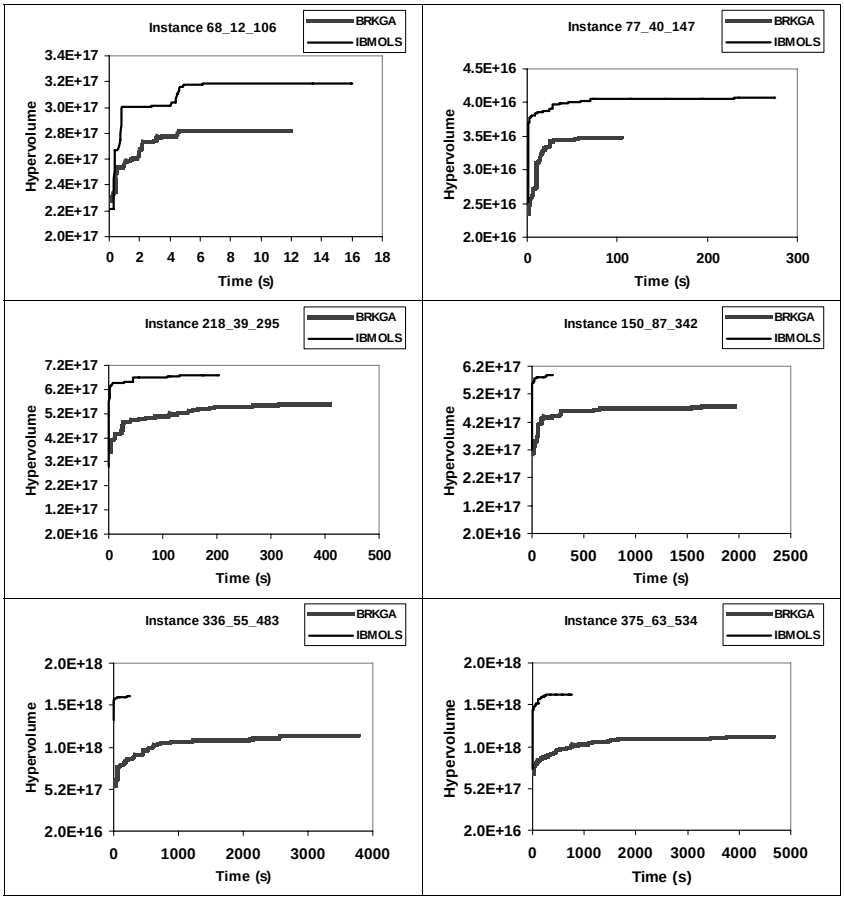


Fig. 5.10 – Comparaison de l'évolution des valeurs d'hypervolume en fonction d'un critère d'arrêt dynamique.

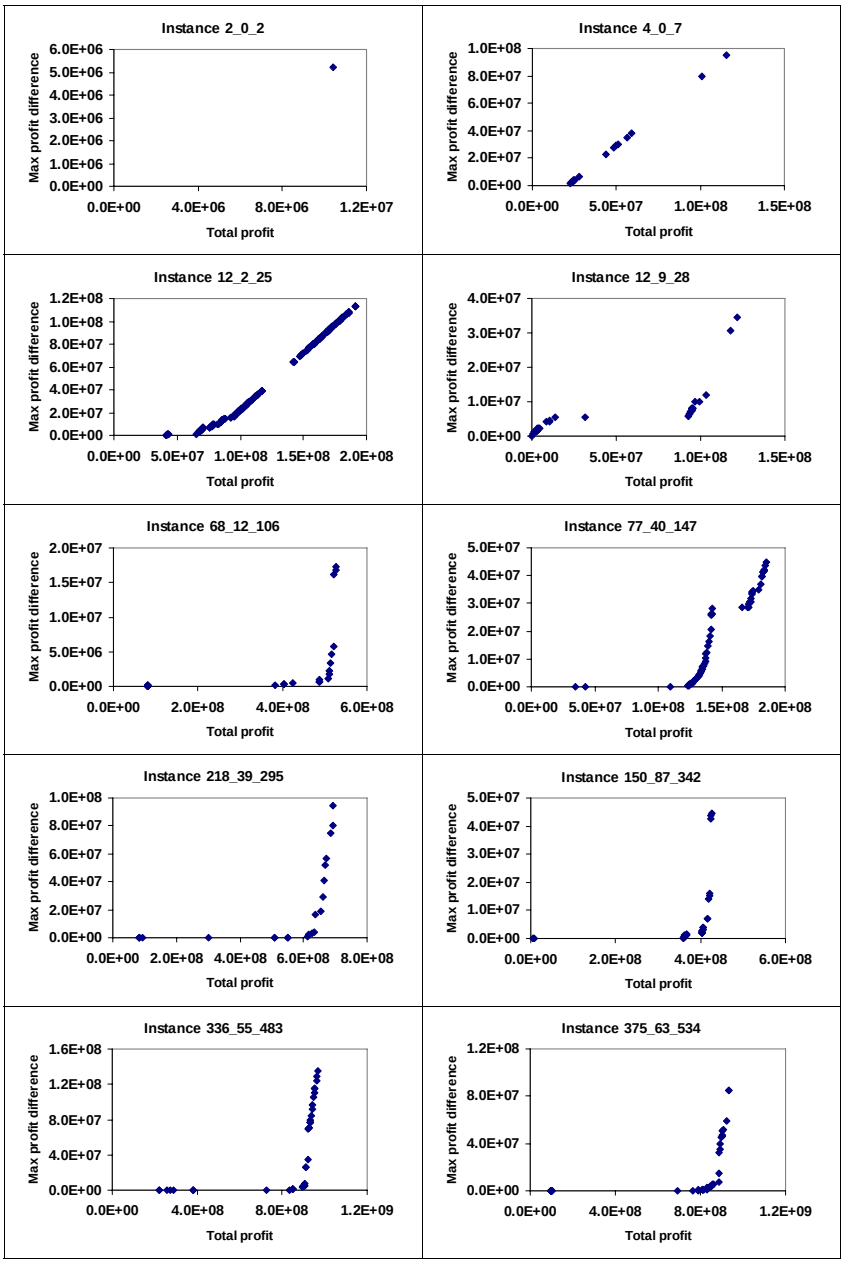


Fig. 5.11 – Meilleure approximation pour chaque instance.

Conclusions et perspectives

Ce mémoire porte sur l'optimisation combinatoire multi-objectif. Des contributions sont faites en termes de méthodes originales et d'applications à différents problèmes dans le transport terrestre et le spatial. Le but de ce chapitre est de fournir des conclusions générales et des perspectives de continuation sur l'ensemble des travaux et deux perspectives de recherche à long terme.

Le premier aspect évoqué dans ce domaine sont les méthodes d'optimisation multi-objectif. Si elles ont connu un essor important en 30 ans, elles sont longtemps restées focalisées sur l'utilisation de méta-heuristiques, notamment les algorithmes évolutionnistes. De nombreux mécanismes ont été proposés pour les rendre plus efficaces et répondre aux challenges posés par les problèmes multi-objectif. Cependant, la plupart des travaux se posent uniquement la question de l'évaluation des solutions et de la préservation de la diversité de l'ensemble de solutions retournées. Il y a à notre avis un manque de questionnement au niveau des opérateurs. En effet, si on regarde la littérature quels que soient les objectifs, les opérateurs sont le plus souvent classiques. Ils sont soit complètement déconnectés du problème, par exemple dans le cas où les solutions sont des vecteurs binaires. On peut alors douter de l'efficacité de la méthode. Soit empruntés à la littérature pour les problèmes mono-objectif, comme par exemple pour des problèmes de tournées de véhicules et ils sont souvent de fait efficaces pour un des objectifs uniquement. Ce document essaie de proposer une approche où l'opérateur est déconnecté du problème pour ne pas favoriser l'un ou l'autre objectif mais où un mécanisme permet d'évaluer une solution en prenant en compte l'ensemble des objectifs pour ne pas créer un déséquilibre entre eux. Des recherches continuant dans cette voie sont nécessaires notamment en utilisant des méta-heuristiques différentes des algorithmes génétiques comme par exemple des recherches à large voisinage adaptatives. En effet, l'utilisation de plusieurs heuristiques "simples" qui peuvent prendre en compte plus facilement l'aspect multi-objectif et les systèmes d'adaptabilité de sélection de ces méta-heuristiques sont une voie intéressante à explorer en conjonction avec les principes d'évaluation basée sur les indicateurs et de l'optimisation basée sur des ensembles.

Les autres contributions méthodologiques de ce document portent sur l'utilisation de la programmation mathématique, notamment la programmation linéaire en nombres entiers, pour la définition de méthodes exactes ou d'outils, comme le calcul de bornes inférieures, pour l'optimisation combinatoire multi-objectif. Au niveau des méthodes exactes, la plupart des méthodes existantes, notamment les premières, ont été influencées par les travaux d'Ulungu et Teghem qui portaient sur l'itération de méthodes exactes. Dans les travaux originaux, cela se comprenait car les problèmes obtenus à chaque itération étaient polynômiaux. Cependant, par la suite, la plupart des travaux ont continué à suivre ce schéma mais en itérant la résolution de problèmes NP-difficiles. Ces approches sont à notre avis vouées à une impasse. Un algorithme de

séparations et coupes multi-objectif a été proposé pour montrer qu'il n'est pas nécessaire de suivre forcément ce principe. La question importante est de savoir où il faut itérer. La différence entre la méthode proposée et les méthodes précédentes est qu'il a été choisi d'itérer un grand nombre de fois un problème simple, un programme linéaire ici, plutôt qu'un nombre restreint de fois un problème NP-difficile. Si la méthode reste très proche de la méthode mono-objectif, la présence de plusieurs objectifs a amené à définir des mécanismes particuliers comme des notions d'élagage partiel ou le branchement sur plusieurs variables. Ces mécanismes pourraient être affinés, voire de nouveaux mécanismes ou stratégies ont besoin d'être explorés comme par exemple des algorithmes de plans sécants multi-objectif. Un autre aspect important est le calcul de bornes inférieures que ce soit pour valider des méthodes existantes ou pour être utilisé dans les méthodes de recherche arborescente. Dans ce manuscrit, cette tâche est abordée via l'étude de l'utilisation du paradigme de génération de colonnes. Cette approche a été très peu explorée dans le cadre de l'optimisation combinatoire multi-objectif. Les travaux du document portent sur des stratégies de recherche de colonnes. Des perspectives sont ici multiples que ce soit l'exploration de nouvelles stratégies de recherche de colonnes, de mécanismes pour limiter les recherches inutiles, de l'intégration dans un algorithme de branch-and-price ... Les travaux sur l'algorithme de séparations et coupes et la génération de colonnes ont en commun l'idée qu'il est intéressant de traiter un problème multi-objectif via l'utilisation d'un programme linéaire en nombres entiers et d'une méthode de scalarisation. L'utilisation d'un PLNE permet de garder une représentation commune à l'ensemble des solutions et la méthode de scalarisation permet d'avoir un contrôle plus précis de la recherche avec un impact limité sur le PLNE puisqu'il s'agit juste de changer soit des coefficients dans la fonction objectif soit un membre droit dans des contraintes. Il est donc envisagé d'explorer comment rendre le travail plus "intelligent" via l'utilisation de l'analyse de sensibilité par exemple. Dans les travaux exposés dans ce mémoire, la méthode scalaire utilisée est le plus souvent la méthode ϵ-contrainte mais d'autres méthodes plus génériques et potentiellement plus simples à utiliser comme le simplexe paramétrique devront être envisagées. Enfin, concernant l'utilisation des méthodes basées sur la programmation linéaire en nombres entiers, une dernière voie à continuer d'explorer est leur utilisation comme heuristique. On peut par exemple essayer de définir des méthodes de type branchement local multi-objectif ou des heuristiques basées sur la génération de colonnes. Il est probable que ces méthodes puissent faire de la concurrence aux méthodes utilisant des mécanismes stochastiques comme les algorithmes génétiques via un meilleur contrôle de la recherche comme discuté ci-dessus.

Ce mémoire s'intéresse aussi à l'étude de problèmes qui servent notamment à illustrer l'utilisation des méthodes proposées. Ils illustrent aussi trois familles qui sont intéressantes à étudier dans un contexte multi-objectif. Il s'agit des problèmes de séquencement avec labels, des problèmes de tournées avec visites facultatives et des problèmes d'équilibrage. Les problèmes de séquencement avec labels sont illustrés ici par le problème du voyageur de commerce avec labels. De nombreux problèmes peuvent ici être étudiés en ajoutant des coûts au label, en définissant des coûts de changement entre les labels, sous forme de problèmes de tournées ou d'ordonnancement ... Les problèmes de tournées avec visites facultatives sont souvent étudiés. Une perspective serait de tenter de les unifier en un seul problème et de fournir une méthode efficace pour les résoudre tous. Enfin, les problèmes d'équilibrage peuvent poser des défis très intéressants à relever. Prenons le cas de problèmes de tournées de véhicules où l'équilibrage se fait entre les tournées en minimisant par exemple l'écart entre la plus longue tournée et la plus courte. Pour que la solution au problème ait un sens, il ne faut pas que cet équilibre soit "artificiellement" atteint en détériorant la tournée

la plus courte pour que sa longueur s'approche de la tournée la plus longue. Toutes les tournées de la solution ont besoin d'être optimales en termes de longueur pour que cela ait un sens. Cet aspect demande un intérêt particulier dans l'utilisation par exemple de la programmation linéaire en nombres entiers car cette optimalité ne peut pas être prise en compte via une simple contrainte ou un simple objectif. Dans les heuristiques, cela demande une réoptimisation des tournées régulièrement, ce qui peut se révéler coûteux, voire difficile selon les contraintes que l'on impose à la tournée.

Nous allons maintenant exposer trois perspectives de recherche : une sur les méthodes de recherche arborescente, une seconde sur la logistique collaborative et une dernière sur la prise en compte d'incertitudes en optimisation multi-objectif.

Recherche arborescente pour l'optimisation combinatoire multi-objectif

Une première perspective est de continuer à explorer l'utilisation de méthodes basées sur des arbres de recherche pour l'optimisation combinatoire multi-objectif. Les travaux présentés ici montrent qu'au-delà de la définition et du calcul de bornes inférieures, il y a des recherches à mener dans les mécanismes propres à ces méthodes : branchement et élagage. Des mécanismes génériques et faciles à implémenter ont besoin d'être explorés. Des recherches sur les méthodes de plans sécants pour l'optimisation multi-objectif sont aussi à envisager. A l'instar de la recherche de solutions de coût réduit négatif, il est intéressant de rechercher des coupes qui influent sur l'ensemble non-dominé complet. Il peut être par exemple intéressant d'explorer des mécanismes qui prennent leur décision non seulement en fonction des variables mais aussi de l'espace des objectifs comme dans [133]. A partir de là, l'utilisation des méthodes en tant qu'heuristique est la voie naturelle à suivre à l'image des méthodes locales de branchement [51]. La complexité des mécanismes à prendre en compte est telle qu'il faudra faire des recherches sur des moyens pour contenir les temps de calcul ; une voie a exploré notamment est l'utilisation du parallélisme [102, 103].

Logistique collaborative

La logistique collaborative est motivée par l'intérêt entre différents acteurs de mutualiser l'utilisation de ressources et la prise de décisions. En France, une étude récente du Ministère de l'Économie, des Finances et de l'Industrie [119] met en avant les intérêts d'une mutualisation des outils de la chaîne logistique entre différents acteurs : massification des flux (collecte en amont, consolidation/déconsolidation, distribution en aval) et la mutualisation des moyens (organisation, ressources humaines, moyens de transport, plates-formes, outils informatiques). La prise en compte de la logistique collaborative dans les entreprises est récente. Des premiers retours montrent des gains significatifs de l'ordre de 20% sur les coûts de transport, le niveau des stocks et les émissions de CO_2. Le transport de matières dangereuses et la logistique collaborative se prêtent très bien à des études multi-objectif via la présence d'objectifs naturels contradictoires. On peut citer en exemple l'opposition entre le coût du transport et la minimisation des risques pour le transport de matières dangereuses et le conflit entre coût économique et équité entre les acteurs en logistique collaborative. La logistique collaborative entraîne de nouvelles contraintes et de

nouveaux objectifs. Il y a donc besoin ici de définir de nouveaux modèles. C'est un aspect encore peu étudié de la chaîne logistique. On trouve des études en planification collaborative [124]. En transport, il s'agit principalement d'études portant sur des réseaux monomodaux avec une seule étude portant sur un cas de transport multimodal [113]. La définition de nouveaux modèles et méthodes est donc un enjeu important dans un domaine encore jeune et peu étudié. Les problèmes rencontrés se prêtent facilement à la définition de plusieurs objectifs. En plus des objectifs de coût standard, il est intéressant de prendre en compte avec la même importance des aspects liés aux clients (qualité de service, minimisation des risques ...), aux salariés (équité dans la charge de travail, respect des préférences ...), à la flotte (meilleure utilisation de l'ensemble de la flotte, meilleure adaptation de la flotte ...), aux marchandises (minimiser les stocks, minimiser le temps de transport pour les produits périssables ...), au consortium (équilibre entre les partenaires, maximisation des coûts communs, minimisation de l'échange d'information nécessaire en cas de données sensibles ...). Il est aussi important de prendre en compte de nouveaux objectifs comme ceux liés à l'environnement (minimisation des rejets de CO_2, minimisation de la consommation d'énergie non renouvelable ...) ou des impacts sur les infrastructures (minimisation de la congestion liée au trafic, minimisation de la taille occupée par l'infrastructure grâce à une minimisation des stocks nécessaires, une minimisation des besoins en ressources ...). Ces problèmes forment aussi des challenges permettant d'explorer la définition de nouvelles méthodes multi-objectif plus efficaces. Notamment dans la définition de méthodes rapides et robustes.

Optimisation multi-objectif avec incertitude

La prise en compte d'incertitudes est de plus en plus explorée en optimisation combinatoire. En optimisation multi-objectif, on trouve plusieurs études portant sur la présence d'incertitudes sous différentes formes et via l'utilisation de différentes méthodes [67, 38, 13, 60]. Cependant, elles restent peu nombreuses notamment en comparaison des travaux sur la prise en compte d'incertitudes en optimisation mono-objectif. Le but ici n'est pas de se placer dans un cadre "dynamique" où l'information se dévoile au fur et à mesure de l'optimisation. En effet, même s'il est possible de définir des méthodes multi-objectif a priori rapides pour prendre en compte en "temps réel" les nouvelles informations, il reste le problème de choisir une solution parmi l'ensemble de solutions non-dominées générées. On est ici plus dans l'optique de définir des méthodes pouvant être employées au niveau stratégique ou tactique et non au niveau opérationnel. Deux approches seront envisagées : la programmation stochastique [19] et l'optimisation robuste [82].

Programmation stochastique multi-objectif On s'intéressera aux problèmes avec recours de la forme :

$$\min_{x\in X}\{c^1x + Q_1(x), c^2x + Q_2(x) : Ax = b\}$$

avec

$$Q_1(x) = \mathbb{E}_\zeta[v_1(h^1(\omega) - T^1(\omega)x)], v_1(s) = \min_{y\in Y_1}\{q^1(\omega)y : W^1y = s\}$$

$$Q_2(x) = \mathbb{E}_\zeta[v_2(h^2(\omega) - T^2(\omega)x)], v_2(s) = \min_{y\in Y_2}\{q^2(\omega)y : W^2y = s\}$$

Trois types de problèmes peuvent se présenter : i) des problèmes où l'incertitude n'influe que sur un seul objectif, c'est-à-dire que seul un objectif comporte un mécanisme de recours ; ii) des

problèmes où l'incertitude influe sur les deux objectifs mais où les deux mécanismes de recours sont indépendants l'un de l'autre ; iii) des problèmes où l'incertitude influe sur les deux objectifs et où les deux mécanismes de recours forment eux-mêmes un problème bi-objectif. Si dans les deux premiers cas, pour une décision donnée au premier niveau, on obtient une solution après prise en compte de l'incertitude, dans le dernier cas, pour une solution du premier niveau, on peut avoir plusieurs solutions de compromis après prise en compte de l'incertitude.

La plupart des problèmes stochastiques multi-objectif sont traités en retirant soit l'aspect multi-objectif soit l'aspect stochastique via l'utilisation de techniques courantes [13]. Les méthodes préservant les aspects multi-objectif et stochastiques durant la totalité de la recherche sont rares (voir [60] pour un état de l'art récent). Pour les problèmes avec recours, la plupart des études (comme par exemple [53, 32, 132]) se limitent au cas (i) où on peut utiliser des approches classiques de scalarisation comme la méthode ϵ-contrainte [60]. Parmi les méthodes appliquées aux autres cas, on peut citer [1, 114].

On cherchera dans ces perspectives à adapter des méthodes issues de la programmation linéaire en nombres entiers à l'instar de ce qui a été fait pour les algorithmes de séparations et coupes dans [71] par exemple. On s'intéressera notamment à la méthode «Integer L-Shaped» [85]. Des problèmes de design de réseaux [32] serviront d'applications aux méthodes proposées.

Optimisation discrète robuste multi-objectif Dans le cadre de l'optimisation robuste, on ne considère pas de probabilité pour représenter l'incertitude. Une distribution probabiliste pour certains problèmes pouvant ne pas être facile à déterminer ou ne pas être réaliste. Ici, on considère l'incertitude par un (éventuellement très grand) ensemble de scénarios. Pour un scénario possible, on optimise un problème standard. Le but est donc de fournir une solution qui est valide pour tous les scénarios tout en minimisant une fonction objectif de pire cas. Il est par exemple possible de minimiser le plus grand coût sur tous les scénarios ou le plus grand regret, c'est-à-dire le plus grand écart possible avec l'optimum si le scénario réel était connu.

Il a été remarqué qu'il existe un lien entre l'optimisation robuste et l'optimisation multi-objectif [3, 24]. Il est possible par exemple de considérer les scénarios comme autant de différents objectifs. Cependant, on se retrouve ici avec des problèmes multi-objectif ayant un très grand nombre d'objectifs à résoudre. Il est aussi possible de considérer la fonction de regret comme un second objectif associé à un objectif évaluant la qualité globale de la solution. Dans ces perspectives, on souhaite s'intéresser à des problèmes où la robustesse n'est pas prise en compte dans les objectifs mais où on cherche un ensemble de solutions "non-dominées" robuste. A l'instar de la remarque sur les problèmes stochastiques, il peut y avoir plusieurs cas de figure selon que tous les objectifs ou seulement un sous-ensemble varient sur les scénarios. On cherchera à adapter des méthodes issues de l'optimisation mono-objectif en définissant des méthodes qui ne sont pas une itération de résolutions de problèmes obtenus par scalarisation. On explorera notamment des méthodes employant des relaxations de scénarios [18, 7].

Bibliographie

[1] M. Abbas and F. Bellahcene. Cutting plane method for multiple objective stochastic integer linear programming. *European Journal of Operational Research*, 168 :967–984, 2006.

[2] H. M. Afsar, N. Jozefowiez, and P. Lopez. A branch-and-price algorithm for the windy rural postman problem. *RAIRO-RO*, 45 :320–331, 2011.

[3] H. Aissi, C. Bazgan, and D. Vanderpooten. Min-max and min-max regret versions of combinatorial optimization problems : a survey. *European Journal of Operational Research*, 197 :427–438, 2009.

[4] M. J. Alves and J. Clímaco. A review of interactive methods for multiobjective integer and mixed-integer programming. *European Journal of Operational Research*, 180 :99–115, 2007.

[5] A. Arias-Montano, C. A. Coello Coello, and E. Mezura-Montes. Multi-objective evolutionary algorithms in aeronautical and aerospace engineering. *IEEE Transaction on Evolutionary Computation*, 16 :662–694, 2012.

[6] C. Artigues, N. Jozefowiez, and M. Ali Aloulou. An exact method for the bi-objective one-machine problem with maximum lateness and unit family setup cost objectives. In *Proceedings of ISCO 2010 - International Symposium on Combinatorial Optimization*, volume 36 of *Electronic Notes in Discrete Mathematics*, pages 1233–1240. Elsevier, 2010.

[7] T. Assavapokee, M. J. Realff, J. C. Ammons, and I. Hong. Scenario relaxation algorithm for finite scenario-based min-max regret and min-max relative regret robust optimization. *Computers & Operations Research*, 35 :2093–2102, 2008.

[8] E. Balas. The prize collecting traveling salesman problem. *Networks*, 19 :621–636, 1989.

[9] J. F. Bard and H. W. Purnomo. Preference scheduling for nurses using column generation. *European Journal of Operational Research*, 164 :514–534, 2005.

[10] M. Basseur and E. K. Burke. Indicator-based multi-objective local search. In *IEEE Congress on Evolutionary Computation (CEC 2007)*, pages 3100–3107, Singapore, 2007. IEEE Press.

[11] N. Bataille, M. Lemaître, and G. Verfaillie. Efficiency and fairness when sharing the use of a satellite. In *5th International Symposium on Artificial Intelligence, Robotics, and Automation in Space*, pages 465–470, 1999.

[12] J. C. Bean. Genetic algorithms and random keys for sequencing and optimizaton. *ORSA Journal on Computing*, 6 :154–160, 1994.

[13] F. Ben Abdelaziz. Solution approaches for the multiobjective stochastic programming. *European Journal of Operational Research*, 216 :1–16, 2012.

[14] H. Benson. Existence of efficient solutions for vector maximization problems. *Journal of optimization theory and applications*, 26 :569–580, 1978.

[15] J.-F. Bérubé, M. Gendreau, and J.-Y. Potvin. An exact ϵ-constraint method for bi-objective combinatorial optimization problems : Application to the travelling salesman problem with profits. *European Journal of Operational Research*, 194 :39–50, 2009.

[16] N. Beume, B. Naujoks, and M. Emmerich. SMS-EMOA : multi-objective selection based on dominated hypervolume. *European Journal of Operational Research*, 181 :1653–1669, 2007.

[17] N. Bianchessi, J.-F. Cordeau, J. Desrosiers, G. Laporte, and V. Raymond. A heuristic for the multi-satellite, multi-orbit and multi-user management of Earth observation satellites. *European Journal of Operational Research*, 177 :750–762, 2007.

[18] D. Bienstock and N. Özbay. Computing robust basestock levels. *Discrete optimization*, 5 :389–414, 2008.

[19] J. R. Birge and F. Louveaux. *Introduction to stochasting programming*. Springer Series in Operations Research and Financial Engineering. Springer, 2nd ed. edition, 2011.

[20] G. R. Bitran and J. M. Rivera. A combined approach to solve binary multicriteria problems. *Naval Research Logistics Quaterly*, 29 :181–201, 1982.

[21] L. Blander Reinhardt and D. Pisinger. Multi-objective and multi-constrained non-addictive shortest path problems. *Computers & Operations Research*, 28 :605–616, 2011.

[22] N. Boland, J. Dethridge, and I. Dumitrescu. Accelerated label setting algorithms for the elementary resource constrained shortest path problem. *Operations Research Letters*, 34 :58–68, 2006.

[23] S. Boussier, D. Feillet, and M. Gendreau. An exact algorithm for team orienteering problems. *4OR*, 5 :211–230, 2007.

[24] J. Branke, G. Avigad, and A. Moshaiov. Multi-objective worst case optimization by means of evolutionary algorithms. WBS working paper, WBS, University of Warwick, Coventry, UK, 2013.

[25] R. Carraway, T. L. Morin, and H. Moskowitz. Generalized dynamic programming for multicriteria optimization. *European Journal of Operational Research*, 44 :95–104, 1990.

[26] R. Cerulli, P. Dell'Olmo, M. Gentili, and A. Raiconi. Heuristic approaches for the minimum labelling hamiltonian cycle problem. *Electronic Notes in Discrete Mathematics*, 25 :131–138, 2006.

[27] V. Chankong and Y. Haimes. *Multiobjective decision making : Theory and methodology*. Elsevier Science Publishing Co., 1983.

[28] N. Christofides and S. Eilon. An algorithm for the vehicle dispatching problem. *Operational Research Quarterly*, 20 :309–318, 1969.

[29] N. Christofides, A. Mingozzi, P. Toth, and C. Sandi, editors. *Combinatorial Optimization*, chapter 11. John Wiley, Chichester, 1979.

[30] C. A. Coello Coello and D. A. Van Veldhuizen. *Evolutionary algorithms for solving multi-objective problems*. Springer, New York, NY, USA, 2nd ed. edition, 2007.

[31] J.-F. Cordeau and G. Laporte. Maximizing the value of an Earth observation satellite orbit. *Journal of the Operational Research Society*, 56 :962–968, 2005.

[32] Y. Cordora-Valdés, A. Alvarez, and D. Ozdemir. A bi-objective supply chain design problem with uncertainty. *Transportation Research Part C*, 19 :821–832, 2011.

[33] G. B. Dantzig, D. R. Fulkerson, and S. Johnson. Solution of a large-scale traveling salesman problem. *Operations Research*, 2 :393–410, 1954.

[34] I. Das and J. Dennis. A closer look at drawbacks of minimizing weighted sums of objectives for Pareto set generation in multicriteria optimization problems. *Structural Optimization*, 14 :63–69, 1997.

[35] T. Dean and M. Boddy. An analysis of time-dependent planning. In *AAAI-88*, pages 49–54, Menlo Park, CA, 1988. Association for the Advancement of Artificial Intelligence.

[36] K. Deb. *Multi-objective optimization using evolutionary algorithms*. Wiley, 2001.

[37] K. Deb. A robust evolutionary framework for multi-objective optimization. In *Genetic and Evolutionary Computation Conference (GECCO 2008)*, pages 633–640, Atlanta, GA, USA, 2008. ACM.

[38] K. Deb and H. Gupta. Introducing robustness in multi-objective optimization. *Evolutionary Computation*, 14 :463–494, 2006.

[39] K. Deb, A. Pratap, S. Agarwal, and T. Meyarvan. A fast and elitist multiobjective genetic algorithm : NSGA II. *IEEE Transaction on Evolutionary Computation*, 6 :182–197, 2002.

[40] M. Dell'Amico, F. Maffioli, and P. Värbrand. On prize-collecting travelling salesman problem. *International Transactions in Operational Research*, 2 :297–308, 1995.

[41] C. Dhaenens, J. Lemesre, and E. Talbi. K-PPM : A new exact method to solve multi-objective combinatorial optimization problems. *European Journal of Operational Research*, 200 :45–53, 2011.

[42] F. Y. Edgeworth. *Mathematical physics*. P. Keagan, London, 1881.

[43] M. Ehrgott. *Multicriteria optimization*, volume 491 of *Lecture Notes in Economics and Mathematical Systems*. Springer, 2nd ed. edition, 2005.

[44] M. Ehrgott and X. Gandibleux. *Multiple criteria optimization : State of the art annotated biobliographic surveys*, chapter Multiobjective combinatorial optimization. Kluwer, Boston, 2002.

[45] M. Ehrgott and X. Gandibleux. Bound sets for biobjective combinatorial optimization problems. *Computers & Operations Research*, 34 :2674–2694, 2007.

[46] M. Ehrgott, D. Tenfelde-Podehl, and T. Stephan. A level set method for multiobjective combinatorial optimization : Application to the quadratic assignment problem. *Pacific Journal on Optimization*, 2 :521–544, 2006.

[47] M. Ehrgott and J. Tind. Column generation in integer programming with applications in multicriteria optimization. Report 651, University of Auckland, School of Engineering, 2007.

[48] D. Feillet. A tutorial on column generation and branch-and-price for vehicle routing problems. *4OR*, 8(407-424), 2010.

[49] D. Feillet, P. Dejax, and M. Gendreau. Traveling salesman problem with profits. *Transportation Science*, 39 :188–205, 2005.

[50] D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints : Application to some vehicle routing problems. *Networks*, 44 :216–229, 2004.

[51] M. Fischetti and A. Lodi. Local branching. *Mathematical programming*, 98 :23–37, 2003.

[52] C. Fonseca and P. J. Flemming. Multiobjective optimization and multiple constraint handling with evolutionary algorithms. *IEEE Transaction on Systems, Man and Cybernetics*, 28 :38–47, 1998.

[53] M. Fonseca, A. Garciá-Sánchez, M. Ortega-Mier, and F. Saldanha da Gama. A stochastic bi-objective location model for strategic reverse logistics. *TOP*, 18 :158–184, 2010.

[54] M. Fourman. Compaction of symbolic layout using genetic algorithms. In *First International Conference on Genetic Algorithm*, pages 141–153, 1985.

[55] V. Gabrel and D. Vanderpooten. Enumeration and interactive selection of efficient paths in a multiple criteria graph for scheduling an Earth observing satellite. *European Journal of Operational Research*, 139 :533–542, 2002.

[56] M. Gendreau, A. Hertz, and G. Laporte. A tabu search heuristic for the vehicle routing problem. *Management Science*, 40 :1276–1290, 1994.

[57] M. Gendreau, G. Laporte, and F. Semet. The covering tour problem. *Operations Research*, 45 :568–576, 1997.

[58] D. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Publishing Company, Reading, Massachussett, USA, 1989.

[59] J. F. Gonçalves and M. G. C. Resende. Biased random-key genetic algorithms for combinatorial algorithms. *Journal of Heuristics*, 17 :487–525, 2011.

[60] W. J. Gutjahr and A. Pichler. Stochastic multi-objective optimization : a survey on non-scalarizing methods. *Annals of Operations Research*, 2013.

[61] M. H. Hà, N. Bostel, A. Langevin, and L.-M. Rousseau. An exact algorithm and a metaheuristic for the multi-vehicle covering tour problem with a constraint on the number of vertices. *European Journal of Operational Research*, 226 :211–220, 2013.

[62] D. Habet, M. Vasquez, and Y. Vimont. Bounding the optimum for the problem of scheduling the photographs of an agile earth observing satellite. *Computational Optimization and Applications*, 47 :307–333, 2010.

[63] M. Hachicha, M. J. Hodgson, G. Laporte, and F. Semet. Heuristics for the multi-vehicle covering tour problem. *Computers & Operations Research*, 27 :29–42, 2000.

[64] Y. Haimes, L. Lasdon, and D. Wismer. On a bicriterion formulation for the problems of integrated system identification and system optimization. *IEEE Transaction on Systems, Man and Cybernetics*, 1 :296–297, 1971.

[65] M. J. Hodgson, G. Laporte, and F. Semet. A covering tour model for planning mobile health care facilities in Suhum district, Ghana. *Journal of Regional Science*, 38 :621–638, 1998.

[66] S. Irnich and D. Villeneuve. The shortest path problem with resource constraints and k-cycle elimination for $k > 3$. *INFORMS Journal on Computing*, 18 :391–406, 2006.

[67] Y. Jin and J. Branke. Evolutionary algorithms in uncertain environments - a survey. *IEEE Transaction on Evolutionary Computation*, 9 :303–317, 2005.

[68] N. Jozefowiez. A branch-and-price algorithm for the multivehicle covering tour problem. Rapport LAAS 12686, LAAS-CNRS, Toulouse, France, 2012.

[69] N. Jozefowiez, F. Glover, and M. Laguna. Multi-objective meta-heuristics for the traveling salesman problem with profits. *Journal of Mathematical Modelling And Algorithms*, 7 :177–195, 2008.

[70] N. Jozefowiez, G. Laporte, and F. Semet. A branch-and-cut algorithm for the minimum labeling Hamiltonian cycle problem and two variants. *Computers & Operations Research*, 38 :1534–1542, 2011.

[71] N. Jozefowiez, G. Laporte, and F. Semet. A generic branch-and-cut algorithm for multiobjective optimization problems : Application to the multilabel traveling salesman problem. *INFORMS Journal on Computing*, 24 :554–564, 2012.

[72] N. Jozefowiez, C. Mancel, and F. Mora-Camino. A heuristic approach based on shortest path problems for integrated flight, aircraft, and passenger rescheduling under disruptions. *Journal of the Operational Research Society*, 64 :384–395, 2013.

[73] N. Jozefowiez, F. Semet, and E. Talbi. Target aiming Pareto search and its application to the vehicle routing problem with route balancing. *Journal of heuristics*, 13 :455–469, 2007.

[74] N. Jozefowiez, F. Semet, and E. Talbi. An evolutionary algorithm for the vehicle routing problem with route balancing. *European Journal of Operational Research*, 195 :761–769, 2009.

[75] N. Jozefowiez, F. Semet, and E.-G. Talbi. The bi-objective covering tour problem. *Computers & Operations Research*, 34 :1929–1942, 2007.

[76] S. Kataoka and S. Morito. An algorithm for the single constraint maximum collection problem. *Journal of the Operational Research Society of Japan*, 31 :515–530, 1988.

[77] A. Khanafer, F. Clautiaux, S. Hanafi, and E.-G. Talbi. The min-conflict packing problem. *Computers & Operations Research*, 39 :2122–2132, 2012.

[78] G. Kiziltan and E. Yucaoglu. An algorithm for multiobjective zero-one linear programming. *Management Science*, 29 :1444–1453, 1983.

[79] J. Knowles and D. Corne. Pareto archived evolution strategy : A new baseline algorithm for Pareto multiobjective optimisation. In *1999 Congress on Evolutionary Computaton (CEC'99)*, volume 1, pages 98–105, 1999.

[80] J. Knowles and D. Corne. On metrics for comparing nondominated sets. In *Congress on Evolutionary Computation (CEC'2002)*, volume 1, pages 711–716. IEEE Service Center, 2002.

[81] J. D. Knowles. *Local-search and hybrid evolutionary algorithms for Pareto optimization.* PhD thesis, University of Reading, UK, 2002.

[82] P. Kouvelis and G. Yu. *Robust discrete optmization and its applications.* Kluwer Academic Publishers, 1997.

[83] E. J. Kuipers. An algorithm for selection and timetabling requests for an Earth observation satellite. Bulletin de la Société Française de Recherche Opérationnelle, 2003.

[84] M. Labbé, G. Laporte, I. Rodríguez Martín, and J. J. Salazar González. The ring star problem : Polyhedral analysis and exact algorithm. *Networks*, 43 :177–189, 2004.

[85] G. Laporte and F. V. Louveaux. The integer L-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters*, 13 :133–142, 1993.

[86] G. Laporte and S. Martello. The selective traveling salesman problem. *Discrete applied mathematics*, 26 :193–207, 1990.

[87] M. Laumanns, L. Thiele, and E. Zitzler. An adaptive scheme to generate the Pareto front based on the epsilon-constraint method. In J. Branke, K. Deb, K. Miettinen, and R. E. Steuer, editors, *Practical approaches multi-objective optimization*, number Seminar 04461. Daghstuhl seminar, 2005.

[88] M. Laumanns, E. Zitzler, and L. Thiele. A unified model for multio-objective evolutionnary algorithms with elitism. In *IEEE Congress on Evolutionary Computation (CEC 2000)*, pages 46–43, Piscataway, New Jersey, USA, 2000. IEEE Press.

[89] M. Lemaître, G. Verfaillie, and N. Bataille. Efficiency and fairness when sharing the use of a satellite. In *16th International Joint Conference on Artificial Intelligence*, volume 1, pages 206–211, 1999.

[90] M. Lemaître, G. Verfaillie, H. Fargier, J. Lang, N. Bataille, and J. M. Lachiver. Sharing the use of Earth observation satellites. In *3rd NASA workshop on planning and scheduling*, 2002.

[91] M. Lemaître, G. Verfaillie, and F. Jouhaud. How to manage the new generation of Agile Earth Observation Satellites. In *6th International SpaceOps Symposium (Space Operations)*, Toulouse, France, 2000.

[92] M. Lemaître, G. Verfaillie, F. Jouhaud, J. M. Lachiver, and N. Bataille. Selecting and scheduling observations of agile satellites. *Aerospace Science and Technology*, 6 :367–381, 2002.

[93] J. Lemesre, C. Dhaenens, and E.-G. Talbi. Parallel partitioning method (PPM) : A new exact method to solve bi-objective problem. *Computers & Operations Research*, 34 :2450–2462, 2007.

[94] Y. Li, M. Xu, and R. Wang. Scheduling observations of agile satellites with combined genetic algorihm. In *Third International Conference on Natural Computation (ICNC 2007)*, volume 3, pages 29–33, 2007.

[95] A. Liefooghe, L. Jourdan, N. Jozefowiez, and E.-G. Talbi. On the integration of a tsp heuristic into an EA for the bi-objective ring star problem. In *Hybrid Metaheuristics*, volume 5296 of *Lecture Notes in Computer Science*, pages 117–130. Springer, 2008.

[96] F. Lootsma, T. Athan, and P. Papalambros. Controlling the search for a compromise solution in multi-objective optimization. *Enginering Optimization*, 25 :65–81, 1995.

[97] M. A. A. Mansour and M. M. Dessouky. A genetic algorithm approach for solving the daily photograph selection problem of SPOT 5 satellite. *Computers & Industrial Engineering*, 58 :509–520, 2010.

[98] E. Martins. On a multicriteria shortest path problem. *European Journal of Operational Research*, 16 :236–237, 1984.

[99] G. Mavrotas and D. Diakoulaki. A branch and bound algorithm for mixed zero-one multiple objective linear programming. *European Journal of Operational Research*, 107 :530–541, 1998.

[100] G. Mavrotas and D. Diakoulaki. Multi-criteria branch-and-bound : A vector maximization algorithm for mixed 0-1 multiple objective linear programming. *Applied mathematics and computation*, 171 :53–71, 2005.

[101] J. J. M. Mendes, J. F. Gonçalves, and M. G. C. Resende. A random key based genetic algorithm for the resource constrained project scheduling problem. *Computers & Operations Research*, 36 :92–109, 2009.

[102] M. Mezmaz, N. Melab, and E. Talbi. Combining metaheuristics and exact methods for solving exactly multi-objective problems on the grid. *Journal of Mathematical Modelling And Algorithms*, 6 :393–409, 2007.

[103] M. Mezmaz, N. Melab, and E. Talbi. An efficient load balancing balancing strategy for grid-based branch and bound algorithm. *Parallel Computing*, 33 :302–313, 2007.

[104] G. L. Nemhauser and L. A. Wolsey. *Integer and combinatorial optimization.* Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons, 1999.

[105] S. Ngueveu, C. Prins, and R. Calvo. An effective memetic algorithm for the cumulative capacitated vehicle routing problem. *Computers & Operations Research*, 27 :1877–1885, 2010.

[106] O. Özpeynirci and M. Köksalan. An exact algorithm for finding extremme supported nondominated points of multiobjective mixed integer programs. *Management Science*, 56 :2302–2315, 2010.

[107] L. Paquete and T. Stützle. A study of stochastic local search algorithms for the biobjective QAP with correlated flow matrices. *European Journal of Operational Research*, 169 :943–959, 2006.

[108] V. Pareto. *Cours d'économie politique.* Rouge, Lausanne, Suisse, 1896.

[109] C. Prins. A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers and Operations Research*, 31 :1985–2002, 2004.

[110] A. Przybylski, X. Gandibleux, and M. Ehrgott. Two phase algorithms for the bi-objective assignment problem. *European Journal of Operational Research*, 185 :509–533, 2008.

[111] A. Przybylski, X. Gandibleux, and M. Ehrgott. A recursive algorithm for finding all nondominated extreme points in the outcome set of a multiobjective integer program. *INFORMS Journal on Computing*, 22 :371–386, 2010.

[112] A. Przybylski, X. Gandibleux, and M. Ehrgott. A two phase method for multi-objective integer programming. *Discrete optimization*, 7 :149–165, 2010.

[113] C. Puettmann and H. Stadtler. A collaborative planning approach for intermodal freight transportation. *OR Spektrum*, 32 :809–830, 2010.

[114] S. Rath, M. Gendreau, and W. J. Gutjahr. Bi-objective stochastic programming models for determining depot locations in disaster relief operations planning. Technical report, 2012.

[115] G. Ribeiro and G. Laporte. An adpative large neighborhood search heuristic for the cumulative vehicle routing problem. *Computers & Operations Research*, 29 :728–735, 2012.

[116] G. Righini and M. Salani. New dynamic programming algorithms for the resource constrained elementary shortest path problem. *Networks*, 51 :155–170, 2008.

[117] L.-M. Rousseau, M. Gendreau, and D. Feillet. Interior point stabilization for column generation. *Operations Research Letters*, 35 :660–668, 2007.

[118] E. Salari and J. Unkelbach. A column-generation-based method for multi-criteria direct aperture optimization. *Physics in Medecine and Biology*, 58 :621–639, 2013.

[119] K. Salmon. Pratiques de logistique collaborative : quelles opportunités pour les PME/ETI. Etude, PIPAME, Ministère de l'Economie, des Finances et de l'Industrie, Paris, France, Mars 2011.

[120] B. M. Sarpong, C. Artigues, and N. Jozefowiez. The bi-objective multi-vehicle covering tour problem : Formulation and lower bound by column generation. Rapport LAAS 12562, LAAS-CNRS, Toulouse, France, 2012.

[121] B. M. Sarpong, C. Artigues, and N. Jozefowiez. Column generation for bi-objective vehicle routing problems with a min-max objective. In *13th Workshop on Algorithmics Approaches for Transportation Modelling, Optimization, and Systems*, OASIcs, 2013.

[122] D. Schaffer. Multiple objective optimization with vector evaluated genetic algorithm. In *First International Conference on Genetic Algorithm*, pages 93–100, 1985.

[123] F. Sourd and O. Spanjaard. A multiobjective branch-and-bound framework : Application to the biobjecitve spanning tree problem. *INFORMS Journal on Computing*, 20 :435–449, 2008.

[124] H. Stadtler. A framework for collaborative planning and state-of-the-art. *OR Spektrum*, 31 :5–30, 2009.

[125] R. E. Steuer. *Multiple criteria optimization : Theory, computation and application.* John Wiley, New York, 1986.

[126] E. Talbi, M. Rahoual, M. Mabed, and C. Dhaenens. A hybrid evolutionary approach for multicriteria optimization problems : Applications to the flow shop. In E. Zitzler et al., editors, *Evolutionary Multi-Criterion Optimization*, volume 1993 of *Lecture Notes in Computer Science*, pages 416–428. Springer, 2001.

[127] E.-G. Talbi. *Metaheuristics : From design to implementation.* Wiley-Blackwell, 2009.

[128] M. Tamiz and D. Jones. Goal programming and Pareto efficiency. *Journal of Information & Optimization Sciences*, 17 :291–307, 1996.

[129] P. Tangpattanakul. *Multi-objective optimization of Earth observing satellite mission.* PhD thesis, Institut National des Sciences Appliquées de Toulouse, Université de Toulouse, 2013.

[130] P. Tangpattanakul, N. Jozefowiez, and P. Lopez. Multi-objective optimization for selecting and scheduling observations by agile Earth observing satellites. In C. A. Coello Coello, V. Cutello, K. Deb, S. Forrest, G. Nicosia, and M. Pavone, editors, *Parallel Problem Solving from Nature (PPSN XII)*, volume 7492 of *Lecture Notes in Computer Science*, pages 112–121. Springer, 2012.

[131] P. Tangpattanakul, N. Jozefowiez, and P. Lopez. Biased random key genetic algorithm with hybrid decoding for multi-objective optimization. In *Workshop on Combinatorial Optimization 2013*. IEEE Service Center, 2013.

[132] F. Tricoire, A. Graf, and W. J. Gutjahr. The bi-objective stochastic covering tour problem. *Computers & Operations Research*, 39 :1582–1592, 2012.

[133] F. Tricoire and S. N. Parragh. Bound sets for the biobjective team orienteeing problem with time windows. In *VeRoLog 2013*, Southampton, UK, 2013.

[134] E. L. Ulungu and J. Teghem. The two phases method : An efficient procedure to solve biobjective combinatorial optimization problems. *Foundation of Computer Decision Science*, 20 :149–165, 1995.

[135] F. Vanderbeck. Computational study of a column generation algorithm for bin packing and cutting stock problems. *Mathematical programming*, 86 :565–594, 1999.

[136] G. Verfaillie, M. Lemaître, N. Bataille, and J. M. Lachiver. Management of the mission of Earth observation satellites : Challenge description. Technical report, Centre National d'Etude Spatiale, France, 2002.

[137] B. Villarreal and M. H. Karwan. Multicriteria integer programming : A (hybrid) dynamic programming recursive approach. *Mathematical programming*, 21 :204–223, 1981.

[138] T. Vincent, F. Seipp, S. Ruzika, A. Przybylski, and X. Gandibleux. Multiple objective branch and bound for mixed 0-1 linear programming : Corrections and improvements for the biobjective case. *Computers & Operations Research*, 40 :498–509, 2013.

[139] M. Visée, J. Teghem, M. Pirlot, and E. Ulungu. Two-phases method and branch and bound procedures to solve the biobjective knapsack problem. *Journal of Global Optimization*, 12 :139–155, 1998.

[140] J. Wand, N. Jing, J. Li, and Z. H. Chen. A multi-objective imaging scheduling approach for Earth observing satellites. In *9th annual conference on Genetic and Evolutionary Computation (GECCO'07)*, pages 2211–2218, 2007.

[141] L. A. Wolsey. *Integer programming*. Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons, 1998.

[142] Y. Xiong, B. Golden, and E. Wasil. *Extending the horizons : Advances in computing, optimization, and decision technologies*, chapter The colorful traveling salesman problem, pages 115–123. Springer, Boston, 2007.

[143] R. R. Yager. On order weighted averaging aggregation operators in multicriteria decisionmaking. *IEEE Transaction on Systems, Man and Cybernetics*, 18 :183–190, 1988.

[144] E. Zitzler. *Evolutionary algorithms for multiobjective optimization : Methods and applications*. PhD thesis, Ecole Polytechnique Fédérale (ETH), Zurich, Suisse, 1999.

[145] E. Zitzler and S. Künzli. Indicator-based selection in multiobjective search. In X. Yao et al., editors, *Parallel Problem Solving from Nature (PPSN VIII)*, volume 3242 of *Lecture Notes in Computer Science*, pages 832–842. Springer-Verlag, 2004.

[146] E. Zitzler, M. Laumanns, and S. Bleuler. A tutorial on evolutionary multiobjective optimization. In *Metaheuristics for Multobjective Optimization*, volume 535 of *Lecture Notes in Economics and Mathematical Systems*, pages 3–38. Springer-Verlag, 2004.

[147] E. Zitzler, L. Thiele, and J. Bader. On set-based multiobjective optimization. *IEEE Transaction on Evolutionary Computation*, 14 :58–79, 2010.

[148] E. Zitzler, L. Thiele, M. Laumanns, C. Fonseca, and V. Grunert da Fonseca. Performance assessment of multiobjective optimizers : An analysis and review. *IEEE Transaction on Evolutionary Computation*, 7 :117–132, 2003.

Annexe A

CV détaillé

Nicolas JOZEFOWIEZ
Né le 12 décembre 1978 (34 ans)
à Sainte-Catherine (62)
Célibataire
E-mail :
nicolas.jozefowiez@laas.fr

Maître de conférences (27e section)
INSA Toulouse, LAAS-CNRS

Fonctions occupées

02/2008-...	**Maître de conférences en Informatique** *INSA/LAAS-CNRS, Toulouse*
07/2007-01/2008	**Stagiaire postdoctoral** *Ecole des Sciences de la Gestion de l'UQAM et Centre Interuniversitaire de Recherche sur les Réseaux d'Entreprise, la Logistique et le Transport, Montréal, QC, Canada*
09/2006-06/2007	**Attaché Temporaire à l'Enseignement et à la Recherche en Informatique** *Université de Valenciennes et du Hainaut-Cambrésis*
09/2005-07/2006	**Chercheur visitor (Fulbright grant)** *Leeds School of Business, University of Colorado at Boulder, USA*
10/2001-09/2005	**Attaché Temporaire à l'Enseignement et à la Recherche (1/2 poste) en Informatique** *Université de Lille 1*
2000-2001	**Vacataire de l'Enseignement Supérieur.** *Université de Lille 1*

Formation

10/2001-12/2004	**Doctorat d'informatique** à l'Université des Sciences et Technologies de Lille. Allocation Ministérielle (MNERT) + Monitorat CIES Laboratoire d'Informatique Fondamentale de Lille (LIFL, 59). Equipe OPAC (Optimisation PArallèle et Collaborative). Sujet : "Modélisation et résolution approchée de problèmes de tournées de véhicules multi-objectif" Thèse soutenue le 06 Novembre 2004, Mention Très Honorable Jury M. M. DAUCHET (Professeur, Université de Lille 1, Président) M. J. FIGUEIRA (Professeur associé, Université de Coimbra, Portugal, Rapporteur) M. C. PRINS (Professeur, Université de Technologies de Troyes, Rapporteur) M. X. GANDIBLEUX (Professeur, Université de Nantes, Examinateur) Directeurs : M. F. SEMET, M. E-G. TALBI
2000-2001	**DEA d'Informatique** , Mention "Bien", *Université de Lille 1*
1999-2001	**Maîtrise d'Informatique** , Mention "Bien", *Université de Lille 1*
1998-1999	**Licence d'Informatique** , Mention "Bien", *Université de Lille 1*

Encadrement

Thèses

- Panwadee Tangpattanakul, *Optimisation multi-objectif de missions de satellites d'observation de la Terre,* bourse Franco-Thaïlandaise (50%, avec P. Lopez) (2009-2013). Soutenance : 26/09/2013.
  - *Publications :* ROADEF 2012, PPSN 2012, ROADEF 2013, WCO 2013
- Boadu Mensah Sarpong, *Column generation for bi-objective integer programs : Application to vehicle routing problems,* allocation ministérielle (50% avec C. Artigues) (2009-2013).
  - *Publications :* ROADEF 2012, ODYSSEUS 2012, NOW 2013, VeRoLog 2013, ATMOS 2013
- Leonardo Malta, *Problèmes de transport intégré porte-à-porte multi-modaux*, ANR RESPET (50% avec F. Semet) (2012-...).
- Leticia Gloria Vargas Suarez, *Problèmes de tournées couvrantes cumulatives multi-objectif pour la logistique humanitaire,* bourse Erasmus Mundus Laminatec (50% avec S. U. Ngueveu) (2013-...)

Postdoctorants (2)

- Rodrigo Acuna-Agost, , financement Amadeus S.A. (co-encadré avec C. Mancel) (2009-2010).
- H. Murat Afsar, *Optimisation de tournées de collecte de déchets intelligentes*, bourse région Midi-Pyrénées (co-encadré avec P. Lopez) (2008-2009).

Master

- Oussama Ben Ammar (2010) : Université de Toulouse 2 *Ordonnancement bi-objectif sur une machine.* (co-encadré avec C. Artigues)
- Myriam Foucras (2010) : Université Paul Sabatier *Problème du voyageur de commerce multi-modal.*
- Simon Verhnes (2010) : INSA de Toulouse (M1) *Méthode de branchement local pour le problème du voyageur de commerce avec labels.*
- Pauline Desseaux (2009) : EISTI *Méthode de recherche arborescente pour un problème d'ordonnancement bi-objectif.* (co-encadré avec C. Artigues)

Subventions de recherche et contrats

- R&T CNES R-S12/BS-004-002 «Planification Mission Flexible», 11/2012 - 09/2013, Participants : CNES, Astrium SAS, ONERA, LAAS-CNRS.
- «Energy-Aware feeding System» - Projet ECO-INNOVERA - 2012-2015.
- «Gestion de Réseaux de Services Porte-à-Porte Efficace pour le Transport de Marchandises (RESPET)», Agence Nationale de la Recherche (ANR), Programme Transports Terrestres Durables, 560 225 euros, 2012-2015, subvention de groupe (collaborateur).
- «Algorithmes de programmation linéaire embarquable pour le rendez-vous orbital» ; Centre National de l'Etude Spatiale, 30 000 euros, 2010, subvention de groupe (collaborateur).

– «Gestion optimisée des perturbations opérationnelles dans les transports aériens», Amadeus SA, 75 000 euros, 10/2009-09/2010, subvention de groupe (collaborateur).

Activités

– Co-animateur du groupe de travail «Programmation Mathématique Multi-Objectif» de la société française de recherche opérationnelle et d'aide à la décision (ROADEF), 2008 - 2013.
– Responsable de la 3e année «Modélisation, Informatique, Communication» du département «Sciences et Technologies pour l'Ingénieur», INSA Toulouse, 2009 - 2012.
– Membre du comité d'organisation de la conférence de la société française de recherche opérationnelle et d'aide à la décision, ROADEF 2010.
– Membre du jury du programme chercheur de la bourse Fulbright pour la France, 2010.
– Membre élu du conseil de laboratoire du Laboratoire d'Analyse et d'Architecture des Systèmes, 2011 - 2013.
– Organisateur de la 7e réunion du groupe de travail Transport et Logistique du groupe de recherche «Recherche opérationnelle» du CNRS, 5/12/2011.
– Evaluateur pour le programme «Knowledge-building projects for industry» (VERDIKT program) du Research Council of Norway, 2011.
– Evaluateur pour un projet CEFIPRA (agence de financement de projets de recherche bilatérale France-Inde), 2012.
– Evaluateur pour un projet de la Research Foundation - Flanders (FWO), 2012.
– Responsable des séminaires de l'équipe MOGISA et du thème Décision et Optimisation (DO) du LAAS-CNRS.
– Membre élu du bureau de la ROADEF (trésorier), 2013 - .
– Membre du comité d'organisation de la conférence ODYSSEUS 2015.
– Arbitrages pour : *4OR, Artificial Intelligence, Computers & Operations Research, Computers and Industrial Engineering, Engineering Applications of Artificial Intelligence, European Journal of Operational Research, IEEE Computational Intelligence Magazine, IEEE Transaction on Evolutionary Computation, IJCAI 2011, INFOR, International Journal Mathematics in Operations Research, Journal of Heuristics, Journal of Mathematical Modelling and Algorithms, Omega, Operations Research, Optimization Letters, SIAM Journal on Computing, Simulation Modelling Theory and Practice.*

Annexe B

Bibliographie personnelle

Chapitres de livre

1. N. Jozefowiez, F. Semet, and E-G. Talbi, "From single-objective to multi-objective vehicle routing problems : Motivations, case studies, and methods", chapter in "The vehicle routing problem : Latest advances and new challenges", B. L. Golden, S. Raghavan, E. Wasil (Eds), p. 445-471, Springer, 2008.
2. N. Jozefowiez, F. Semet, and E-G. Talbi, "A multi-objective evolutionary algorithm for the covering tour problem", chapter 11 in "Applications of multi-objective evolutionary algorithms", C. A. Coello Coello and G. B. Lamont (editors), p 247-267, World Scientific, December 2004.

Articles publiés dans des revues avec comité de lecture

1. N. Jozefowiez, C. Mancel, F. Mora-Camino, "A heuristic approach based on shortest path problems for integrated flight, aircraft and passenger rescheduling under disruptions", Journal of the Operational Research Society, 64, p. 383-395, 2013.
2. N. Jozefowiez, G. Laporte, F. Semet, "A generic branch-and-cut algorithm for multiobjective optimization problems : Application to the multilabel traveling salesman problem", INFORMS Journal on Computing, 24, p. 554-564, 2012.
3. H. M. Afsar, N. Jozefowiez, P. Lopez, "A Branch-and-Price Algorithm for the Windy Rural Postman Problem", RAIRO-RO, 45, p. 320-331, 2011.
4. N. Jozefowiez, G. Laporte, F. Semet, "A branch-and-cut algorithm for the minimum labeling Hamiltonian cycle problem and two variants", Computers & Operations Research, 38, p. 1534 - 1542, 2011.
5. N. Jozefowiez, F. Semet, and E-G. Talbi, "An evolutionary algorithm for the vehicle routing problem with route balancing", European Journal of Operational Research, 195, p. 761-769, 2009.
6. N. Jozefowiez, F. Glover, and M. Laguna, "Multi-objective meta-heuristics for the traveling salesman problem with profits", Journal of Mathematical Modelling and Algorithms, 7, p. 177-195, 2008.

7. N. Jozefowiez, F. Semet, and E-G. Talbi, "Multi-objective vehicle routing problems", European Journal of Operational Research, 189, p. 293-309, 2008.
8. N. Jozefowiez, F. Semet, and E-G. Talbi, "The bi-objective covering tour problem", Computers and Operations Research, 34, p. 1929-1942, 2007.
9. N. Jozefowiez, F. Semet, and E-G. Talbi, "Target Aiming Pareto Search and its application to the vehicle routing problem with route balancing", Journal of Heuristics, 13, p. 455-469, 2007.

Articles publiés dans des actes de congrès avec comité de lecture

1. B. M. Sarpong, C. Artigues, N. Jozefowiez, "Column generation for bi-objective vehicle routing problems with a min-max objective", in Proceedings of the 13th Workshop on Algorithmics Approaches for Transportation Modelling, Optimization, and Systems, OASIcs, Volume 33, p.137-149, 2013.
2. P. Tangpattanakul, N. Jozefowiez, P. Lopez, "Biased Random Key Genetic Algorithm with Hybrid Decoding for Multi-objective Optimization", in Proceedings of the Workshop on Computational Optimization 2013, IEEE, 2013 (accepté).
3. P. Tangpattanakul, N. Jozefowiez, P. Lopez, "Multi-objective optimization for selecting and scheduling observations by agile Earth observing satellites", in Parallel Problem Solving from Nature (PPSN) 2012, Lecture Notes in Computer Science, Vol. 7492, p. 112-121, Springer, 2012.
4. C. Artigues, N. Jozefowiez, M. Ali Aloulou, "An exact method for the bi-objective one-machine problem with maximum lateness and unit family setup cost objectives", Proceedings of ISCO 2010 - International Symposium on Combinatorial Optimization, Electronic Notes in Discrete Mathematics, Vol. 36, p. 1233-1240, Elsevier, Hammamet, Tunisia, 2010.
5. A. Liefooghe, L. Jourdan, N. Jozefowiez, E-G. Talbi, "On the Integration of a TSP Heuristic into an EA for the Bi-objective Ring Star Problem", in Hybrid Metaheuristics, Lecture Notes in Computer Science, Volume 5296, p. 117-130, Springer, 2008.
6. N. Jozefowiez, F. Semet, and E-G. Talbi, "Enhancements of NSGA II and its application to the vehicle routing problem with route balancing", in 7th International Conference on Artificial Evolution - EA 2005, Lecture Notes in Computer Science, Volume 3871, p. 131-142, Springer, 2006.
7. N. Jozefowiez, F. Semet, and E-G. Talbi, "Parallel and hybrid models for multi-objective optimization : Application to the vehicle routing problem", in Parallel Problem Solving from Nature - PPSN VII, Lecture Notes in Computer Science Vol. 2439, p. 271-280, Springer-Verlag, 2002.

Articles dans des congrès avec comité de sélection

1. N. Jozefowiez, "Column generation for the multiple vehicle covering tour problem", ODYSSEUS 2012, Mikonos, Greece, 2012.
2. B. M. Sarpong, C. Artigues, N. Jozefowiez, "The bi-objective multi-vehicle covering tour problem : Formulation and lower bound", ODYSSEUS 2012, Mikonos, Greece, 2012.

3. R. Acuna-Agost, M. Boudia, N. Jozefowiez, C. Mancel, F. Mora-Camino, "Passenger improver - A second phase method for integrated aircraft-passenger recovery systems", TRISTAN VII, Tromso, Norway, 2010.
4. N. Jozefowiez, G. Laporte, F. Semet, "The multi-modal traveling salesman problem", TRISTAN VII, Tromso, Norway, 2010.
5. N. Jozefowiez, G. Laporte, F. Semet, "A multi-objective branch-and-cut algorithm for the multi-modal traveling salesman problem", ODYSSEUS 2009, Cesme, Turkey, 2009.
6. N. Jozefowiez, F. Glover, and M. Laguna, "Multi-objective meta-heuristics for the traveling salesman problem with profits", META 2006, Hammamet, Tunisia, November 2006.
7. N. Jozefowiez, M. Laguna, and F. Glover, "A multi-objective evolutionary algorithm for the bi-objective traveling salesman problem with profits", Multi-Objective Programming and Goal Programming 2006 (MOPGP'06), Tours, France, June 2006.

Séminaires et conférences sur invitation

1. N. Jozefowiez, C. Artigues, B. M. Sarpong, "Column generation for bi-objective problems with a min-max objective", NOW 2013, Syracuse, Italy, 2013.
2. N. Jozefowiez, "Solving multiobjective problems : Principles and new methods", Département d'Informatique et de Recherche Opérationnelle, Université de Montréal, Canada, Janvier 2012.
3. N. Jozefowiez, "Optimisation multiobjectif et problèmes de tournées de véhicules", Ecole des Sciences de la Gestion, Université du Québec à Montréal, Canada, Janvier 2012.
4. N. Jozefowiez, F. Semet, E-G. Talbi, "Multi-objective vehicle routing problems", IN3-Barcelona, Universitat Oberta de Catalunya, Barcelone, Espagne, Novembre 2011.
5. N. Jozefowiez, G. Laporte, F. Semet, "A multi-objective branch-and-cut algorithm and its application to the multi-label traveling salesman problem", Laboratoire d'Informatique de Paris Nord, Université Paris XIII, France, Février 2011
6. N. Jozefowiez, G. Laporte, F. Semet, "Traveling salesman and multi-modality", NOW 2010, Ajaccio , France, 2010.
7. N. Jozefowiez, F. Semet, E-G. Talbi, "A study of the bi-objective covering tour problem", LAAS-CNRS, Toulouse, France, Juin 2007
8. N. Jozefowiez, F. Glover, M. Laguna, "Multi-objective meta-heuristics for the traveling salesman problem with profits ", Laboratoire d'Informatique Fondamentale de Lille, Université des Sciences et Technologies de Lille, Villeneuve d'Ascq, France, Novembre 2006.
9. N. Jozefowiez, F. Glover, M. Laguna, "Une méta-heuristique hybride pour le problème du voyageur de commerce avec profits", LAMIH, Université de Valenciennes et du Hainaut-Cambrésis, Valenciennes, France, 2006
10. N. Jozefowiez, F. Glover, M. Laguna, "Une méta-heuristique hybride pour le problème du voyageur de commerce", Ecole des Mines de Nantes, Nantes, France, 2006.

Annexe C

Articles

Les trois premières publications portent sur des travaux présentés dans ce document et étendent leur présentation. Les articles suivants correspondent à des travaux réalisés après ma thèse mais non présentés dans ce mémoire.

- B. M. Sarpong, C. Artigues, N. Jozefowiez, "Column generation for bi-objective vehicle routing problems with a min-max objective", in Proceedings of the 13th Workshop on Algorithmics Approaches for Transportation Modelling, Optimization, and Systems, OASIcs, Volume 33, p. 137-149, 2013.
- N. Jozefowiez, G. Laporte, F. Semet, "A generic branch-and-cut algorithm for multiobjective optimization problems : Application to the multilabel traveling salesman problem", INFORMS Journal on Computing, 24, p. 554-564, 2012.
- N. Jozefowiez, G. Laporte, F. Semet, "A branch-and-cut algorithm for the minimum labeling Hamiltonian cycle problem and two variants", Computers & Operations Research, 38, p. 1534 - 1542, 2011.
- N. Jozefowiez, C. Mancel, F. Mora-Camino, "A heuristic approach based on shortest path problems for integrated flight, aircraft and passenger rescheduling under disruptions", Journal of the Operational Research Society, 64, p. 383-395, 2013.
- H. M. Afsar, N. Jozefowiez, P. Lopez, "A Branch-and-Price Algorithm for the Windy Rural Postman Problem", RAIRO-RO, 45, p. 320-331, 2011.
- N. Jozefowiez, F. Glover, and M. Laguna, "Multi-objective meta-heuristics for the traveling salesman problem with profits", Journal of Mathematical Modelling and Algorithms, 7, p. 177-195, 2008.
- A. Liefooghe, L. Jourdan, N. Jozefowiez, E-G. Talbi, "On the Integration of a TSP Heuristic into an EA for the Bi-objective Ring Star Problem", in Hybrid Metaheuristics, Lecture Notes in Computer Science, Volume 5296, p. 117-130, Springer, 2008.

Column Generation for Bi-Objective Vehicle Routing Problems with a Min-Max Objective

Boadu Mensah Sarpong[1,2], **Christian Artigues**[2,3], **and Nicolas Jozefowiez**[1,2]

1 **CNRS, LAAS, 7 avenue du colonel Roche, F–31400 Toulouse, France**
{bmsarpon,artigues,njozefow}@laas.fr

2 **Université de Toulouse, INSA, LAAS, F–31400 Toulouse, France**

3 **Université de Toulouse, LAAS, F–31400 Toulouse, France**

Abstract

Column generation has been very useful in solving single objective vehicle routing problems (VRPs). Its role in a branch-and-price algorithm is to compute a lower bound which is then used in a branch-and-bound framework to guide the search for integer solutions. In spite of the success of the method, only a few papers treat its application to multi-objective problems and this paper seeks to contribute in this respect. We study how good lower bounds for bi-objective VRPs in which one objective is a min-max function can be computed by column generation. A way to model these problems as well as a strategy to effectively search for columns are presented. We apply the ideas to two VRPs and our results show that strong lower bounds for this class of problems can be obtained in "reasonable" times if columns are intelligently managed. Moreover, the quality of the bounds obtained from the proposed model are significantly better than those obtained from the corresponding "standard" approach.

1998 ACM Subject Classification G.1.6 Integer Programming, G.2.3 Applications

Keywords and phrases multi-objective optimization, column generation, integer programming, vehicle routing

Digital Object Identifier 10.4230/OASIcs.ATMOS.2013.137

1 Introduction

Bounds (lower and upper) have been the backbone of methods for solving difficult single objective problems including VRPs. For this reason, it is natural to expect that bounds will also be useful for multi-objective problems. It is, thus, necessary to develop models and strategies for computing good bounds for multi-objective problems. In this paper, we study the use of column generation in computing strong lower and upper bounds for bi-objective VRPs in which one objective is a min-max function. We will use the acronym BOVRPMMO to refer to a problem of this kind.

A BOVRPMMO can be defined by means of a Dantzig-Wolfe decomposition as the selection of a set of columns with minimum total cost such that the maximum value of an attribute associated with the set is minimized. More formally, we consider problems of the

13th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS'13).
Editors: Daniele Frigioni, Sebastian Stiller; pp. 137–149

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

form:

$$\text{Minimize} \quad \sum_{k\in\Omega} c_k\lambda_k \tag{1}$$

$$\text{Minimize} \quad \Gamma_{\max} \tag{2}$$

$$\text{subject to} \quad \sum_{k\in\Omega} a_{ik}\lambda_k \geq b_i \qquad (i \in I)\,, \tag{3}$$

$$\Gamma_{\max} \geq \sigma_k\lambda_k \qquad (k \in \Omega)\,, \tag{4}$$

$$\lambda_k \in \{0,1\} \qquad (k \in \Omega)\,, \tag{5}$$

where λ_k and $\Gamma_{\max}$ are decision variables, Ω is the set of all feasible columns whose description depends on the particular problem, and I is an index set. For each column $k \in \Omega$, c_k and σ_k are two associated values which we suppose to be integers. We need to select columns with minimum sum of c_k such that $\Gamma_{\max} = \max_{k\in\Omega}\{\sigma_k\lambda_k\}$ is also minimized. Bi-objective generalizations of several vehicle routing problems satisfying this condition can be defined. In general, we want to minimize the combined cost of a set of routes such that the value of a property associated with the selected routes (eg. the maximum length of a route, max capacity of a route, etc.) is minimized. We will later present two of such problems namely the bi-objective uncapacitated vehicle routing problem (BOUVRP) and the bi-objective multi-vehicle covering tour problem (BOMCTP).

A BOVRPMMO is a special case of a multi-objective combinatorial optimization (MOCO) problem. A general MOCO problem concerns the minimization of a vector of two or more functions $F(x) = (f_1(x), \ldots, f_r(x))$ over a finite domain of feasible solutions $\mathcal{X}$. The vector $x = (x_1, \ldots, x_n)$ is the decision variable or solution, $\mathcal{Y} = F(\mathcal{X})$ corresponds to the images of the feasible solutions in the objective space, and $y = (y_1, \ldots, y_r)$, where $y_i = f_i(x)$, is a point of the objective space. A solution x' *dominates* another solution x'' if for any index $i \in \{1, \ldots, n\}$, $f_i(x') \leq f_i(x'')$ and there is at least one index $i \in \{1, \ldots, n\}$, such that $f_i(x') < f_i(x'')$. A feasible solution dominated by no other feasible solution is said to be *efficient* or *Pareto optimal* and its image in the objective space is said to be *nondominated*. The set of all efficient solutions is called the *efficient set* (denoted $\mathcal{X}_E$) and the set of all nondominated points is the *nondominated set* (denoted $\mathcal{Y}_N$). Although the meaning of bounds in single objective optimization is well studied and understood, the situation is quite different in the multi-objective case. Ideal and nadir points are well known lower and upper bounds, respectively, of the set $\mathcal{Y}_N$. The coordinates of the ideal point are obtained by optimizing each objective function independently of the others, whereas the coordinates of the nadir point correspond to the worse value of each objective function when we consider the set $\mathcal{X}_E$. From Figure 1, it can be seen that these points are usually poor bounds since they just estimate the whole region where a member of $\mathcal{Y}_N$ may lie. In this paper, we will be interested in bounds that can reduce the region where the members of $\mathcal{Y}_N$ are and thus narrow down the search for nondominated points.

Given that a MOCO problem is a discrete problem, its lower bound can be defined as a finite set of points such that the image of every feasible solution is dominated by at least one of these points [15]. The members of a lower bound set do not necessarily belong to $\mathcal{Y}$. An upper bound may also be defined as a finite set of points in $\mathcal{Y}$ that do not dominate one another. This idea of *bound sets* for bi-objective combinatorial optimization (BOCO) problems has recently been revisited by other authors [3, 6, 14]. They compute strong lower bounds based on a weighted sum scalarization which can then be used in developing exact algorithms for the problems they consider. Although each objective function f_i of the vector F can either be a *sum objective* as in (1) or a *min-max objective*, examples given in the

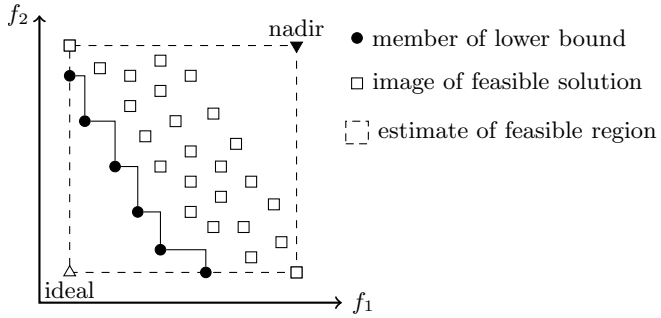


Figure 1 Lower and upper bounds of a bi-objective combinatorial optimization problem.

cited papers consider only the "sum" type of objectives. This in a way justifies the use of a weighted sum method since they can efficiently find supported efficient solutions (those that correspond to points on the convex part of $\mathcal{Y}_N$) by using well known single objective optimization methods. Very good lower bounds for many problems can be defined from the set of supported efficient solutions. The situation is quite different when we consider a combination of a sum and a min-max objective function as in the case of a BOVRPMMO. Indeed, a general linearizing method for the min-max objective destroys the problem structure and so the desirable quality of being able to use known methods for the resulting problem does not necessarily apply [5]. A similar thing happens when we use a standard ε-constraint approach although this latter approach can find non-supported solutions which cannot be found by the weighted sum method. Moreover, as shown by results in [6], the quality of the bounds obtained by the weighted sum method for set covering problems are not very good. Nevertheless, the quality of lower bounds produced from set covering based formulations for single objective VRPs are one the best and so we can expect good quality lower bounds from formulations of this type in the multi-objective case too. these reasons, the approach proposed in this paper uses a variant of the ε-constraint method applied to a set covering based formulation for the BOVRPMMO.

The main contribution of this work is to define the application of column generation to a BOVRPMMO that does not rely on any "standard" multi-objective technique. In this way, we avoid some drawbacks such as the impossibility to find non-supported solutions by a weighted sum method or the possible loosening of a lower bound by explicitly adding constraints on objectives as it in the case of a standard ε-constraint approach. Moreover, if the problem linked to the first objective is a well-studied problem for which an efficient column generation algorithm exists then it is possible to reuse the pricing scheme to solve the bi-objective problems obtained by adding a second objective linked to a property of a selected set of columns. Another contribution is the computation of bound sets by using a scalar method other than the weighted sum method which has been used in published papers (to the best of our knowledge).

The use of column generation to compute bounds for a BOVRPMMO is explained in Section 2. Applications problems are discussed in Section 3. Computational results and conclusions are provided in Sections 4 and 5, respectively.

2 Column Generation for a BOVRPMMO

A close examination of formulation (1 – 5) reveals that a BOVRPMMO decomposes naturally into two problems. For any set of feasible columns, the associated value of $\Gamma_{\max}$ can easily

be computed. We can therefore use a variant of the ε-constraint method with one main difference. Instead of explicitly adding a constraint of the form $\Gamma_{\max} \leq \varepsilon$ to the formulation, we rather drop (4) and use it to redefine the feasibility of a column. Thus, we define a new set of feasible columns $\bar{\Omega}$ where the feasibility of a column $k \in \bar{\Omega}$ now depends on its associated value σ_k. Depending whether or not a column $k \in \Omega$ may be associated with more than one value of σ_k, we may have a larger set of feasible columns after the redefinition. The strength of the model is conserved at the expense of having a possibly more difficult problem due to the possible increase in the number feasible columns. The master problem (MP) becomes the following single-objective program:

$$\text{Minimize} \sum_{k \in \bar{\Omega}} c_k \lambda_k \tag{6}$$

$$\text{subject to} \quad \sum_{k \in \bar{\Omega}} a_{ik} \lambda_k \geq b_i \qquad (i \in I)\,, \tag{7}$$

$$\lambda_k \in \{0,1\} \quad (k \in \bar{\Omega})\,. \tag{8}$$

Before solving MP for a given limit ε on the value of $\Gamma_{\max}$, we need to set $\lambda_k = 0$ for all columns $k \in \bar{\Omega}_k$ having $\sigma_k > \varepsilon$. The linear relaxation of MP (ie. $\lambda_k \geq 0\ \forall k \in \bar{\Omega}$) is denoted as LMP.

2.1 Computing Lower and Upper Bounds

For a BOVRPMMO, $\Gamma_{\max}$ can only take on a finite number of values. If the complete set of feasible columns $\bar{\Omega}$ is known, a lower bound can be computed by using a variant of the ε-constraint approach as given in Algorithm 1. The algorithm starts with no restriction on the value of σ_k for a column. At each iteration, a linear relaxation of the problem is solved after which the optimal value as well as the value of $\Gamma_{\max}$ are determined. In the next iteration, the problem is updated to exclude columns k for which σ_k is greater than $\Gamma_{\max}$. This iterative process continues for as long as the problem remains feasible. In practice, the cardinality of $\bar{\Omega}$ is too large and so a column generation method needs to be used by considering only a subset $\bar{\Omega}_1$ of $\bar{\Omega}$. The restriction of MP to $\bar{\Omega}_1$ is called the restricted master problem (RMP) and the resulting linear relaxation (ie. $\lambda_k \geq 0\ \forall k \in \bar{\Omega}_1$) is denoted LRMP. Let π_i $(i \in I)$ be the dual variables associated with LRMP. The subproblem is defined as

$$S(\varepsilon) = \min_{k \in \bar{\Omega} \setminus \bar{\Omega}_1} \left\{ c_k - \sum_{i \in I} \pi_i a_{ik} : \sigma_k \leq \varepsilon \right\}, \tag{9}$$

where ε is a maximum allowed value of $\Gamma_{\max}$ in LRMP. In applying column generation to compute a lower bound, we need to be able to efficiently search for relevant columns and so we explore some strategies in the next subsection.

Algorithm 1 Computing a lower bound

1: Set $lb \leftarrow \emptyset$.
2: **while** LMP is feasible **do**
3: Solve LMP. Let c^* be the optimum, and λ^* be the optimal solution vector.
4: Compute $\Gamma_{\max} = \max_{k \in \bar{\Omega}} \sigma_k \lambda_k^*$.
5: $lb \leftarrow lb \cup \{(c^*, \Gamma_{\max})\}$.
6: Set $\lambda_k \leftarrow 0$ for all k such that $\sigma_k \geq \Gamma_{\max}$.
7: **end while**

After computing a lower bound, a simple way to compute an upper bound is to solve RMP (the integer program) several times by following the idea of Algorithm 1. That is, we consider the RMP with the columns it contains after computing a lower bound and follow Algorithm 1 by replacing LMP with RMP. This is perhaps the simplest column generation heuristic. Although an upper bound is made up of feasible points, they do not necessary belong to $\mathcal{Y}_N$ since the RMP may not contain all relevant columns. Nevertheless, if the columns in RMP are relevant for the integer program then we expect that the upper bound produced will be a good approximation of $\mathcal{Y}_N$.

2.2 Column Search Strategies

A first approach of applying column generation to compute a lower bound of a BOIPMMO follows the idea of a standard ε-constraint method. For a fixed value of ε, LRMP is solved to optimality by column generation before moving on to another value of ε. An iteration of column generation consists in solving LRMP once to obtain a vector of dual values, and then solving the corresponding subproblem to obtain new columns to add to LRMP. The method converges when no new columns are produced from the subproblem. We denote this approach as the "Point-by-Point Search (PPS)". Although PPS is simple and easy to implement, it takes no advantage of the similarities in the subproblems for the different values of ε and so a tremendous amount of computational time may be spent in computing each member of a lower bound.

Using heuristics to generate columns can improve the performance of column generation [4]. These heuristics are used to cheaply generate other relevant columns from those found by a subproblem algorithm. In the bi-objective case, we are interested in heuristics that can take advantage of similarities in the different subproblems solved when computing each point of a lower bound. That is, once the cost of finding a first column has been paid we wish to quickly generate other relevant columns that are relevant for the current subproblem and may also be relevant for other subproblems. A column which has negative reduced cost for a current subproblem, does not necessary have a negative reduced cost for another subproblem since the associated dual variables do not necessarily have the same values. Nevertheless, it can be expected that two subproblems that are close in terms of objectives, may also be close in terms of the solution of LRMP and therefore close in terms of dual variable values. For this reason, a column generated by a heuristic may also be of negative reduced cost for several other subproblems apart from the current one. In addition, standard algorithms used to solve a subproblem are most times only interested in finding the best columns. This means that many columns having negative reduced costs are left out because the algorithm finds "better" columns. This may be desirable in the single objective case. In the bi-objective case, however, a column which may not be so good for a subproblem may be the best for another subproblem so we are interested in heuristics that can efficiently search for these columns by modifying the ones found by a subproblem algorithm. We denote an approach which incorporate such heuristics as "Improved Point-by-Point Search (IPPS)". IPPS can be useful as a column generation based heuristic since at each iteration it tries to generate a set of columns that are relevant for several subproblems. IPPS is summarized in Algorithm 2 and the heuristic used in Step 7 obviously depends on the problem at hand.

3 Application Problems

In this section, we apply the ideas presented so far to compute lower and upper bounds for two BOVRPMMO. Just as for most VRPs, the subproblem encountered in both examples is

Algorithm 2 Improved Point-by-Point Search (IPPS)

Set $\varepsilon \leftarrow \infty$, and $lb \leftarrow \emptyset$.
while LRMP is feasible **do**
  Solve LRMP once to obtain a vector of dual values.
  Let c^* be the optimum, λ^* the optimal vector, and compute $\Gamma_{\max} = \max_{k \in \bar{\Omega}} \sigma_k \lambda_k^*$.
  Solve the subproblem $S(\varepsilon)$ and let Λ be the set of columns obtained.
  if $|\Lambda| \neq 0$ **then**
    For each column in Λ use heuristics to generate other relevant columns from it.
  else
    $lb \leftarrow lb \cup \{(c^*, \Gamma_{\max})\}$.
    Set $\lambda_k \leftarrow 0$ for all k such that $\sigma_k \geq \sigma^*$, and $\varepsilon \leftarrow \Gamma_{\max} - 1$.
  end if
end while

an elementary shortest part problem with resource constraints (ESPPRC) which we solve by the decremental state space relaxation algorithm (DSSR) [1, 13]. For each considered problem, we discuss the specific implementation of DSSR as well as the heuristic used in implementing IPPS. A complete description of DSSR can be obtained from the two references. We will also be using the same notations introduced earlier and so only their specific meanings for each example will be mentioned.

3.1 The Bi-Objective Uncapacitated Vehicle Routing Problem

The bi-objective uncapacitated VRP (BOUVRP) is defined on an undirected graph $G = (V, E)$ where $V = \{v_0, \ldots, v_n\}$ is a set of nodes and $E = \{(v_i, v_j) : v_i, v_j \in V, i \neq j\}$ is a set of edges. Node v_0 is the depot where all routes should start and end. The other nodes represent n customers with each having a fixed demand. A distance matrix $D = (d_{ij})$ which satisfies the triangle inequality is defined on E. The problem is to design a set of routes with the objectives of minimizing both the total length of all routes and the maximum total demand of customers served by any single route. The demand of each customer is to be met by at least one visiting route and the number of available vehicles as well as the capacity of a vehicle are unlimited.

Master Problem and Subproblem

Following the notations used in Section 2, we let Ω be the set of all feasible columns. A column $k \in \Omega$ is a Hamiltonian cycle on a subset of V which includes the depot. For each column k, c_k is its length and σ_k is the sum of the demands of the customers visited by the route it represents. By redefining the feasibility of a column to take into account the total demand of the customers visited by the route it represents, we obtain a new set of feasible columns $\bar{\Omega}$. The master problem (MP) then becomes a capacitated VRP given by:

$$\text{Minimize} \quad \sum_{k \in \bar{\Omega}} c_k \lambda_k \tag{10}$$

$$\text{subject to} \quad \sum_{k \in \bar{\Omega}} a_{ik} \lambda_k \geq 1 \qquad (v_i \in V \backslash \{v_0\}) , \tag{11}$$

$$\lambda_k \in \{0, 1\} \qquad (k \in \bar{\Omega}) , \tag{12}$$

where $a_{ik} = 1$ if the route of column k visits customer v_i. As explained in the previous section, the second objective to minimize the total demand of customers served by a single route ($\Gamma_{\max}$) does not appear in the master problem. The subproblem is given by:

$$S(\varepsilon) = \min_{k \in \bar{\Omega} \setminus \bar{\Omega}_1} \left\{ c_k - \sum_{v_i \in V \setminus \{v_0\}} \pi_i a_{ik} : \sigma_k \leq \varepsilon \right\}, \tag{13}$$

where π_i are dual values and ε is a limit imposed on the sum of the demands of customers visited by the same route.

Subproblem Algorithm

In (13), we are supposed to find feasible routes with negative reduced costs such that the sum of demands of the subset of customers it visits is at most ε. The reduced cost of a route is given by its length minus the sum of the profits (dual values) associated to the nodes it visits. The only considered resource when implementing DSSR is the sum of the demands a route visits. This sum cannot be more than ε. For two labels l_1 and l_2 arriving at the same node, l_1 dominates l_2 if l_1 has a smaller reduced cost and smaller sum of demands than l_2. In this situation, l_2 is rejected.

IPPS Heuristic

Although DSSR is able to return several columns with negative reduced cost, some columns are not generated since they are dominated by other generated ones. Nevertheless, some of these rejected columns have negative reduced costs and may be relevant for a subproblem corresponding to a different value of ε. The purpose of the heuristic is to find these kind of routes by using the following idea. Given a column with negative reduced cost, we try to remove a visited node from (or insert a non-visited node in) the route in such a way that we obtain a new route which is still of negative reduce cost but has different values of c_k and σ_k. Due to the change in the value of σ_k, a new route found in this way can be valid for a different subproblem for which the original route was not valid. This new route may also be of negative reduced cost for a different subproblem.

3.2 The Bi-Objective Multi-Vehicle Covering Tour Problem

The bi-objective multi-vehicle covering tour problem (BOMCTP) is an extension of the covering tour problem (CTP) [7]. The CTP consists in designing a route over a subset of locations with the aim of minimizing the length of the route. In addition, each location not visited by the route should lie within a fixed radius of a visited location. The fixed radius is called the *cover distance*. A generic application of the CTP is given in the design of bi-level transportation networks where the aim is to construct a primary route such that all points that are not on it can easily reach it [2]. Other applications are the post box location problem [11] and in the delivery of medical services to villages in developing countries [2, 9]. A bi-objective generalization [10] as well as a multi-vehicle extension [8] of the CTP have been proposed. In the bi-objective version, the cover distance is not fixed in advance but rather induced by the constructed route. It is computed by assigning each non-visited location to the closest visited location and calculating the maximum of these distances. The objectives are to minimize the length of the route as well as the *induced cover distance*. In the multi-vehicle version, the combined length of a set of routes is minimized for a fixed cover distance. The number of locations that a single route can visit is limited by a predetermined constant p.

The BOMCTP discussed here, is a combination of the bi-objective and the multi-vehicle extensions of the CTP and it is defined on an undirected graph $G = (V \cup W, E)$. Set V represents locations which can be visited by a route whereas the members of W are to be assigned to visited locations of V. Node $v_0 \in V$ is the depot where all routes must start and also end. Set E consist of edges connecting all pairs of nodes in $V \cup W$ and a distance matrix $D = (d_{ij})$ satisfying the triangle inequality is defined on this set. The problem consists in minimizing both the total length of a set of routes constructed over a subset of V and the induced cover distance.

Master Problem and Subproblem

Let Ω represent the set of all feasible columns. A feasible column $k \in \Omega$ is defined as a route R_k which is a Hamiltonian cycle on a subset of V, includes the depot and visits not more than p nodes. The length of R_k is denoted c_k. For each route R_k, we choose a subset $\Psi_k \subseteq W$ of nodes it may cover and define σ_k as the maximum distance between a node of Ψ_k and the closest node of R_k. The constant $a_{ik} = 1$ if $w_i \in \Psi_k$ and $a_{ik} = 0$ otherwise. Let $\Gamma_{\max}$ represent the cover distance induced by a set of routes. Just as before, we define a new set of feasible columns $\bar{\Omega}$ where the feasibility of a column $k \in \bar{\Omega}$ depends not just on R_k but also on σ_k. The master problem (MP) is given by:

$$\text{Minimize} \quad \sum_{k \in \bar{\Omega}} c_k \lambda_k \tag{14}$$

$$\text{subject to} \quad \sum_{k \in \bar{\Omega}} a_{ik} \lambda_k \geq 1 \qquad (w_i \in W)\,, \tag{15}$$

$$\lambda_k \in \{0, 1\} \quad (k \in \bar{\Omega})\,. \tag{16}$$

The function (14) minimizes the total length of the set of routes and (15) ensures that each node of W is covered by a selected route. The second objective to minimize $\Gamma_{\max}$ does not appear in the above formulation for the same reasons as before. If π_i are the dual values associated with (15) then the subproblem is

$$S(\varepsilon) = \min_{k \in \bar{\Omega} \setminus \bar{\Omega}_1} \left\{ c_k - \sum_{w_i \in W} \pi_i a_{ik} : \sigma_k \leq \varepsilon \right\}. \tag{17}$$

Subproblem Algorithm

Given that $R_k \subseteq V$ whereas $\Psi_k \subseteq W$, we need to construct a route on a subset of V with the aim of minimizing its length c_k and also choose a subset of W with the aim of maximizing the profits (π_i) associated to its members. The profit associated to a node of $w_i \in W$ can be collected at most once on any single route even though different nodes of the route may be able to cover w_i. Two resources are considered in implementing DSSR for this problem. The first is concerned with the number of nodes a route visits which is limited to a maximum of p. The second resource constraint is that a route may only cover nodes of W that lie within a radius of ε from a node of V it visits. During the extension of a label, nodes of W not yet covered by the label but which can be covered are identified and the resulting profit is subtracted from the current reduced cost of the label. Doing so ensures that we obtain the minimum possible reduced cost for each label and this is the goal of the subproblem. Checking whether a label l_1 dominates another label l_2 follows the usual rules when comparing the consumption of resources. When comparing the reduced costs, however, a factor F_{12} which represents the sum of the profits associated to nodes of W covered by l_1

but not yet covered by l_2 should be subtracted from the reduced cost of l_2. This is to ensure that no label that can lead to an optimal solution is eliminated. Similar dominance rules used in dynamic programming algorithms for solving shortest path problems like the one encountered here (called non-additive shortest path problems) are discussed in [12].

IPPS Heuristic

The subproblem constructs a column k by taking Ψ_k to be all the nodes of W that can be covered by the constructed route R_k. This helps with the goal of minimizing the reduced cost. We note, however, that Ψ_k does not necessarily need to include all the nodes of W that can be covered by R_k. Indeed, Ψ_k can be chosen to be any subset of W such that the sum of the associated profits exceeds the cost c_k. A column defined in this way is never found by DSSR for the current subproblem since it is dominated by another column defined by the same route, but covers some more nodes of W. The IPPS heuristic employed here relies on this observation. For each column k given by (R_k, Ψ_k) that is found by DSSR, we successively remove the node of Ψ_k that induces the value of σ_k (i.e. which is farthest from the closest node of R_k) in order to create another column k' with $c'_k = c_k$ but $\sigma'_k < \sigma_k$. We recall that $\sigma_k = \max\{d_{ij} : v_i \in R_k \text{ and } w_j \in \Psi_k\}$. A column found in this way can be valid (and possibly have a negative reduced cost) for another subproblem for which the original column returned by DSSR is not valid. This is due to the different value of ε each subproblem is based on.

4 Computational Results

Evaluating the Quality of Lower and Upper Bounds

In order to evaluate the quality of the computed lower and upper bound sets, we used a distance based measure (μ_1), and an area based measure (μ_2) which were presented in [6]. Combining an area based measure with a distance based measure gives a better indication of the quality of the bounds. Roughly speaking, μ_1 represents the worst distance (with respect to the range of objective values) between a point of the upper bound and a point of lower bound closest to it. Also, μ_2 represents the fraction of the area that is dominated by the lower bound but not by the upper bound. This is, the area where additional points of $\mathcal{Y}_N$ can be found. If a lower bound and a corresponding upper bound are good then we expect that both μ_1 and μ_2 will be small in value. The smaller both values are, the better the quality of the bounds. These two measures complement each other so the quality of the bounds cannot be said to be very good if just one of the measures is small in value but the other is very big. As explained by the author, these measures can be seen to play a role analogous to the optimality gap in single objective optimization. The reader is referred to the relevant paper [6] for further explanation of the measures.

Experiments

Experiments were conducted to evaluate the quality of lower and upper bounds obtained from the model presented (by redefining the feasibility of a column) with respect to a standard ε-constraint model (by explicitly adding constraints on objectives in the master problem). We recall that in the standard ε-constraint model, constraints that limit the value of σ_k for a column are directly added to the master problem and nothing special is done in the subproblem. On the other hand, the new model does not explicitly add constraints to the master problem but rather limit the value of σ_k for a column by redefining the meaning

Table 1 Comparison of the quality of bound sets for the BOUVRP.

	Standard					PPS			IPPS		
Instance	$\|lb\|$	$\|ub\|$	$\mu_1\%$	$\mu_2\%$	$\|lb^*\|$	$\|ub\|$	$\mu_1\%$	$\mu_2\%$	$\|ub\|$	$\mu_1\%$	$\mu_2\%$
eil7	2	6	2.60	37.32	6	6	0.00	3.26	6	0.00	3.26
eil13	2	32	2.92	25.22	99	33	0.13	2.72	34	0.10	2.67
eil22	11	36	1.17	11.68	100	41	0.15	2.24	40	0.11	2.11
eil23	2	11	1.48	50.14	130	14	0.30	16.24	17	0.13	15.90
eil30	5	18	9.12	24.30	152	19	1.08	5.32	22	0.98	4.89
eil31	13	44	5.30	10.95	173	41	0.88	2.69	42	0.40	2.65

of a feasible column both in the master problem and subproblem. The principle used to determine the values of ε is the same for both models. Given a value for $\Gamma_{\max}$ in an iteration, we define $\varepsilon = \Gamma_{\max} - 1$ in the next iteration. We also wanted to compare the quality of the bounds and the computational times of PPS and IPPS. Capacitated VRP instances from the TSPLIB (`http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/`) having up to 31 nodes were used for the BOUVRP. For the BOMCTP, the Mersenne Twister random number generator (`http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html`) was used to generate instances similar to those described in the literature [7, 8, 10] but which are not publicly available. The node sets were obtained by generating $|V| + |W|$ points in the $[0, 100] \times [0, 100]$ square with the depot restricted to lie in $[25, 75] \times [25, 75]$. Set V is taken to be the first $|V|$ points and set W is taken as the remaining points. The distance between two points is calculated as the Euclidean distance. Five instances for every combination of $|V| \in \{40, 50\}$ and $|W| \in \{2|V|, 3|V|\}$ were generated and values of $p \in \{5, 8\}$ were tested. The exact instances used for our experiments can be found at `http://homepages.laas.fr/bmsarpon/ctp_instances.zip`. All computer codes were written in C/C++ and the LRMP was solved with ILOG CPLEX 12.4. Tests were run on an Intel(R) Core(TM)2 Duo CPU E7500 @ 2.93GHz computer with a 2 GiB RAM. Summary of results for the BOUVRP are given in Tables 1 and 2 whereas those for the BOMCTP are given in Tables 3 and 4. The values for the BOMCTP are averages over the five instances as explained. The column headings of the tables have the following meaning: $|lb|$ and $|ub|$ are the cardinalities of a lower bound and upper bound set, respectively; $\mu_1\%$ and $\mu_2\%$ are the values of the quality measures multiplied by 100; *time* is the computational time in cpu seconds; *dssr* is the number of times the sub-problem was solved with DSSR; *cols* is the total number of columns generated. Since PPS and IPPS are based on the same model, the same lower bounds were obtained and this conforms to the theory of column generation which is an exact method for the LMP. The cardinality of the common lower bound is given in the column $|lb^*|$.

From the results obtained, we see that the bounds obtained by the model that redefines the feasibility of a column are significantly better than those obtained by a standard ε-constraint approach. This is seen by comparing the values of μ_1 and μ_2 for PPS and IPPS with their counterparts from "Standard". For example, in Table 1 a standard ε-constraint approach obtained the values $(\mu_1\% = 9.12, \mu_2\% = 24.30)$ for the instance eil30 whereas those for PPS and IPPS were $(\mu_1\% = 1.08, \mu_2\% = 5.32)$ and $(\mu_1\% = 0.98, \mu_2\% = 4.89)$, respectively. A similar thing can be seen in Table 3 for $p = 8$, $|V| = 40$, $|W| = 80$. The values for a standard ε-constraint approach for this instance were $(\mu_1\% = 8.38, \mu_2\% = 25.86)$ which are very huge when compared to $(\mu_1\% = 0.32, \mu_2\% = 3.04)$ for PPS and $(\mu_1\% = 0.38, \mu_2\% = 2.46)$ for IPPS.

Table 2 Comparison of computational times for the BOUVRP.

	Standard			PPS			IPPS		
Instance	time	dssr	cols	time	dssr	cols	time	dssr	cols
eil7	0.1	15	24	0.1	24	29	0.1	13	51
eil13	1.4	60	335	12.3	355	916	8.8	270	1982
eil22	246.3	298	2295	428.7	648	3480	289.6	482	6026
eil23	4216.7	421	3386	7578.5	1238	7946	5979.1	916	12951
eil30	5861.2	541	2976	9746.2	1110	7312	7114.7	828	14922
eil31	2851.0	368	2719	5372.4	957	4514	5027.3	804	7389

Table 3 Comparison of the quality of bound sets for the BOMCTP.

			Standard					PPS			IPPS		
p	$\|V\|$	$\|W\|$	$\|lb\|$	$\|ub\|$	$\mu_1\%$	$\mu_2\%$	$\|lb^*\|$	$\|ub\|$	$\mu_1\%$	$\mu_2\%$	$\|ub\|$	$\mu_1\%$	$\mu_2\%$
5	40	80	7	23	6.41	23.22	26	26	1.07	7.69	27	1.01	6.09
5	40	120	8	24	4.67	23.35	27	28	0.91	11.92	29	0.92	10.33
5	50	100	7	28	4.74	21.94	32	33	0.70	9.48	33	0.68	7.85
5	50	150	10	24	4.33	26.44	30	30	1.20	16.67	30	1.20	15.41
8	40	80	7	26	8.38	25.86	27	27	0.32	3.04	27	0.38	2.46
8	40	120	8	28	6.48	21.70	29	30	0.40	4.50	30	0.40	3.70
8	50	100	7	32	6.13	21.83	32	32	0.32	3.83	33	0.31	3.65
8	50	150	9	30	5.00	18.70	30	30	0.31	4.30	30	0.36	3.66

It seems natural that better values for the measures are obtained when the lower and upper bound sets contain more elements. This is probably where a standard ε-constraint method falls short since the lower bounds it computes contain very few points in comparison to those computed by the model on which PPS and IPPS are based. Better values of μ_1 and μ_2 were obtained for IPPS in comparison to PPS for the tested instances of both the BOUVRP and the BOMCTP. Since the same lower bounds were computed by both approaches for any given instance, we can attribute the better values of the measures for IPPS to the quality of the upper bounds it produces. In terms of computational times, the standard approach is the fastest and this is not so surprising given the number of points it computes and the poor quality of the bounds it produces when compared to the others. When computing the lower bounds, IPPS needs to solve fewer subproblems than PPS (see values for *dssr*) and also generates significantly more columns than (see values for *cols*). The effect is that, the computational times is significantly reduced for IPPS in comparison to PPS. Finally, since the members of an upper bound set correspond to images of feasible solutions, the results mean PPS and IPPS can be used to provide very good approximations of the nondominated set $\mathcal{Y}_N$ for the class of problems considered.

5 Conclusions

This paper discusses the application of column generation to bi-objective VRPs in which one objective is a min-max function. An idea for formulating these problems based on a variant of the ε-constraint method is presented. Instead of adding constraints on an objective

Table 4 Comparison of computational times for the BOMCTP.

			Standard			PPS			IPPS		
p	$\|V\|$	$\|W\|$	time	dssr	cols	time	dssr	cols	time	dssr	cols
5	40	80	27.5	137	790	49.5	228	1597	40.5	155	2099
5	40	120	38.9	163	1027	126.8	330	2571	94.0	201	3388
5	50	100	68.5	197	1240	205.8	390	3035	153.9	226	3459
5	50	150	42.2	150	875	392.5	486	4053	287.0	247	4054
8	40	80	61.0	217	1498	113.4	302	2283	103.6	218	2961
8	40	120	132.1	299	2205	511.2	481	3949	503.9	293	4663
8	50	100	281.2	326	2398	1343.7	522	4306	1012.5	335	5071
8	50	150	333.0	380	2872	1525.2	672	5799	1186.2	384	6005

in the master problem, we rather redefine the set of feasible columns to take the objective into account. We keep the strength of the model at the expense of a possibly more difficult problem. The advantages of using this model is clearly exhibited from the quality of the bounds obtained from it. We also investigate a strategy to accelerate and improve the column generation method. The proposed ideas are applied to two VRPs and the results obtained indicate that an intelligent management of columns in a multi-objective perspective can yield significant speedups in computing lower and upper bounds. Given that the time needed to compute such quality bounds can be very long, future works are aimed at finding a good compromise between the quality of bounds and the computational time. It will also be interesting to develop branching rules in order to explore the idea of a multi-objective branch-and-price algorithm.

References

1 Natashia Boland, John Dethridge, and Irina Dumitrescu. Accelerated label setting algorithms for the elementary resource constrained shortest path problem. *Operations Research Letters*, 34(1):58–68, 2006.

2 John R. Current and David A. Schilling. The median tour and maximal covering tour problems: Formulations and heuristics. *European Journal of Operational Research*, 73(1):114–126, 1994.

3 Charles Delort and Olivier Spanjaard. Using bound sets in multiobjective optimization: Application to the biobjective binary knapsack problem. In *Experimental Algorithms*, pages 253–265. Springer, 2010.

4 Guy Desaulniers, Jacques Desrosiers, and Marius M. Solomon. Accelerating Strategies in Column Generation Methods for Vehicle Routing and Crew Scheduling Problems. In *Essays and Surveys in Metaheuristics*, volume 15 of *Operations Research/Computer Science Interfaces Series*, pages 309–324. Springer US, 2002.

5 Matthias Ehrgott. A discussion of scalarization techniques for multiple objective integer programming. *Annals of Operations Research*, 147(1):343–360, 2006.

6 Matthias Ehrgott and Xavier Gandibleux. Bound sets for biobjective combinatorial optimization problems. *Computers & Operations Research*, 34(9):2674–2694, 2007.

7 Michel Gendreau, Gilbert Laporte, and Frédéric Semet. The Covering Tour Problem. *Operations Research*, 45(4):568–576, 1997.

8 Mondher Hachicha, M John Hodgson, Gilbert Laporte, and Frédéric Semet. Heuristics for the multi-vehicle covering tour problem. *Computers & Operations Research*, 27(1):29–42, 2000.

9 M. John Hodgson, Gilbert Laporte, and Frederic Semet. A Covering Tour Model for Planning Mobile Health Care Facilities in SuhumDistrict, Ghama. *Journal of Regional Science*, 38(4):621–638, 1998.

10 Nicolas Jozefowiez, Frédéric Semet, and El-Ghazali Talbi. The bi-objective covering tour problem. *Computers & Operations Research*, 34(7):1929–1942, 2007.

11 Martine Labbé and Gilbert Laporte. *Maximizing User Convenience and Postal Service Efficiency in Post Box Location*. Cahiers du GÉRAD. Université de Montréal, Centre de recherche sur les transports, 1986.

12 Line Blander Reinhardt and David Pisinger. Multi-objective and multi-constrained non-additive shortest path problems. *Computers & Operations Research*, 38(3):605–616, 2011.

13 Giovanni Righini and Matteo Salani. New dynamic programming algorithms for the resource constrained elementary shortest path problem. *Networks*, 51(3):155–170, 2008.

14 Francis Sourd and Olivier Spanjaard. A multiobjective branch-and-bound framework: Application to the biobjective spanning tree problem. *INFORMS Journal on Computing*, 20(3):472–484, 2008.

15 Bernardo Villarreal and Mark H. Karwan. Multicriteria integer programming: A (hybrid) dynamic programming recursive approach. *Mathematical Programming*, 21(1):204–223, 1981.

INFORMS Journal on Computing
Vol. 24, No. 4, Fall 2012, pp. 554–564
ISSN 1091-9856 (print) | ISSN 1526-5528 (online)

http://dx.doi.org/10.1287/ijoc.1110.0476

A Generic Branch-and-Cut Algorithm for Multiobjective Optimization Problems: Application to the Multilabel Traveling Salesman Problem

Nicolas Jozefowiez
CNRS, LAAS, F-31077 Toulouse, France; and Université de Toulouse, UPS, INSA, INP, ISAE, LAAS, F-31077 Toulouse, France, nicolas.jozefowiez@laas.fr

Gilbert Laporte
CIRRELT, HEC Montréal, Montréal, Québec H3T 2A7, Canada, gilbert.laporte@cirrelt.ca

Frédéric Semet
Laboratoire d'Automatique, Génie Informatique et Signal, École Centrale de Lille, Cité Scientifique, 59651 Villeneuve d'Ascq Cedex, France, frederic.semet@ec-lille.fr

This paper describes a generic branch-and-cut algorithm applicable to the solution of multiobjective optimization problems for which a lower bound can be defined as a polynomially solvable multiobjective problem. The algorithm closely follows standard branch and cut except for the definition of the lower and upper bounds and some optional speed-up mechanisms. It is applied to a routing problem called the multilabel traveling salesman problem, a variant of the traveling salesman problem in which labels are attributed to the edges. The goal is to find a Hamiltonian cycle that minimizes the tour length and the number of labels in the tour. Implementations of the generic multiobjective branch-and-cut algorithm and speed-up mechanisms are described. Computational experiments are conducted, and the method is compared to the classical ϵ-constraint method.

Key words: decision analysis, multiple criteria; programming, multiple criteria; programming, integer, algorithm, cutting plane; networks–graphs, traveling salesman
History: Accepted by Karen Aardal, Area Editor for Design and Analysis of Algorithms; received September 2009; revised August 2010, April 2011; accepted June 2011. Published online in *Articles in Advance* September 15, 2011.

1. Introduction

The purpose of this paper is to introduce a generic branch-and-cut methodology applicable to the solution of multiobjective integer linear programs; we will refer to this method by the abbreviation MOB&C. The main difference between MOB&C and standard branch and cut lies in the definition of the lower and upper bounds. The introduction of cuts is not an essential feature of the method, which works just as well in a standard branch-and-bound setting. As in classical branch-and-cut algorithms, cuts can strengthen the linear relaxation of the initial model and may increase the lower bound values. Our method applies to any problem for which the lower bound can be defined as a polynomially (or pseudo-polynomially) solvable problem. This is the case of several routing problems such as the multilabel traveling salesman problem (MLTSP), introduced later in this paper. Although the method is generic, it must be tailored to the problem at hand to achieve good results.

A multiobjective problem can be stated formally as follows:

$$\text{(MOP)} \qquad \underset{x \in D}{\text{minimize}}\ F(x) = \big(f_1(x), f_2(x), \ldots, f_n(x)\big),$$

where $n \geq 2$ is the number of objective functions, $x = (x_1, x_2, \ldots, x_r)$ is the decision variable vector or solution, D is the feasible solution set, and $F(x)$ is the objective vector. The set $O = F(D)$ corresponds to the images of the feasible solutions in the objective space, and $y = (y_1, y_2, \ldots, y_n)$, where $y_i = f_i(x)$, is a point of the objective space. The Pareto dominance between the solutions is defined as follows.

Definition 1. A solution x dominates ($\preceq$) a solution z if and only if $\forall i \in \{1, \ldots, n\}$, $f_i(x) \leq f_i(z)$ and $\exists i \in \{1, \ldots, n\}$, such that $f_i(x) < f_i(z)$.

In the rest of this paper, we sometimes write $x \preceq z$ when x or z represents directly a point in the objective space and not a solution. A feasible solution dominated by no other feasible solution is said to be efficient, or Pareto-optimal. Its image in the objective space is said to be nondominated. The set of all efficient solutions is called the efficient set, and the set of all nondominated points is the nondominated set. Solutions in an approximation of the efficient set will be said to be potentially efficient, i.e., the solutions found by a heuristic that are not dominated by another solution found by the heuristic. Note that in this paper, we do not really search for the complete efficient set, because one feasible solution for each point in the nondominated set is enough. However, it is an implementation detail, and the method remains valid if the complete efficient set is searched.

Most multiobjective exact methods use weighted sums (Ehrgott and Gandibleux 2002). The most popular is the two-phase method by Ulungu and Teghem (1995), which was later improved by Przybylski et al. (2008). It solves several weighted sum problems to find the supported solutions, i.e., the solutions whose images in the objective space are on the convex hull formed by the nondominated set, and then identifies unsupported solutions. The original two-phase method is limited to biobjective problems. Extensions to multiple-objective problems were later proposed by Ehrgott et al. (2006), Przybylski (2006), and Przybylski et al. (2010b, a). Ehrgott and Gandibleux (2002) state that the second-most popular exact methods use the ϵ-constraint strategy of Chankong and Haimes (1983). In ϵ-constraint methods only one objective is minimized, and the others are added as constraints bounded by constants (the ϵ values). Varying these constants provides several solutions. A biobjective general ϵ-constraint method was applied by Bérubé et al. (2009) to the traveling salesman problem with profits. Another application to the biobjective covering tour problem was proposed by Jozefowiez et al. (2007). An extension to problems with more than two objectives was developed by Laumanns et al. (2005). Lemesre et al. (2007) have put forward a method called 2-Parallel Partioning Method (2-PPM) to solve biobjective problems. It works by partitioning the objective space and finding one nondominated point and an associated solution in each part. The remaining solutions are then found by exploring the feasible solution set, reduced thanks to the previously identified solutions. This method was later extended to any number of objectives by Dhaenens et al. (2010). These algorithms consist one way or the other of a repeated application of a single-objective method, which can be problematic if the problem under consideration is NP-hard. As far as we know, only two references (Mavrotas and Diakoulaki 1998, Sourd and Spanjaard 2008) propose an adaptation of a standard and generic branch-and-bound algorithm to multiobjective optimization. These methods can find the efficient set in a single run. However, the method of Mavrotas and Diakoulaki (1998) uses a simple and crude lower bound consisting of the ideal point, whereas the method of Sourd and Spanjaard (2008) cannot efficiently solve the multiobjective problem if the weighted aggregation of the objectives results in an NP-hard single-objective problem, as is the case of the MLTSP, for example. This is because it would need to solve an NP-hard problem to compute the lower bound.

Our main contribution is the development of a generic multiobjective branch-and-cut algorithm. We wish to achieve this goal while modifying the generic structure of the standard branch-and-cut algorithm as little as possible, and without introducing a new multiobjective framework. A secondary contribution of this paper is the application of this method to the MLTSP.

The remainder of this paper is organized as follows. The MOB&C algorithm is presented in §2. The MLTSP is introduced, modeled, and solved in §3. Computational results follow in §4, and conclusions are provided in §5.

2. A Generic Multiobjective Branch-and-Cut Algorithm

The MOB&C algorithm we propose closely follows the standard branch-and-cut algorithm. This means that unlike most exact multiobjective methods, it does not iteratively solve a sequence of single-objective problems, which are often NP-hard. Rather, it constructs the efficient set in a single run. Moreover, at a given node of the search tree, a diversified lower bound consisting of several nondominated points in the objective space and the associated, not necessarily feasible, solutions is searched. It can therefore be used heuristically as an anytime algorithm (Dean and Boddy 1988) that can be stopped to return an approximation of the efficient set. In this sense, it can be compared with multiobjective evolutionary algorithms that work on a population of solutions and offer a well-diversified approximation (Deb 2001, Coello Coello et al. 2002). Such methods constitute the main class of metaheuristics for multiobjective problems. The MOB&C algorithm is summarized in Algorithm 1. We will now define the lower and upper bounds and show how a subtree can be pruned. We will also discuss some particular aspects of branching, pruning, and lower bound computation in a multiobjective context.

Algorithm 1 (Multiobjective branch-and-cut algorithm (MOB&C))

Step 1 **(Root of the tree)**
Generate an initial upper bound *ub*.
Define a first subproblem.
Insert the subproblem in a list Λ.

Step 2 **(Stopping criterion)**
If $\Lambda = \varnothing$, then STOP; else choose a subproblem from Λ and remove it from Λ.

Step 3 **(Subproblem solution)**
Solve the subproblem to obtain the lower bound *lb*.

Step 4 **(Constraint generation)**
If some integer solutions have been found, try to insert them in *ub*.
if $ub \preceq lb$, **then**
  Go to Step 2.
else
  if violated constraints are identified for fractional solutions associated to points of the lower bound, **then**
    Add them to the model and go to Step 3.
  else
    Go to Step 5.
  end if
end if

Step 5 **(Branching)**
Branch on variables and introduce new subproblems in Λ. Go to Step 2.

2.1. Lower and Upper Bounds

Computing lower and upper bounds for multiobjective problems can be rather intricate; see, for instance, Ehrgott and Gandibleux (2007). We define the lower bound (*lb*) as a set of points in the objective space such that any point corresponding to a feasible solution is either dominated by at least one of these points or nondominated. We suppose that each point in the lower bound can be associated to a solution, not necessarily feasible, and that these solutions constitute the set $S_{lb} = \{x \mid F(x) \in lb\}$. The upper bound (*ub*) is a set of points in the objective space corresponding to feasible solutions that do not dominate each other. We define $S_{ub} = \{x \in D \mid F(x) \in ub\}$ as the set of feasible solutions corresponding to a point in the upper bound set. In the rest of the paper, we will consider the partition of S_{lb} into two subsets with respect to S_{ub}: $S_{lb}^{\succ} = \{s \in S_{lb} \mid \exists y \in S_{ub},\ y \preceq s\}$ and $S_{lb}^{\prec} = \{s \in S_{lb} \mid \nexists y \in S_{ub},\ y \preceq s\}$. Note that solutions in $S_{lb}^{\prec}$ are necessarily fractional because integer-infeasible solutions are eliminated through the use of cutting planes.

The computation of the lower bound should remain polynomial or at worst pseudo-polynomial. Therefore, two goals should be kept in mind while defining the lower bound: (i) the number of points in *lb* should be kept polynomial or pseudo-polynomial with respect to the size of the problem, and (ii) the method to generate one or several points of *lb* should be polynomial or pseudo-polynomial. The principles exposed in this paper remain true if the generation of the lower bound is not polynomial, but the method would rapidly become intractable. The design of the lower-bound generation algorithm is clearly problem dependent. The upper bound is managed as an archive of found solutions not dominated by other solutions found by MOB&C. The archive of identified solutions is updated whenever feasible solutions are found.

In multiobjective integer problems, the lower bound can be defined by a set Φ of subproblems built from the linear relaxation of the integer program by means of scalarization techniques. As a matter of fact, the size of Φ may not be bounded. For instance, if the aggregation approach is used, the number of possible weight combinations is infinite, but only a subset is solved as in the first phase of the two-phase method (Ulungu and Teghem 1995). Therefore, if the lower bound is defined by Φ, its computation may only require the solution of a subset $\bar{\Phi} \subseteq \Phi$. If $|\bar{\Phi}|$ is kept polynomial or pseudo-polynomial, then our goal of computing the lower bound within a reasonable time will be achieved. Each solution in $s \in S_{lb}$ can be associated to a subset $\Phi_s \subseteq \Phi$. An algorithm for a problem $\phi \in \Phi_s$ returns s or a solution s' such that $F(s) = F(s')$ (we suppose that we keep only one solution for each point in *lb*).

Because we solve linear relaxations, the points in *lb* may correspond to fractional solutions, and cutting planes may be used to speed up the search. This is why the method is called a multiobjective branch-and-cut algorithm even if the cutting part is not essential.

This strategy can be used, for instance, on problems in which every objective but one is a min-max objective that can take a polynomial or a pseudo-polynomial number of values, such as the biobjective covering tour problem (Jozefowiez et al. 2007). This approach can also be applied to any problem that can be modelled as a multiobjective integer program and could be solved by exact methods relying on scalarization techniques such as those of Bérubé et al. (2009) and of Dhaenens et al. (2010). However, instead of iteratively solving NP-hard problems, our approach solves linear programs. An example of a problem that can be tackled this way is the MLTSP presented later. An example of a real-life problem that could be solved by MOB&C is the vehicle routing problem arising in rural Spain described by Corberán et al. (2002) and Pacheco and Martí (2006). The method can also be applied to problems where the lower bound can be expressed as a pseudo-polynomial problem. Examples

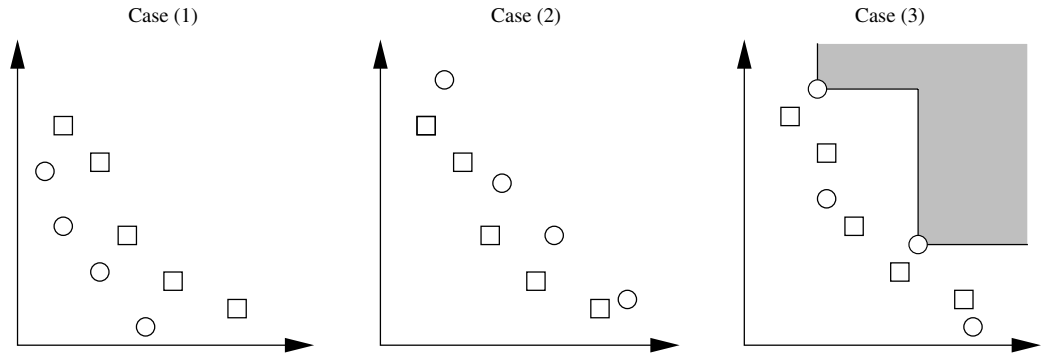


Figure 1 Dominance Between the Lower Bound and the Upper Bound
Note. The squares represent the upper bound, and the circles represent the lower bound.

include the traveling salesman problem with profits (Feillet et al. 2005, Jozefowiez et al. 2008, Bérubé et al. 2009) and the ring star problem (Labbé et al. 2004; Liefooghe et al. 2008a, b).

2.2. Pruning

Pruning can be performed if any of the three following cases is satisfied: (i) infeasibility: $lb = \varnothing$; (ii) optimality: $\forall x \in S_{lb}$, $x \in D$; or (iii) dominance: $ub \preceq lb$.

Dominance between two sets S_1 and S_2 is defined as follows: $S_1 \preceq S_2 \Leftrightarrow \forall x \in S_2$, $x \in S_1$ or $\exists y \in S_1$, $y \preceq x$. This is illustrated in Figure 1. In Case (1), the lower bound dominates the upper bound, and the search must be continued. Conversely, in Case (2), the lower bound is dominated by the upper bound, and the node can be pruned. Case (3) is an example where neither the lower bound nor the upper bound dominates the other.

2.3. Partial Pruning

In standard branch and cut, if the lower bound exceeds the upper bound or yields an integer solution, the node is pruned. In the multiobjective case, the lower bound may consist of points that are dominated by the upper bound and of points that are fractional and nondominated (e.g., Case (3) in Figure 1) for the subtree (i.e., given the branching choices made). An area of the objective space necessarily dominated for the current subtree (i.e., given the branching choices made) can be built from $S_{lb}^{\succ}$. This area is composed of the points in the objective space dominated by the images in the objective space of the solutions in $S_{lb}^{\succ}$. In Figure 1, Case (3), this is represented by the grey area. Therefore, points located in these areas should not be generated because this part of the lower bound does not need to be tightened. Avoiding the generation of points for the lower bound in this area strongly depends on the problem at hand and on the lower bound computation method. For instance, contraints can be added, and ad hoc mechanisms can also be designed. An instance will be provided for the MLTSP where a list containing the values of the second objective for which the solutions are dominated by the upper bound is maintained.

2.4. Parallel Branching

As in a standard branch-and-bound or branch-and-cut algorithm, different strategies can be used to select the branching variable. First, note that only the solutions in $S_{lb}^{\prec}$ need to be considered in this step. Because the lower bound is not composed of a single solution but of a set of solutions, a new difficulty arises. Indeed, there may not be a unique fractional variable for all the solutions s in $S_{lb}^{\prec}$. Moreover, even if a variable is fractional for several solutions in $S_{lb}^{\prec}$, its values may not be the same, leading to several intervals to be excluded. Hence, there may not be a unique variable that will define a *valid* branching for all fractional solutions. This means that not all solutions in $S_{lb}^{\prec}$ may be affected by the branching.

Suppose that branching is performed on a fractional variable χ by excluding an interval $]\alpha, \beta[$ with $\alpha \in \mathbb{N}$ and $\beta = \alpha + 1$. Denote by χ^s the value of χ in $s \in S_{lb}^{\prec}$. The set $S_{lb}^{\prec}$ can be partitioned into three subsets: (i) $S_{lb}^{\preceq\chi} = \{s \in S_{lb}^{\prec} \mid \chi^s \leq \alpha\}$, (ii) $S_{lb}^{=\chi} = \{s \in S_{lb}^{\prec} \mid \alpha < \chi^s < \beta\}$, and (iii) $S_{lb}^{\succeq\chi} = \{s \in S_{lb}^{\prec} \mid \chi^s \geq \beta\}$. When $\chi \leq \alpha$ is imposed, solutions in $S_{lb}^{\preceq\chi}$ will not be affected. Note that imposing $\chi \leq \alpha$ does not always yield an effective constraint because the value χ^s may already be integer for some solution $s \in S_{lb}^{\succeq\chi}$. A similar reasoning can be applied to the subproblem in which $\chi \geq \beta$ is imposed.

It would be more efficient if all the solutions in $S_{lb}^{\prec}$ were affected in the subproblem. Here, parallel branching could be performed. To this end, branching choices would need to be tracked separately for each problem of Φ. More formally stated, k distinct couples $(\chi_\kappa,]\alpha_\kappa, \beta_\kappa[)$ $(\kappa \in \{1, \ldots, k\})$ should be selected such that (i) $\bigcup_k S_{lb}^{=\chi_k} = S_{lb}^{\prec}$; (ii) $\bigcap_k S_{lb}^{=\chi_k} = \varnothing$. The first subproblem (respectively, the second subproblem) to be in included in the list L is created as follows: for all $\kappa \in \{1, \ldots, k\}$, for all $s \in S_{lb}^{=\chi_\kappa}$, and for all $\phi \in \Phi_s$,

the constraint $\chi_\kappa \leq \alpha_\kappa$ (respectively, $\chi_\kappa \geq \beta_\kappa$) is added to ϕ.

2.5. Numerical Example

Consider the following multiobjective integer program:

$$\text{minimize} -1.00x_1 - 0.64x_2, \quad (1)$$

$$\text{minimize}\, x_3, \quad (2)$$

$$50x_1 + 31x_2 \leq 250, \quad (3)$$

$$3x_1 - 2x_2 \geq -4, \quad (4)$$

$$x_1 + x_3 \leq 2, \quad (5)$$

$$x_1, x_2 \geq 0 \quad \text{and integer}, \quad (6)$$

$$x_3 \in \{0, 1, 2\}. \quad (7)$$

We build $\Phi = \{\phi_\epsilon, \epsilon \in \{0, 1, 2\}\}$ as follows: subproblem ϕ_ϵ is obtained by relaxing the multiobjective integer program and setting x_3 to ϵ. We will denote the solution of ϕ_ϵ by s_ϵ.

Figure 2 represents the search tree if branching is performed on the most fractional variable. We suppose that each problem in Φ is solved at each node. Here, we branch first on x_2 according to its value in s_1. The left subproblem ($x_2 \leq 3$) is then pruned by optimality. In the right subproblem ($x_2 \geq 4$), we branch with respect to x_1. The efficient set is found within five nodes and requires the solution of 15 linear programs.

Because s_2 is integer at the root of the tree, it is efficient. Therefore, branching will always lead to a dominated or infeasible solution for ϕ_2. Similarly, in the right subproblem at the first level of the tree, s_1 becomes infeasible, and therefore it is not useful to compute it again in the subproblems. Partial pruning can be used to avoid these useless computations, leading to the search tree presented in Figure 3. Here, the efficient set is still found within five nodes, but it only requires the solution of nine linear programs.

Concerning branching, it appears in this example that in the right subproblem at the second level of the tree in which $x_2 \geq 4$ is imposed, ϕ_0 is not affected, and the computation leads to the same solution as at the root node. This is one of the situations that parallel branching helps to avoid. Figure 4 shows the search if parallel branching is applied. For s_0, we branch on x_2 by imposing $x_2 \leq 4$ in one subproblem and $x_2 \geq 5$ in the other. We also branch on x_2 for s_1, but we exclude the interval $]3, 4[$. By choosing a relevant branching for each subproblem, we make each computation count. For example, the optimal solution for ϕ_0 is found directly in the first subproblem, whereas it is found almost at the end in Figure 2. Here, the tree is made up of three nodes and requires the solution of seven linear programs.

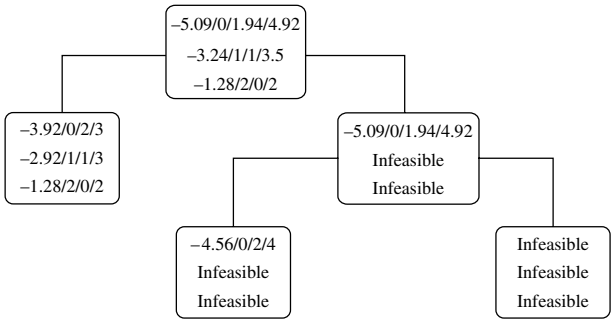


Figure 2 Standard Search Tree

Note. The first line (respectively, the second and third lines) gives the value of both objectives and of x_1 and x_2 for s_0 (respectively, s_1 and s_2).

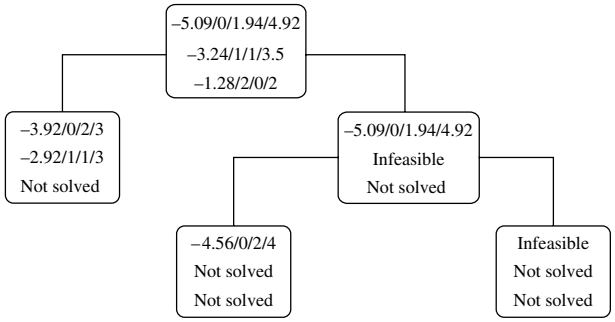


Figure 3 Search Tree with Partial Pruning

Note. The first line (respectively, the second and third lines) gives the value of both objectives and of x_1 and x_2 for s_0 (respectively, s_1 and s_2).

3. Application to the Multilabel Traveling Salesman Problem

We now describe the application of MOB&C to the biobjective MLTSP. The single-objective version was studied in Jozefowiez et al. (2011). This variant of the symmetric traveling salesman problem (TSP) can be defined as follows. Let $G = (V, E)$ be an undirected graph, where V is the vertex set and E is the edge set, and let L be a set of labels (sometimes called colors; see Xiong et al. 2007). Edge $e \in E$ has a cost c_e and a label $\delta(e) \in L$.

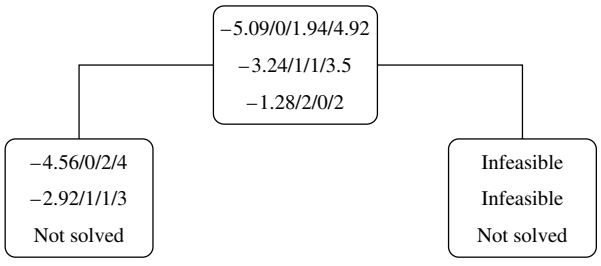


Figure 4 Search Tree with Partial Pruning and Parallel Branching

Note. The first line (respectively, the second and third lines) gives the value of both objectives and of x_1 and x_2 for s_0 (respectively, s_1 and s_2).

Labels can be interpreted as different transportation companies or production technologies, depending on the context. The goal is to determine a Hamiltonian cycle of least length and least number of labels. A label $k \in L$ is said to be used if an edge $e \in \zeta(k)$ belongs to the cycle with $\zeta(k) = \{e \in E \mid \delta(e) = k\}$. To our knowledge, this problem has never been studied as a biobjective problem but only as a single-objective problem called the minimum labelling Hamiltonian cycle problem or the colorful traveling salesman problem (Cerulli et al. 2006, Xiong et al. 2007), in which only the number of labels is minimized.

3.1. Mathematical Model for the MLTSP

The MLTSP can be formulated as an extension of the Dantzig et al. (1954) model for the TSP. We define binary variables x_e equal to 1 if and only if e is used and binary variables u_k equal to 1 if and only if $k \in L$ is used. The problem is then

$$\text{minimize} \sum_{e \in E} c_e x_e, \tag{8}$$

$$\text{minimize} \sum_{k \in L} u_k, \tag{9}$$

$$\sum_{e \in \omega(\{i\})} x_e = 2 \quad (i \in V), \tag{10}$$

$$\sum_{e \in \omega(S)} x_e \geq 2 \quad (S \subset V, 3 \leq |S| \leq |V| - 3), \tag{11}$$

$$x_e \leq u_{\delta(e)} \quad (e \in E), \tag{12}$$

$$x_e \in \{0, 1\} \quad (e \in E), \tag{13}$$

$$u_k \in \{0, 1\} \quad (k \in L), \tag{14}$$

where $\omega(S) = \{e = (i, j) \in E \mid i \in S,\ j \in V \backslash S\}$.

Objective (8) is the minimization of the cycle length, and objective (9) is the minimization of the number of labels used in the solution. Constraints (10) are degree constraints, and constraints (11) are connectivity constraints. Constraints (12) ensure that if an edge $e \in E$ is used, then its label $\delta(e)$ is also used. Constraints (13) and (14) are binary constraints. We have proven in Jozefowiez et al. (2011) that the following inequalities are valid:

$$u_k \leq \sum_{e \in \zeta(k)} x_e \quad (k \in L), \tag{15}$$

$$\sum_{e \in \omega(\{i\}) \cap \zeta(k)} x_e \leq 2u_k \quad (i \in V,\ k \in L,$$

$$3 \leq |\omega(\{i\}) \cap \zeta(k)| \leq |V| - 3), \tag{16}$$

$$\sum_{e \in E(S) \cap \zeta(k)} x_e \leq (|S| - 1)u_k \quad (k \in L,\ S = \{i, j, k\} \subset V$$

$$\forall e \in S \times S,\ u_{\delta(e)} = k), \tag{17}$$

$$\sum_{k \in L} \gamma_k(S) u_k \geq 2 \quad (S \subset V,\ 3 \leq |S| \leq |V| - 3), \tag{18}$$

with

$$\gamma_k(S) = \begin{cases} |\omega(S) \cap \zeta(k)| & \text{if } |\omega(S) \cap \zeta(k)| \leq 1, \\ 2 & \text{otherwise.} \end{cases}$$

Constraints (15) mean that if a label k is used, then at least one of the edges of label k must also be used. Constraints (16) are a consequence of the degree constraints, and constraints (17) are a consequence of the subtour elimination constraints. They state that no more than two edges having the same label can be incident to the same vertex. Constraints (18) are connectivity constraints based on labeling variables. They follow from the fact that for any partition $\{S, V \backslash S\}$ of V, labels must be chosen to ensure that the solution contains at least two edges between S and $V \backslash S$.

3.2. Heuristic for the MLTSP

We have developed the following heuristic to compute the initial upper bound. It follows an ϵ-constraint method (Chankong and Haimes 1983), which consists of replacing objective (9) with the constraint

$$\sum_{k \in L} u_k \leq \epsilon, \tag{19}$$

and then solving the problem for different values of ϵ. This problem P^* consists of the following objective, where m is a small constant equal to 10^{-5} in our implementation:

$$\text{minimize} \sum_{e \in E} c_e x_e + m \sum_{k \in L} u_k, \tag{20}$$

subject to constraints (10), (12), (14)–(17), (19), and the linear relaxation of constraints (13). Constraints (15), (16), and (17) are directly added to the model because there is a polynomial number of them. Objective (20) is used instead of (8) to avoid dominated points.

Initially, a TSP is solved to bound the possible values of ϵ. The problem is solved for a given ϵ, and a graph G' is built based on the obtained solution. G' is the same graph as G except for the cost of the edges. For e, c'_e is equal to c_e if $u_{\delta(e)} = 1$ and to $|V| \max_{e \in E} c_e$ (i.e., an upper bound on the maximum tour length) otherwise. The large cost is introduced to disfavor edges for which $u_{\delta(e)} = 0$ but may be necessary to ensure the presence of a Hamiltonian cycle. Then, a minimum length Hamiltonian cycle is sought on the graph G'. The resulting tour is then considered as a candidate for inclusion in the upper bound set ub. Two cases can happen here. In the first case, the cardinality of U, the set of labels used in the cycle, can be smaller than ϵ, and the next value for ϵ will be $|U| - 1$. If $|U|$ is greater than ϵ, then we try $\epsilon - 1$. This process is summarized in Algorithm 2.

Algorithm 2 (Computation of the upper bound for the MLTSP)

Set $ub \leftarrow \varnothing$
Solve a TSP on G. Add the obtained solution in ub. Let U be the set of labels used in the cycle.
Set $\epsilon \leftarrow |U| - 1$
while $\epsilon \neq 0$ **do**
  Solve P^* and build the corresponding graph G'.
  if P^* has a solution, **then**
    Solve a TSP on G'. Try to add the obtained solution in ub. Let U be the set of labels used in the cycle.
    $\epsilon \leftarrow \min\{\epsilon - 1, |U| - 1\}$
  else
    $\epsilon \leftarrow 0$
  end if
end while

3.3. Implementation of the MOB&C

The implementation features specific to Algorithm 1 are the definition of the lower bound, the partial pruning mechanism, the parallel branching strategy, and the constraint-generation phase.

3.3.1. Definition of the Lower Bound. At the root of the search tree, $\Phi = \{\phi_\epsilon \mid \epsilon \in \{1, \ldots, |C|\}\}$. The problem ϕ_ϵ is defined by objective (20), constraints (10), (12), (15)–(17), (19), and the linear relaxation of constraints (13) and (14). The computation of this bound is polynomial. In the remainder of this section, we will consider that the solutions $s_i = (u^i, x^i) \in S_{lb}$ are sorted: if $1 \leq i < j \leq |S_{lb}|$, then $\sum_{k \in L} u_k^i < \sum_{k \in L} u_k^j$. Note that we necessarily have $\sum_{e \in E} c_e x_e^i > \sum_{e \in E} c_e x_e^j$. We will now explain how we have implemented the partial pruning and parallel branching mechanisms. Then, we will explain the constraint-generation process. Finally, we will show that only a subset of Φ needs to be solved to generate lb in order to avoid useless computations.

3.3.2. Partial Pruning. To avoid useless computations, we use a list, Λ_{pruned}, to keep track of the values of ϵ for which the solution of ϕ_ϵ will necessarily be dominated by the upper bound or for which ϕ_ϵ is infeasible. At the root node, Λ_{pruned} is empty. It is updated according to the lower bound and passed to the descendant subproblems and restored during backtracking. The updating is done as follows:

- $\Lambda_{\text{pruned}} \leftarrow \Lambda_{\text{pruned}} \cup \{1, \ldots, \lfloor \sum_{k \in L} u_k^1 \rfloor\}$;
- If $s_{|S_{lb}|} \in S_{lb}^{\succ}$, $\Lambda_{\text{pruned}} \leftarrow \Lambda_{\text{pruned}} \cup \{\lceil \sum_{k \in L} u_k^{|S_{lb}|} \rceil, \ldots, |L|\}$;
- $\forall 1 \leq i < |S_{lb}|$, if $s_i \in S_{lb}^{\succ}$, then $\Lambda_{\text{pruned}} \leftarrow \Lambda_{\text{pruned}} \cup \{\lceil \sum_{k \in L} u_k^i \rceil, \ldots, \lfloor \sum_{k \in L} u_k^{i+1} \rfloor\}$.

3.3.3. Parallel Branching. To enable parallel branching, we use a matrix B of size $|L| \times (|L| + |E|)$. $B[r, y]$ indicates for ϕ_r whether variable y is fixed to 0 or 1, or is free. At the start of the search, $B[k, y] =$ free for all $k \in \{1, \ldots, |L|\}$ and for all variables y. The matrix is updated as follows after the computation of the lower bound. We start by choosing the first variable to branch on. Let $k = \arg\max_{k \in L}(|S_{lb}^{=u_k}|)$. If $|S_{lb}^{=u_k}| = 0$, all u variables are integer, and we consider the x variables. Then, $e = \arg\max_{e \in E}(|S_{lb}^{=x_e}|)$. If $|S_{lb}^{=u_k}| > 0$, let $v = u_k$; otherwise, $v = x_e$.

Then, for each solution s_i in $S_{lb}^{=v}$, and each value r in $\{\lceil \sum_{k \in L} u_k^i \rceil, \ldots, \lfloor \sum_{k \in L} u_k^{i+1} \rfloor\}$ for which $B[r, v] =$ free, we set $B[r, v] = 0$ in one subproblem and $B[r, v] = 1$ in the other. If $s_{|S_{lb}|} \in S_{lb}^{=v}$, we select r in $\{\lceil \sum_{k \in L} u_k^i \rceil, \ldots, |L|\}$. If $S_{lb}^{\prec} \backslash S_{lb}^{=v} = \varnothing$, a branching variable has been chosen for each nondominated fractional solution, and we proceed to the next subproblem; otherwise, we set $S_{lb}^{\prec} \leftarrow S_{lb}^{\prec} \backslash S_{lb}^{=v}$ and reiterate the process.

3.3.4. Constraint Generation. When for a given ϵ value the solution to the subproblem is neither integer nor dominated by the upper bound, violated constraints (11) are searched by identifying the connected components and the minimum cut in each component. A constraint is then generated and added for each connected component (if there is more than one) and each minimum cut of value strictly less than 2. Each subset generated this way is also used to verify whether the associated constraints (18) are violated or not. The generated constraints are added to all $\phi \in \Phi$.

3.3.5. Computation of the Lower Bound. A relevant observation is that it is not necessary to solve all the problems in Φ. If for a given value ϕ_{ϵ_0} we obtain a solution s for which $\sum_{k \in L} u_k^s = \epsilon_1 \leq \epsilon_0 - 1$, then solving the subproblems $\phi_{\epsilon \in [\epsilon_1, \epsilon_0 - 1]}$ will provide the same point for the lower bound. These computations should be avoided. The relevant values of ϵ are determined by applying Algorithm 3. Problem ϕ_ϵ is solved by means of a cutting plane approach described in Algorithm 4. This algorithm also updates Λ_{pruned} and the upper bound, and it determines whether the node can be pruned or not.

More precisely, Algorithm 3 works as follows. Starting from an allowed value for ϵ, it solves ϕ_ϵ using Algorithm 4. If a solution is found, ϵ is fixed to the next integer strictly smaller than the value of objective (9), and the search continues. It stops when ϵ reaches a value for which there is no solution. At the end of the search, if *pruned* is true, then the node can be declared explored because the lower bound is dominated by the upper bound or because the problem has no solution.

Algorithm 3 (Computation of the lower bound for the MLTSP)

ub is the upper bound
Λ_{pruned} is the pruned list
Set *pruned* ← TRUE
Set $\epsilon \leftarrow \alpha$ with $\alpha = \max\{\alpha \in \mathbb{N} \mid \alpha \notin \Lambda_{\text{pruned}}$ and $\alpha \leq |C|\}$.
while $\epsilon > 0$ **do**
  Solve ϕ_ϵ (Algorithm (4))
  Let o^* be the number of labels used in the optimal solutions, or 0 if there was no solution.
  Set $\epsilon \leftarrow \alpha$ with $\alpha = \max\{\alpha \in \mathbb{N} \mid \alpha \notin \Lambda_{\text{pruned}}$ and $\alpha \leq o^*\}$.
end while

Algorithm 4 (Solution of the lower bound and constraint generation)

Set *continue* ← TRUE
while *continue* is TRUE **do**
  continue ← FALSE
  Solve ϕ_ϵ.
  Let o^* be the number of labels used in the optimal solutions, or 0 if there was no solution.
  if a solution is found, **then**
    if the solution is feasible and integer or the solution is dominated by *ub*, **then**
      if the solution is feasible and integer, **then**
        Try to add it in *ub* and update *ub* if necessary
      end if
      $\Lambda_{\text{pruned}} \leftarrow \Lambda_{\text{pruned}} \cup \{\lceil o^* \rceil, \ldots, \epsilon\}$
    else
      if violated constraints are identified, **then**
        Add them to the model
        continue ← TRUE
      else
        pruned ← FALSE
      end if
    end if
  else
    $\Lambda_{\text{pruned}} \leftarrow \Lambda_{\text{pruned}} \cup \{1, \ldots, \epsilon\}$
  end if
end while

Algorithm 4 first solves the linear program ϕ_ϵ. If there is no solution, it stops and includes the values of ϵ between 1 and the current value of ϵ in Λ_{pruned}. Indeed, for these values, there will never be a solution to the linear program. If there is a solution that is integer or dominated by the upper bound, it also stops and adds the values between the value of the second objective and the current value of ϵ in Λ_{pruned}, because they can only provide a solution dominated by the upper bound. If the solution is integer, it is considered a candidate for inclusion in the upper bound set. Otherwise, a search for violated constraints is carried out. If some violated constraints are found, they are added to the model, and the process is reiterated; otherwise, the search stops with fractional solutions that are not dominated by the upper bound. In this case, the node cannot be pruned, and branching must be initiated.

3.4. Comparison with the ϵ-Constraint Method

To assess the efficiency of MOB&C, it was compared to the ϵ-constraint method. The latter algorithm was implemented to solve the MLTSP by iteratively applying a branch-and-cut algorithm for a single-objective variation of the MLTSP in which the length is minimized and the number of used labels is bounded by ϵ. This branch-and-cut algorithm is standard and uses the same model and the same cuts as the biobjective algorithm. The values of ϵ are chosen using the same process as when the multiobjective linear relaxation of the MLTSP is solved. At the start of the algorithm, a potentially efficient set is computed using the heuristic for the MLTSP described in §3.2. This set is used to provide an initial upper bound for the branch-and-cut algorithm. This initial upper bound is the length of the solution with the largest number of used labels less than or equal to ϵ. Whenever the branch-and-cut algorithm identifies an integer solution, an attempt is made to add it to the upper bound. Also, the identified violated constraints during the solution for a given value of ϵ are kept for the remaining part of the ϵ-constraint algorithm.

4. Computational Results

The algorithm was coded in C and executed on an Intel Core 2 Duo E6550 2.33 GHz CPU. Linear programs were solved using CPLEX 11.1.2, and TSP was solved using the Concorde library (http://www.tsp.gatech.edu/concorde/index.html).

The algorithm was tested on a series of randomly generated benchmarks. Tests were conducted for $|L| = 20, 30, 40, 50$ and $|V| = 20, 30, 40, 50$. For each value of V, five different graphs were generated, and for each combination of $|C|$ and $|V|$, five different label settings were created.

Results are reported in Table 1. Each line represents the average values over 25 instances. For each instance size, we report the average size of the efficient set (#Par). For the MOB&C algorithm and the ϵ-constraint method (ϵCM), the number of nodes (#Nodes), the number of generated cuts (#Cuts), and the time in seconds (Seconds) are given. For the MOB&C algorithm, we also report the average time in seconds to identify the efficient set (Seconds*). For the ϵ-constraint method, the number of nodes is the sum of nodes expanded in all branch-and-cut algorithms launched.

Table 1 Summary of Computational Results for the Branch-and-Cut Algorithm

			MOB&C				ϵCM		
$\|L\|$	$\|V\|$	#Par	#Nodes	#Cuts	Seconds	Seconds*	#Nodes	#Cuts	Seconds
20	20	10.3	227.6	56.2	1.4	1.0	561.2	59.2	1.4
20	30	13.9	452.2	130.9	8.8	6.6	1,225.0	128.8	9.5
20	40	15.9	839.2	231.3	34.6	25.0	2,206.4	232.0	33.6
20	50	17.4	1,509.6	346.3	96.1	69.8	3,248.2	324.8	100.3
30	20	12.4	418.3	87.0	2.9	2.1	1,320.6	100.8	3.8
30	30	16.4	1,006.8	188.8	26.3	19.3	3,463.2	207.0	31.7
30	40	18.8	2,552.0	388.2	177.1	126.8	6,745.8	384.9	170.9
30	50	21.7	4,735.3	588.2	431.0	286.9	12,920.3	582.9	606.5
40	20	12.1	606.8	98.9	4.2	3.1	1,571.0	107.6	5.0
40	30	17.8	1,913.0	258.9	58.7	42.7	5,806.0	273.4	67.2
40	40	21.7	4,406.6	548.2	503.0	349.8	17,462.0	578.3	665.8
40	50	26.6	15,360.6	926.0	1,845.9	1,374.5	45,306.6	1,037.9	3,334.5
50	20	12.4	718.9	102.8	4.4	3.4	2,296.6	116.1	6.8
50	30	18.8	3,248.3	355.0	144.0	110.2	12,687.6	428.1	224.9
50	40	23.9	8,722.7	738.3	1,374.4	1,097.7	36,339.4	797.5	1,636.9
50	50	27.7	20,680.3	1,204.9	4,094.0	2,902.5	74,336.6	1,307.5	5,938.4

The main conclusion is that MOB&C is faster, on average, than the ϵ-constraint method. The relative difference in performance increases with instance size (especially the number of labels). Our algorithm explored far fewer nodes than the ϵ-constraint method. Again, the difference increases with instance size, which suggests MOB&C will be more efficient on larger instances. We believe this is because some variable branchings are performed several times in the ϵ-constraint method, since for each value of ϵ, the optimization method starts with a clean state regarding the branching possibilities. This is not the case for MOB&C, which is able to improve the upper bound for all values of ϵ in a single step. The fact that MOB&C remains faster, on average, tends to show that the Λ_{pruned} list management mechanism and the multiple variable branching are effective in avoiding useless computations. There does not seem to be a significant difference in the number of cuts needed by the two methods. However, on larger instances, the difference seems to increase on average in favor of MOB&C.

On larger instances, it seems that the proposed method can be stopped earlier than the ϵ-constraint method because the efficient set is identified a long time before optimality can be proved. This can easily be observed on instances with 50 labels and 50 nodes. To better understand this, and to see whether MOB&C could be stopped earlier and could still provide a good approximation of the efficient set, we have stopped the algorithm after a certain percentage of the nodes in the search tree have been explored. Table 2 reports the results obtained when this percentage is 25%, 50%, and 75%. In addition to the number of solutions in the efficient sets (#Par), the information given for each percentage is the size of the set of potentially efficient solutions found (Size) and the percentage of the solutions from the efficient set retrieved (%). The column "Gap" is computed as follows. For each solution $r = (u^r, x^r)$ in the approximation A, we identify the solution (u^{r*}, x^{r*}) in the efficient set such that $\sum_{k\in L} u_k^{r*} - \sum_{k\in L} u_k^r$ is positive and as small as possible. We then compute $g_r = \sum_{e\in E} c_e x_e^r / \sum_{e\in E} c_e x_e^{r*}$, and the value of gap is equal to $\sum_{r\in A} g_r/|A|$. It appears that MOB&C is able to find a significant part of the efficient set. For instance, if we explore only 50% of the nodes, on average, more than half of the efficient set is found. Also, the fact that the size of the approximation is almost the size of the efficient set and that the gap is less than 1% indicates that the approximation is close to the efficient set. The results are not as good when only 25% of the nodes are explored, but they remain interesting. Indeed, the number of efficient solutions found drops, but the gap between the approximation and the optimal set is never large. When 75% of the nodes are explored, the approximation is even better, and the corresponding computation time (Seconds) is smaller than 75% of the average computational time reported in Table 1. This indicates that MOB&C can efficiently be used heuristically to provide a good-quality approximation, whereas this may not be possible for existing exact methods that mainly iterate between single-objective solutions for a given pattern.

We report in Table 3 the number of identified solutions in the efficient set (#Par), the number of potentially efficient solutions found by the heuristic (#Parub), the percentage of efficient solutions with respect to #Par (%) found by the heuristic, and the time in seconds (Seconds). The column "Gap" is defined as before but between the approximation identified by the heuristic and the efficient set.

Table 2 Summary of Computational Results for the Truncated Branch-and-Cut Algorithm

			25%			50%			75%			
$\|L\|$	$\|V\|$	#Par	Size	%	Gap	Size	%	Gap	Size	%	Gap	Seconds
20	20	10.3	10.0	60.2	1.010	10.3	72.8	1.006	10.3	88.3	1.002	0.9
20	30	13.9	13.8	56.8	1.007	13.9	71.9	1.004	13.9	90.6	1.001	5.1
20	40	15.9	15.6	47.8	1.008	15.8	65.4	1.005	15.8	83.6	1.002	17.0
20	50	17.4	17.0	46.6	1.006	17.3	60.9	1.004	17.3	79.9	1.002	48.2
30	20	12.4	12.0	62.1	1.009	12.2	76.6	1.004	12.3	92.7	1.001	1.9
30	30	16.4	16.2	49.4	1.008	16.4	67.7	1.004	16.4	89.6	1.001	14.3
30	40	18.8	18.4	38.3	1.010	18.7	51.6	1.007	18.7	75.5	1.002	78.0
30	50	21.7	21.0	40.6	1.009	21.4	55.3	1.005	21.6	82.5	1.002	188.8
40	20	12.1	11.5	58.7	1.011	11.9	76.0	1.005	12.0	87.6	1.002	2.7
40	30	17.8	17.3	41.6	1.010	17.7	62.9	1.005	17.8	83.7	1.002	30.0
40	40	21.7	21.2	31.3	1.011	21.6	43.8	1.007	21.7	80.2	1.002	200.3
40	50	26.6	25.4	34.2	1.009	26.2	51.9	1.006	26.4	71.8	1.003	708.0
50	20	12.4	11.8	59.7	1.011	12.2	69.4	1.009	12.3	84.7	1.004	2.9
50	30	18.8	18.3	41.0	1.012	18.6	63.8	1.005	18.6	86.2	1.002	75.4
50	40	23.9	23.1	34.3	1.011	23.9	51.9	1.005	23.8	82.0	1.002	601.8
50	50	27.7	26.5	24.5	1.012	27.2	40.8	1.007	27.6	69.7	1.003	1,679.9

Table 3 Summary of Computational Results for the Heuristic

$\|L\|$	$\|V\|$	#Par	#Parub	%	Gap	Seconds
20	20	10.3	9.2	43.7	1.018	0.0
20	30	13.9	12.6	33.8	1.012	2.4
20	40	15.9	14.8	34.0	1.011	7.1
20	50	17.4	16.2	33.3	1.008	16.4
30	20	12.4	9.4	36.3	1.016	0.5
30	30	16.4	13.6	28.0	1.014	4.1
30	40	18.8	15.7	22.3	1.013	14.4
30	50	21.7	18.6	23.5	1.010	37.1
40	20	12.1	8.9	33.9	1.021	0.8
40	30	17.8	13.0	20.8	1.015	6.3
40	40	21.7	16.6	16.1	1.015	28.0
40	50	26.6	20.1	19.5	1.011	72.3
50	20	12.4	8.8	37.1	1.020	0.9
50	30	18.8	13.0	18.1	1.019	9.6
50	40	23.9	17.4	18.0	1.014	41.8
50	50	27.7	19.3	14.4	1.013	117.7

Although the heuristic cannot identify many efficient solutions and does not find a solution for each number of labels present in the efficient set, the solutions it generates turn out to be of good quality when the length objective is considered. Furthermore, the computational time needed by the heuristic remains small compared with the overall computational time.

5. Conclusion

We have introduced a branch-and-cut algorithm applicable to the solution of multiobjective integer programs. The proposed approach closely follows the standard branch-and-cut algorithm. The main difference comes from the definition of the lower and upper bounds. These are defined as sets of points in the objective spaces as opposed to single values. To efficiently compute the lower bound, one must be able to express it as the solution of a pseudo-polynomially solvable multiobjective problem. The MOB&C algorithm was applied to a new biobjective routing problem, called the multilabel traveling salesman problem. We have also proposed speed-up mechanisms based on a list of dominated parameter values and multiple variable branching. These mechanisms can easily be adapted to other problems. Experiments show that MOB&C is faster, on average, than an iterative process such as ϵ-constraint method. Furthermore, because it works on the efficient set, the method can be used heuristically by limiting the computational time. In this sense, it possesses the same advantages that make multiobjective evolutionary algorithms so popular among metaheuristics for multiobjective optimization.

Acknowledgments

This research was partly supported by the Canadian Natural Sciences and Engineering Research Council under Grant 39682-10, the International Campus on Safety and Intermodality in Transportation, the Nord-Pas-de-Calais Region, the European Community, the Regional Delegation for Research and Technology, the French Ministry of Higher Education and Research, and the National Center for Scientific Research. The authors gratefully acknowledge the support of these institutions. Thanks are also due to the referees for their valuable comments.

References

Bérubé, J.-F., M. Gendreau, J.-Y. Potvin. 2009. An exact ϵ-constraint method for bi-objective combinatorial optimization problems: Application to the travelling salesman problem with profits. *Eur. J. Oper. Res.* **194**(1) 39–50.

Cerulli, R., P. Dell'Olmo, M. Gentili, A. Raiconi. 2006. Heuristic approaches for the minimum labelling Hamiltonian cycle problem. *Electr. Notes Discrete Math.* **25**(1) 131–138.

Chankong, V., Y. Y. Haimes. 1983. *Multiobjective Decision Making: Theory and Methodology*. North-Holland, Amsterdam.

Coello Coello, C. A., G. B. Lamont, D. A. Van Veldhuizen. 2002. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer, New York.

Corberán, A., E. Fernández, M. Laguna, R. Martí. 2002. Heuristic solutions to the problem of routing school buses with multiple objectives. *J. Oper. Res. Soc.* **53**(5) 427–435.

Dantzig, G. B., D. R. Fulkerson, S. Johnson. 1954. Solution of a large-scale traveling-salesman problem. *Oper. Res.* **2**(4) 393–410.

Dean, T., M. Boddy. 1988. An analysis of time-dependent planning. *AAAI-88*, Association for the Advancement of Artificial Intelligence, Menlo Park, CA, 49–54.

Deb, K. 2001. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, Chichester, UK.

Dhaenens, C., J. Lemesre, E.-G. Talbi. 2010. K-PPM: A new exact method to solve multi-objective combinatorial optimization problems. *Eur. J. Oper. Res.* **200**(1) 45–53.

Ehrgott, M., X. Gandibleux. 2002. Multiobjective combinatorial optimization. M. Ehrgott, X. Gandibleux, eds. *Multiple Criteria Optimization: State of the Art Annotated Bibliographic Surveys*. Kluwer, Boston, 369–444.

Ehrgott, M., X. Gandibleux. 2007. Bound sets for biobjective combinatorial optimization problems. *Comput. Oper. Res.* **34**(9) 2674–2694.

Ehrgott, M., D. Tenfelde-Podehl, T. Stephan. 2006. A level set method for multiobjective combinatorial optimization: Application to the quadratic assignment problem. *Pacific J. Optim.* **2**(3) 521–544.

Feillet, D., P. Dejax, M. Gendreau. 2005. Traveling salesman problems with profits. *Transportation Sci.* **39**(2) 188–205.

Jozefowiez, N., F. Glover, M. Laguna. 2008. Multi-objective metaheuristics for the traveling salesman problem with profits. *J. Math. Model. Algorithms* **7**(2) 177–195.

Jozefowiez, N., G. Laporte, F. Semet. 2011. A branch-and-cut algorithm for the minimum labeling Hamiltonian cycle problem and two variants. *Comput. Oper. Res.* **38**(11) 1534–1542.

Jozefowiez, N., F. Semet, E.-G. Talbi. 2007. The bi-objective covering problem. *Comput. Oper. Res.* **34**(7) 1929–1942.

Labbé, M., G. Laporte, I. R. Martín, J. J. S. González. 2004. The ring star problem: Polyhedral analysis and exact algorithm. *Networks* **43**(3) 177–189.

Laumanns, M., L. Thiele, E. Zitzler. 2005. An adaptive scheme to generate the Pareto front based on the epsilon-constraint method. J. Branke, K. Deb, K. Miettinen, R. E. Steuer, eds. *Practical Approaches Multi-Objective Optim.*, Dagstuhl Seminar Proc. Internationales Begegnungs—und Forschungszentrum für Informatik (IBFI), Dagstuhl, Germany, Seminar 04461. http://drops.dagstuhl.de/opus/volltexte/2005/246.

Lemesre, J., C. Dhaenens, E.-G. Talbi. 2007. Parallel partitioning method (PPM): A new exact method to solve bi-objective problems. *Comput. Oper. Res.* **34**(8) 2450–2462.

Liefooghe, A., L. Jourdan, M. Basseur, E.-G. Talbi, E. K. Burke. 2008a. Metaheuristics for the bi-objective ring star problem. J. Van Hemert, C. Cotta, eds. *Eighth Eur. Conf. Evolutionary Comput. Combin. Optim.* (*EvoCOP 2008*). Lecture Notes in Computer Science, Vol. 4972. Springer, Berlin, 206–217.

Liefooghe, A., L. Jourdan, N. Jozefowiez, E.-G. Talbi. 2008b. On the integration of a TSP heuristic into an (EA) for the bi-objective ring star problem. M. J. Blesa et al., eds. *Internat. Workshop Hybrid Metaheuristics* (*HM 2008*). Lecture Notes in Computer Science, Vol. 5296. Springer, Berlin, 117–130.

Mavrotas, G., D. Diakoulaki. 1998. A branch-and-bound algorithm for mixed zero-one multiple objective linear programming. *Eur. J. Oper. Res.* **107**(3) 530–541.

Pacheco, J., R. Martí. 2006. Tabu search for a multi-objective routing problem. *J. Oper. Res. Soc.* **57**(1) 29–37.

Przybylski, A. 2006. Méthode en deux phases pour la résolutions exacte de problèmes d'optimisation combinatoire comportant plusieurs objectifs: Nouveaux développements et application au problème d'affectation linéaire. Ph.D. thesis, Université de Nantes, Nantes, France.

Przybylski, A., X. Gandibleux, M. Ehrgott. 2008. Two phase algorithms for the bi-objective assignment problem. *Eur. J. Oper. Res.* **185**(2) 509–533.

Przybylski, A., X. Gandibleux, M. Ehrgott. 2010a. A recursive algorithm for finding all nondominated extreme points in the outcome set of a multiobjective integer programme. *INFORMS J. Comput.* **22**(3) 371–386.

Przybylski, A., X. Gandibleux, M. Ehrgott. 2010b. A two phase method for multi-objective integer programming. *Discrete Optim.* **7**(3) 149–165.

Sourd, F., O. Spanjaard. 2008. A multiobjective branch-and-bound framework: Application to the biobjective spanning tree problem. *INFORMS J. Comput.* **20**(3) 472–484.

Ulungu, E. L., J. Teghem. 1995. The two phases method: An efficient procedure to solve bi-objective combinatorial optimization problems. *Foundation Comput. Decision Sci.* **20**(2) 149–165.

Xiong, Y., B. Golden, E. Wasil. 2007. The colorful traveling salesman problem. E. K. Baker, A. Joseph, A. Mehrotra, M. A. Trick, eds. *Extending the Horizons: Advances in Computing, Optimization, and Decision Technologies*. Springer, Boston, 115–123.

Computers & Operations Research 38 (2011) 1534–1542

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

A branch-and-cut algorithm for the minimum labeling Hamiltonian cycle problem and two variants

Nicolas Jozefowiez [a,b,*], Gilbert Laporte [c], Frédéric Semet [d]

[a] LAAS-CNRS, 7 Avenue du Colonel Roche, F-31077 Toulouse, France
[b] Université de Toulouse, UPS, INSA, INP, ISAE, LAAS, F-31077 Toulouse, France
[c] CIRRELT, HEC Montréal, 3000 chemin de la Côte-Sainte-Catherine, Montréal, Canada H3T 2A7
[d] LAGIS, École Centrale de Lille - Cité Scientifique – BP48 - 59651 Villeneuve d'Ascq Cedex, France

ARTICLE INFO

Available online 20 January 2011

Keywords:
Minimum labeling problem
Traveling salesman problem
Branch-and-cut

ABSTRACT

This paper proposes a mathematical model, valid inequalities and polyhedral results for the minimum labeling Hamiltonian cycle problem. This problem is defined on an unweighted graph in which each edge has a label. The aim is to determine a Hamiltonian cycle with the least number of labels. We also define two variants of this problem by assigning weights to the edges and by considering the tour length either as an objective or as a constraint. A branch-and-cut algorithm for the three problems is developed, and computational results are reported on randomly generated instances and on modified instances from TSPLIB.

1. Introduction

The purpose of this paper is to develop mathematical models and a branch-and-cut algorithm for the minimum labeling Hamiltonian cycle problem (MLHCP) and two of its variants. The MLHCP is defined on an undirected and unweighted graph $G=(V, E)$, where V is the vertex set, $n=|V|$, and E is the edge set. Let $\delta(e)$ be a label (or colour) associated with edge e, let C be the set of all labels, $\zeta(k)=\{e\in E|\delta(e)=k\}$, and $p=|C|$. A label $k\in C$ is said to be used if an edge $e\in\zeta(k)$ belongs to the cycle. The aim is to determine a Hamiltonian cycle with the least number of labels.

Labels can represent transporters, types of telecommunication fibers or production technologies. In Xiong et al. [1], the labels correspond to transporters who are paid a monthly fee for their services, and the goal is to design a tour using the least number of transporters. Change and Len [2], who have studied a similar problem on spanning trees, seek a solution minimizing the number of types of communication lines (optic fiber, cable, microvave, telephone line, etc.). This application extends naturally to ring networks [3]. The MLHCP has been solved heuristically by Cerulli et al. [4] and Xiong et al. [1]. The latter authors refer to the problem as the colourful traveling salesman problem.

We also consider two variants of the MLHCP by associating a cost c_e to each edge $e\in E$. In the first problem, the tour length cannot exceed a given limit. This problem will be referred to as the minimum labeling Hamiltonian cycle problem with length constraint (MLHCPLC). The second variant, the label constrained traveling salesman problem (LCTSP), minimizes the tour length while imposing an upper bound m on the number of labels. To our knowledge, these two problems have not previously been studied.

The main contribution of the paper is to propose mathematical models, valid inequalities and polyhedral results for the three problems just described. The valid inequalities are embedded within a branch-and-cut algorithm. The remainder of this paper is organized as follows. Mathematical models, valid inequalities, and polyhedral results are presented in Section 2. The branch-and-cut algorithm and heuristics are described in Section 3. Computational results and conclusions follow in Section 4 and 5, respectively.

2. Mathematical models and polyhedral analysis

The MLHCP and its variants can be formulated along the lines of the model proposed by Dantzig et al. [5] for the traveling salesman problem (TSP).

2.1. Mathematical model for the MLHCP

We define binary variables x_e equal to 1 if and only if $e\in E$ is used, and binary variables u_k equal to 1 if and only if $k\in C$ is used. The problem is then:

$$\text{minimize} \quad \sum_{k\in C} u_k \tag{1}$$

$$\sum_{e\in\omega(\{i\})} x_e = 2 \quad (i\in V) \tag{2}$$

* Corresponding author at: LAAS-CNRS, 7 Avenue du Colonel Roche, F-31077 Toulouse, France.
E-mail addresses: nicolas.jozefowiez@laas.fr (N. Jozefowiez), gilbert.laporte@cirrelt.ca (G. Laporte), frederic.semet@ec-lille.fr (F. Semet).

0305-0548/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2011.01.011

$$\sum_{e \in \omega(S)} x_e \geq 2 \quad (S \subset V, 3 \leq |S| \leq n-3) \tag{3}$$

$$x_e \leq u_{\delta(e)} \quad (e \in E) \tag{4}$$

$$x_e \in \{0,1\} \quad (e \in E) \tag{5}$$

$$u_k \in \{0,1\} \quad (k \in C), \tag{6}$$

where $\omega(S) = \{e = (i,j) \in E | i \in S, j \in V \backslash S\}$. Objective (1) minimizes the number of labels. Constraints (2) and (3) are classical degree and connectivity constraints. Constraints (4) link the variables x_e and u_k by ensuring that if an edge labeled by k is used, then k is counted in the objective function. Constraints (5) and (6) are integrality constraints.

2.2. Adaptations for MLHCPLC and LCTSP

To construct a mathematical model for the MLHCPLC, we impose the constraint

$$\sum_{e \in E} c_e x_e \leq L, \tag{7}$$

where L is the maximum allowed length of the cycle.

To model the LCTSP, we replace objective (1) with

$$\text{minimize} \quad \sum_{e \in E} c_e x_e \tag{8}$$

and we impose the constraint

$$\sum_{k \in C} u_k \leq m. \tag{9}$$

2.3. Valid inequalities

Additional valid inequalities are defined through the following propositions. First note that any valid inequality for the TSP is also valid for the MLHCP, the MLHCPLC, and the LCTSP.

Proposition 1. *The constraints*

$$u_k \leq \sum_{e \in \zeta(k)} x_e \quad (k \in C) \tag{10}$$

are satisfied by all optimal solutions of the MLHCP, the MLHCPLC, and the LCTSP.

Proof. These constraints state that if a label k is used, then at least one of the edges labeled by k must be used. □

Lemma 1. *The following relations exist between the solutions of the TSP and those of the MLHCP*:

(i) *A solution for the TSP corresponds to at least one solution for the MLCHP.*
(ii) *If constraints* (10) *are added to the model, then there exists a one to one correspondence between the solutions of the TSP and those of the MLHCP.*

Proof.

(i) Let $y \in \mathbb{R}^{n(n-1)/2}$ be a feasible TSP solution satisfying constraints (2), (3) and (5). A solution for the MLHCP is obtained by fixing $x=y$ and setting u accordingly in order to satisfy constraints (4).
(ii) Dropping the u variables from an MLHCP, solution (x, u) trivially provides a feasible TSP solution x. From a TSP solution x, a feasible MLHCP solution (x, u) is uniquely determined by the following system for u:

$$u_{\delta(e)} = 1 \quad (e \in E, x_e = 1) \tag{11}$$

$$u_k \leq \sum_{e \in \zeta(k)} x_e \quad (k \in C) \tag{12}$$

$$0 \leq u_k \leq 1. \quad \square \tag{13}$$

Proposition 2. *Let $T \subseteq E$. If the constraint*

$$\sum_{e \in T} \alpha_e x_e \leq \beta \tag{14}$$

is valid for the TSP, then for a label $k \in C$, the inequality

$$\sum_{e \in T \cap \zeta(k)} \alpha_e x_e \leq \beta u_k \tag{15}$$

is valid for the MLHCP, the MLHCPLC, and the LCTSP.

Proof. If (14) is valid for the TSP, then it is also valid for the MLHCP, the MLHCPLC, and the LCTSP. Therefore, for any label $k \in C$, the inequality

$$\sum_{e \in T \cap \zeta(k)} \alpha_e x_e \leq \beta$$

is valid for the MLHCP, the MLHCPLC, and the LCTSP. Since, the left-hand side is equal to 0 if u_k is equal to 0, then (15) is valid. □

Corollary 1. *The constraints*

$$\sum_{e \in \omega(\{i\}) \cap \zeta(k)} x_e \leq 2u_k \quad (i \in V, k \in C, 3 \leq |\omega(\{i\}) \cap \zeta(k)| \leq n-3) \tag{16}$$

are valid for the MLHCP, the MLHCPLC, and the LCTSP.

Proof. From the degree constraints for the TSP, we obtain the following valid inequalities for the TSP:

$$\sum_{e \in \omega(\{i\})} x_e \leq 2 \quad (i \in V).$$

Therefore constraints (16) are valid for the MLHCP, the MLHCPLC, and the LCTSP because of Proposition 2. □

An example of a solution violating constraints (16) but satisfying constraints (2) is depicted in Fig. 1. Note that constraint (16) is valid for $|\omega(\{i\}) \cap \zeta(k)| = 2$ but is dominated by the corresponding constraint (2).

Corollary 2. *The constraints*

$$\sum_{e \in E(S) \cap \zeta(k)} x_e \leq (|S|-1)u_k \quad (k \in C, S \subset V, 3 \leq |S| \leq n-3) \tag{17}$$

are valid for the MLHCP, the MLHCPLC, and the LCTSP.

Proof. From the subtour elimination constraints for the TSP and Proposition 2, constraints (17) are valid for the MLHCP, the MLHCPLC, and the LCTSP. □

An example of a solution violating constraints (17) but satisfying the associated subtour elimination constraints is depicted in Fig. 2.

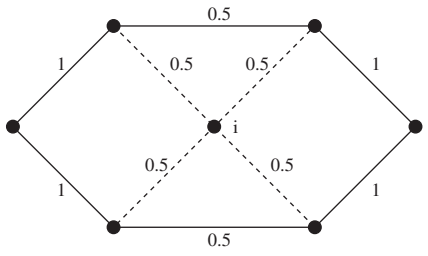


Fig. 1. A constraint (16) is violated for i and the label represented as a dashed line.

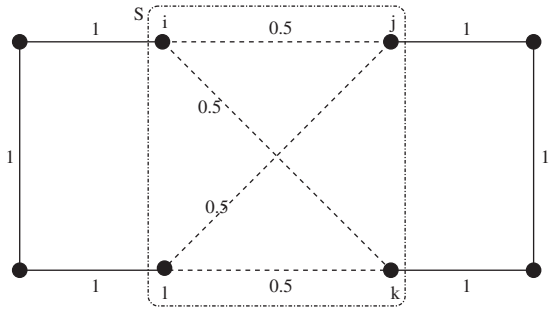


Fig. 2. A constraint (17) is violated for S and the label represented as a dashed line.

Proposition 3. *Let $T \subseteq E$. If the inequality*

$$\sum_{e \in T} \alpha_e x_e \geq \beta \tag{18}$$

is valid for the TSP, then the inequality

$$\sum_{k \in C} \min\left\{ \sum_{e \in T \cap \zeta(k)} \alpha_e, \beta \right\} u_k \geq \beta \tag{19}$$

is valid for the MLHCP, the MLHCPLC, and the LCTSP.

Proof. Since $x_e \leq u_k$, $\forall e \in \zeta(k)$, it follows that $\sum_{k \in C} \sum_{e \in T \cap \zeta(k)} \times \alpha_e u_k \geq \beta$, and (19) follows. □

Corollary 3. *The inequalities*

$$\sum_{k \in C} \gamma_k(S) u_k \geq 2 (S \subset V,\ 3 \leq |S| \leq n-3) \tag{20}$$

with

$$\gamma_k(S) = \begin{cases} |\omega(S) \cap \zeta(k)| & \text{if } |\omega(S) \cap \zeta(k)| \leq 1 \\ 2 & \text{otherwise.} \end{cases}$$

are valid for the MLHCP, the MLHCPLC, and the LCTSP.

Proof. From the connectivity constraints for the TSP and Proposition 3, constraints (20) are valid for the MLHCP, the MLHCPLC, and the LCTSP. □

2.4. Polyhedral results for the MLHCP

Let $\mathcal{C}_{\text{TSP}}$ and $\mathcal{C}_{\text{MLHCP}}$ respectively denote the convex hull of the set of feasible solutions of the TSP and of the MLCHP, that is,

$$\mathcal{C}_{\text{MLHCP}} = \text{conv}\{(x,u) \in \mathbb{R}^{n(n-1)/2} \times \mathbb{R}^p | (x,u) \text{ satisfies (2)–(6)}\}$$

and

$$\mathcal{C}_{\text{TSP}} = \text{conv}\{x \in \mathbb{R}^{n(n-1)/2} | x \text{ satisfies (2), (3) and (5)}\}.$$

In the following, $\mathbb{1}$ will denote the vector u such that $u_k = 1$ for all $k \in C$ and $\mathbb{1}^k$ the vector u such that $u_{k'} = 1$ for all $k' \in C \backslash \{k\}$ and $u_k = 0$.

Theorem 1. *If $F = \{y \in \mathcal{C}_{\text{TSP}} | \pi y = \pi_0\}$ is a proper face of $\mathcal{C}_{\text{TSP}}$ and $\pi x \leq \pi_0$ is a valid MLHCP inequality, then $\overline{F} = \{(x,u) \in \mathcal{C}_{\text{MLHCP}} | \pi x = \pi_0\}$ is a proper face of $\mathcal{C}_{\text{MLHCP}}$.*

Proof. Let y be a TSP solution satisfying $\pi y < \pi_0$ (such a solution exists because F is a proper face of $\mathcal{C}_{\text{TSP}}$). We can construct an MLHCP solution with $x = y$ by setting u accordingly. Thus $\pi x < \pi_0$ and hence $\overline{F} \neq \mathcal{C}_{\text{MLHCP}}$. Let y' be a TSP solution satisfying $\pi y' = \pi_0$. Again, such a solution exists because F is a proper face of $\mathcal{C}_{\text{TSP}}$. We can construct an MLHCP solution with $x' = y'$ by setting u' accordingly. Thus $\pi x' = \pi_0$ and hence $\overline{F} \neq \emptyset$. Therefore $\overline{F}$ is a proper face of $\mathcal{C}_{\text{MLHCP}}$. □

Theorem 2. *Let $k \in C$ and $i \in V$, if $3 \leq |\omega(\{i\}) \cap \zeta(k)| \leq n-3$, and suppose there exists $e \in \zeta(k) \backslash \omega(\{i\})$ such that for $e', e'' \in \omega(\{i\}) \backslash \zeta(k), \{e, e', e''\}$ is not a cycle, then the face $F_{16}^{ki} = \{(x,u) \in \mathcal{C}_{\text{MLHCP}} | \sum_{e \in \omega(\{i\}) \cap \zeta(k)} x_e - 2u_k = 0\}$ defined by constraint (16) for k and i is a proper face of $\mathcal{C}_{\text{MLHCP}}$.*

Proof. Let $e_1, e_2 \in \omega(\{i\}) \cap \zeta(k)$ (these two edges exist because $3 \leq |\omega(\{i\}) \cap \zeta(k)|$). By fixing $x_{e_1} = x_{e_2} = 1$, we have $u_k = 1$ and $x_e = 0$ for all $e \in \omega(\{i\}) \backslash \{e_1, e_2\}$. We can construct a solution $(x,u) \in \mathcal{C}_{\text{MLHCP}}$ with $x_{e_1} = x_{e_2} = 1$, and therefore $F_{16}^{ki} \neq \emptyset$. Let $e', e'' \in \omega(\{i\}) \backslash \zeta(k)$ (these two edges exist because $|\omega(\{i\}) \cap \zeta(k)| \leq n-3$). By fixing $x_{e'} = x_{e''} = 1$, we have $x_e = 0$ for all $e \in \omega(\{i\}) \backslash \{e', e''\}$. We can construct a solution $(x,u) \in \mathcal{C}_{\text{MLHCP}}$ with $x_{e'} = x_{e''} = 1$ by using an edge $e \in \zeta(k) \backslash \omega(\{i\})$ and therefore fixing $u_k = 1$. Thus, the left-hand side is equal to -2 and the right-hand side equal to 0. It follows that $F_{16}^{ki} \neq \mathcal{C}_{\text{MLHCP}}$. □

Theorem 3. *Let $k \in C$ and $S \subset V$, if $3 \leq |S| \leq n-3$ and $\forall i \in S, |\omega(\{i\}) \cap \zeta(k)| \geq |S|/2$, then $F_{17}^{kS} = \{(x,u) \in \mathcal{C}_{\text{MLHCP}} | \sum_{e \in E(S) \cap \zeta(k)} \times x_e - (|S|-1)u_k = 0\}$ defined by the constraint (17) for k and S is a proper face of $\mathcal{C}_{\text{MLHCP}}$.*

Proof. Since $|\omega(\{i\}) \cap \zeta(k)| \geq |S|/2$, Dirac's theorem [6] implies that there exists a Hamiltonian cycle using edges labeled by k to connect the vertices from S. By fixing $x_e = 1$ if and only if e is in the cycle, we have $\sum_{e \in E(s) \cap \zeta(k)} x_e = |S|$ and $u_k = 1$. We need to drop an edge from the cycle to connect the vertices from $V \backslash S$, meaning that $\sum_{e \in E(s) \cap \zeta(k)} x_e = |S|-1$. Thus, $\sum_{e \in E(s) \cap \zeta(k)} x_e - (|S|-1)u_k = 0$, and therefore $F_{17}^{kS} \neq \emptyset$. To show that $F_{17}^{kS} \neq \mathcal{C}_{\text{MLHCP}}$, we will consider two cases: (i) $E(S) \cap \zeta(k) = E(S)$; (ii) $E(S) \backslash \zeta(k) \neq \emptyset$. In the first case, since the subtour elimination constraints define proper faces of $\mathcal{C}_{\text{TSP}}$, there exists a solution $y \in \mathcal{C}_{\text{TSP}}$ satisfying $\sum_{e \in E(S)} y_e < |S|-1$. We can construct an MLCHP solution with $x = y$ and by setting u accordingly (in particular, $u_k = 1$). Therefore, $\sum_{e \in E(S)} x_e = \sum_{e \in E(S) \cap \zeta(k)} x_e < (|S|-1)$ and $(|S|-1)u_k = |S|-1$. In the second case, we can construct an elementary path on S using at least one edge not labeled by k and at least one edge labeled by k, leading to $\sum_{e \in E(S) \cap \zeta(k)} x_e < (|S|-1)$ and $u_k = 1$. This path can be completed into a cycle in order to visit the vertices of $V \backslash S$. This yields an MLHCP solution satisfying $\sum_{e \in E(S) \cap \zeta(k)} x_e < (|S|-1)u_k$. Thus, $F_{17}^{kS} \neq \mathcal{C}_{\text{MLHCP}}$. □

Lemma 2. *For $n \geq 3$, dim $\mathcal{C}_{\text{MLHCP}} = n(n-3)/2 + p$ under the conditions $\forall i \in V, \forall k \in C, |\omega(\{i\}) \backslash \zeta(k)| \geq n/2$.*

Proof. $\mathcal{C}_{\text{MLHCP}}$ is a set of vectors having n$(n-1)/2$ + p components. Since constraints (2) form a system of rank n, we have

$$\dim \mathcal{C}_{\text{MLHCP}} \leq n(n-1)/2 + p - n = n(n-3)/2 + p.$$

We will prove that this upper bound is tight by exhibiting a set of $n(n-3)/2+p+1$ affinely independent points in $\mathcal{C}_{\text{MLHCP}}$. Let $\{x_p,\ p = 1, \ldots, n(n-3)/2+1\}$ be a set of affinely independent TSP solutions [7]. From Lemma 1, we can construct a set $\{(x^p, \mathbb{1}), \text{p} = 1, \ldots, \text{n}(\text{n}-3)/2+1\}$ of MLHCP solutions. We can add to this set an MLHCP solution $(x^k, \mathbb{1}^k)$ for all $k \in C$. A solution $(x^k, \mathbb{1}^k)$ exists because Dirac's theorem holds if label k is forbidden. It is clear that the resulting set is composed of $n(n-3)/2+p+1$ affinely independent points. □

In the following, we define for all $k \in C$, $\mathcal{C}_{\text{MLHCP}}^k = \mathcal{C}_{\text{MLHCP}} \cap \{(x,u) \in \mathbb{R}^{n(n-1)/2} \times \mathbb{R}^p | u_k = 0\}$.

Theorem 4. *If the inequality*

$$\sum_{e \in \overline{E}} \alpha_e x_e \leq \beta \tag{21}$$

defines a facet of $\mathcal{C}_{\text{TSP}}$ and $\forall k \in C, \mathcal{C}_{\text{MLHCP}}^k \neq \emptyset$, then

$$\sum_{e \in \overline{E}} \alpha_e x_e \leq \beta - \sum_{k \in \overline{C}} \theta_k (1 - u_k) \tag{22}$$

is valid for $\mathcal{C}_{\text{MLHCP}}$ for all $\theta_k \leq \overline{\theta}_k$ with $\overline{C} = \{k \in C | \overline{E} \cap \zeta(k) \neq \emptyset\} = \{k_1, k_2, \ldots, k_q\}$ and $\overline{\theta}_{k_i} = \beta - \max(\sum_{e \in \overline{E}} \alpha_e x_e + \sum_{j=1}^{i-1} \theta_{k_j}(1-u_{k_j}) : (x,u) \in \mathcal{C}_{\text{MLHCP}}^{k_i})$. Also, if $\theta_k = \overline{\theta}_k, \forall k \in C$, then (22) is a facet of $\mathcal{C}_{\text{MLHCP}}$.

Proof. We seek for $\theta_k, k \in C$, such that

$$\sum_{e \in \overline{E}} \alpha_e x_e + \sum_{k \in C} \theta_k (1-u_k) \leq \beta.$$

Since (21) is a facet of the TSP, there exist $n(n-3)/2$ affinely independent solutions $(x^i, \mathbb{1})$ satisfying (22) as equalities for any value of $\theta_k, k \in \overline{C}$. Now, we will show that there exist vectors $(x, \mathbb{1}^k)$ $(k \in C)$ forming a set of solutions affinely independent from the vectors $(x^i, \mathbb{1})$, such that (22) holds as an equality. For each $k \in C$, two cases can occur:

- For $k \in C$ such that $\overline{E} \cap \zeta(k) \neq \emptyset$: we sequentially lift the constraint following the order of the $k_i \in \overline{C}$. Let $\overline{\theta}_{k_1} = \beta - \max(\sum_{e \in \overline{E}} : (x,u) \in C^{k_1}_{\text{MLHCP}})$. Since $C^{k_1}_{\text{MLHCP}} \neq \emptyset$, the problem has a solution (x^*, u^*), and $\overline{\theta}_{k_1}$ is defined. Therefore, for all $\theta_{k_1} \leq \overline{\theta}_{k_1}$, the inequality

$$\sum_{e \in \overline{E}} \alpha_e x_e + \theta_{k_1}(1-u_{k_1}) \leq \beta \tag{23}$$

  is valid for the MLHCP. Now suppose that $\theta_{k_1} = \overline{\theta}_{k_1}$. From (x^*, u^*), we construct the solution $(x^*, \mathbb{1}_{k_1})$. This solution is affinely independent from solutions $(x^i, \mathbb{1})$ and satisfies (22) as an equality. We construct $q-1$ other affinely independent solutions by sequentially performing liftings on $\overline{C} \backslash \{k_1\}$.
- For $k \in C$ such that $\overline{E} \cap \zeta(k) = \emptyset$: Let $k \in C \backslash \overline{C}$. We set $\theta_k = 0$. Since (21) is a facet of the TSP and $C^k_{\text{MLHCP}} \neq \emptyset$, there exists at least one solution x' such that

$$\sum_{e \in \overline{E}} \alpha_e x'_e = \beta$$

  and containing no edge from $\zeta(k)$. Therefore for the solution $(x', \mathbb{1}_k)$, (22) is satisfied as an equality. □

Facets for the MLHCP can be derived from SECs by applying Theorem 4. Note that these facets do not dominate constraints (17).

Theorem 5. *The inequalities $x_e \geq 0$ for $e = (n,m) \in E$ define facets of C_{MLHCP} for all $n \geq 4$ and if $\forall i \in V, |\omega(\{i\}) \backslash \zeta(\delta(e))| \geq n/2$ and $\forall k \in C \backslash \{\delta(e)\}$, the following conditions hold: $\forall i \in V \backslash \{n,m\}, |\omega(\{i\}) \backslash \zeta(k)| \geq n/2$, $|\omega(\{n\}) \backslash \zeta(k)| - 1 \geq n/2$, and $|\omega(\{m\}) \backslash \zeta(k)| - 1 \geq n/2$.*

Proof. Let $e \in E$. Since the inequality $x_e \geq 0$ defines a facet of C_{TSP}, there exist dim C_{TSP} affinely independent TSP solutions x^p satisfying $x^p_e = 0$ [7]. From these solutions, we can construct dim C_{TSP} affinely independent MLHCP solutions $(x^p, \mathbb{1})$ satisfying $x^p_e = 0$. If $\forall i \in V, |\omega(\{i\}) \backslash \zeta(\delta(e))| \geq n/2$, then the condition of Dirac's theorem holds and therefore there exists a solution $(x^{\delta(e)}, \mathbb{1}^{\delta(e)})$ for which $x_e = 0$. Then, $p-1$ MLHCP solutions $(x^k, \mathbb{1}^k)$ for all $k \in C \backslash \{\delta(e)\}$ with $x_e = 0$ can be constructed according to Dirac's theorem. The latter solutions, the solutions derived from the TSP and the one obtained by setting $u_k = 0$ form a set of dim C_{MLHCP} affinely independent solutions and $x_e \geq 0$ define a facet of C_{MLHCP}. □

Theorem 6. *The inequalities $x_e \leq u_{\delta(e)}$ for $e \in E$ defines facet of C_{MLHCP} for all $n \geq 4$ under the conditions: $\forall i \in V, \forall k \in C, |\omega(\{i\}) \backslash \zeta(k)| \geq n/2$.*

Proof. Let $e \in E$. Since $x_e \leq 1$ defines a facet of C_{TSP} [7], there exist dim C_{TSP} affinely independent solutions x^p satisfying $x_e = 1$. From these solutions, we can construct dim C_{TSP} affinely independent solutions $(x^p, \mathbb{1})$ satisfying $x_e = u_{\delta(e)}$. Then, $\forall k \in C \backslash \{\delta(e)\}$, we construct a solution $(x^k, \mathbb{1}^k)$ with $x_e = 1$. These solutions exist because the conditions $\forall i \in V, \forall k \in C, |\omega(\{i\}) \backslash \zeta(k)| \geq n/2$ imply that Dirac's theorem holds. To obtain dim C_{MLHCP} affinely independent solutions satisfying $x_e = u_{\delta(e)}$, we add to the previous ones the solution $(x^{\delta(e)}, \mathbb{1}^{\delta(e)})$ (again this solution exists because Dirac's theorem holds under these conditions). □

Theorem 7. *The inequalities $u_k \leq 1$ for $k \in C$ define facets of C_{MLHCP} for all $n \geq 3$ under the conditions: $\forall k \in C, \forall i \in V, |\omega(\{i\}) \backslash \zeta(k)| \geq n/2$.*

Proof. Let $k \in C$. Let (x^p) be dim $C_{\text{TSP}} + 1$ affinely independent TSP solutions. We can construct from these solutions dim $C_{\text{TSP}} + 1$ affinely independent MLHCP solutions $(x^p, \mathbb{1})$ satisfying $u_k = 1$. Then, for all $k' \in C \backslash \{k\}$, we can construct $p-1$ affinely independent solutions $(x^{k'}, \mathbb{1}^{k'})$ satisfying $u_k = 1$ because the conditions $\forall i \in V, \forall k \in C, |\omega(\{i\}) \backslash \zeta(k)| \geq n/2$ imply that Dirac's theorem holds. Therefore, there exist dim C_{MLHCP} affinely independent MLHCP solutions satisfying $u_k = 1$. □

3. Algorithms

We solve the MLHCP, the MLHCPLC, and the LCTSP by means of a branch-and-cut algorithm. The step-by-step description of the method is summarized in Algorithm 1. We now provide details on the steps of the algorithm.

Initial subproblem: The first subproblem is obtained by relaxing the connectivity constraints (3), as well as the integrality constraints on the variables. We also add the inequalities (10), (16) and (17) for all subsets of V of cardinality three for which the three edges have the same label.

Constraint generation: A search for violated constraints (3) is performed by identifying the connected components and the min cut in each component. A constraint is generated for each connected component (if there is more than one), and for each min cut of value strictly less than 2. Each subset generated in this manner is also used to check whether constraints (20) are violated or not. The aim of the min cut problem is to find in a weighted graph $G_m = (V_m, E_m)$ a subset $S_m \subset V_m$ such that the weight of the edges in the set $\omega(S_m)$ is minimal. Min cuts are separated using the algorithm proposed by Padberg and Rinaldi [8].

Branching: Branching is performed in priority on the u_k variable with the largest value of $|\zeta(k)|$. The direction of the first branch is $u_k = 1$. When all u_k variables are integer, the fractional variable x_e closest to 0.5 is selected for branching.

Algorithm 1. Branch-and-cut algorithm

Step 1 (Root of the tree)
Generate an initial upper bound *ub*.
Define a first subproblem.
Insert the subproblem in a list *L*.

Step 2 (Stopping criterion)
If $L = \emptyset$ then stop. Else choose a subproblem from *L* and remove it from *L*.

Step 3 (Subproblem solution)
Solve the subproblem to obtain the lower bound *lb*.

Step 4 (Constraint generation)
if $lb \geq ub$ **then**
  Go to Step 2.
else if the solution is integer **then**
  $ub \leftarrow lb$.
  Go to Step 2.
else
  if violated constraints are identified **then**
    Add them to the model and go to Step 3.
  else
    Go to Step 5.
  end if
end if

Step 5 (Branching)
Branch on a variable and introduce the corresponding subproblems in *L*. Go to Step 2.

Table 1
Summary of computation results for the MLHCP.

n	p	Opt	lb/Opt	#Nodes	#u	#x	#Cuts	Seconds
50	50	4	0.552	113	56	0	33	19
50	100	5	0.691	37	18	0	2	4
50	150	6	0.749	41	20	0	4	3
50	200	7	0.783	3427	1711	2	75	295
100	50	3	0.572	13	6	0	0	26
100	100	4	0.561	143	68	3	7	290
100	150	5	0.587	973	486	0	6	2036
100	200	6	0.602	6587	3289	4	87	12583
150	50	2	0.826	5	1	1	7	46
150	100	3	0.667	11	5	0	9	91
150	150	4	0.626	81	40	0	2	524
150	200	5	0.580	1379	688	1	16	8872
200	50	2	0.812	3	1	0	4	152
200	100	3	0.643	31	3	12	17	289
200	150	4	0.564	285	140	2	12	4105
200	200	4	0.617	133	59	7	40	2630

Initial upper bound: To compute an initial upper bound for the LCTSP, we use the following heuristic. We solve the same problem as the initial subproblem of the branch-and-cut algorithm, except that integrality is enforced for the variables u_k. The search is limited to 30 s as we do not necessarily need to prove optimality to compute the bound. If a feasible solution is returned, then a search is performed to identify a TSP solution on the graph G', which is the same graph as G, except for the edge costs. The cost c'_e of edge e is equal to c_e if $u_{\delta(e)}=1$, and to $n \max_{e \in E} c_e + c_e$ otherwise. This large cost is introduced to disfavour edges for which $u_{\delta(e)}=0$. If the solution uses fewer labels than the maximal allowed number of labels, it is then used as an initial upper bound. If this is not the case, or if a feasible solution has not been found for the Mixed Integer Program (MIP), the right-hand side of constraint (9) is decreased by one and the process is iterated.

A similar heuristic is used for the MLHCPLC with the following differences. In a first step, an optimal TSP solution is computed. It necessarily respects the upper bound on the tour length and therefore if the heuristic does not provide a better solution, the

Table 2
Summary of computational results for the MLHCPLC (part 1).

Inst.	p	%	Opt	lb/Opt	#Nodes	#u	#x	#Cuts	Seconds	ub	ub/Opt	Seconds
eil51	25	697	6/697*	0.766	13	6	0	0	18	6/697	1.000	18
eil51	25	494	12/494*	0.866	17	8	0	15	2	12/494	1.000	2
eil51	25	432	18/432*	0.940	3	1	0	9	1	18/432	1.000	1
eil51	50	599	12/599*	0.884	303	151	0	85	52	13/572	1.083	47
eil51	50	435	25/435*	0.955	173	82	4	56	3	27/429	1.080	2
eil51	50	426	29/426*	0.920	65	32	0	19	0	29/426	1.000	0
eil51	75	519	18/519*	0.924	21	10	0	9	3	18/519	1.000	3
eil51	75	426	35/426*	0.886	83	41	0	15	0	35/426	1.000	0
eil51	75	426	35/426*	0.886	83	41	0	15	0	35/426	1.000	0
eil51	100	477	25/477*	0.952	353	176	0	84	11	27/471	1.080	9
eil51	100	426	37/426*	0.896	161	80	0	39	1	38/426	1.027	0
eil51	100	426	37/426*	0.896	161	80	0	39	1	38/426	1.027	0
berlin52	25	11330	6/11 330*	0.848	3	1	0	4	4	6/11 330	1.000	4
berlin52	25	8516	12/8516*	0.908	51	25	0	39	3	13/8422	1.083	2
berlin52	25	7758	18/7758*	0.966	27	13	0	27	2	19/7669	1.056	2
berlin52	50	10074	12/10 074*	0.914	209	102	2	71	39	13/9756	1.083	35
berlin52	50	7817	25/7817*	0.987	3	1	0	15	0	25/7817	1.000	0
berlin52	50	7542	35/7542*	1.000	1	0	0	2	0	35/7542	1.000	0
berlin52	75	9058	18/9058*	0.955	3	1	0	10	1	18/9058	1.000	1
berlin52	75	7604	37/7604*	0.993	3	1	0	5	0	37/7604	1.000	0
berlin52	75	7542	41/7542*	1.000	1	0	0	4	0	41/7542	1.000	0
berlin52	100	8481	25/8481*	0.962	183	91	0	52	33	26/8411	1.040	32
berlin52	100	7542	42/7542*	1.000	1	0	0	4	0	42/7542	1.000	0
berlin52	100	7542	42/7542*	1.000	1	0	0	4	0	42/7542	1.000	0
eil76	25	837	6/837*	0.794	5	2	0	0	16	6/837	1.000	16
eil76	25	629	12/629*	0.874	381	184	6	162	29	13/617	1.083	17
eil76	25	565	18/565*	0.909	15	7	0	19	5	18/565	1.000	5
eil76	50	747	12/747*	0.837	1675	837	0	338	157	13/726	1.083	60
eil76	50	577	25/577*	0.919	1063	510	21	172	22	27/569	1.080	5
eil76	50	539	37/539*	0.934	175	81	6	56	2	40/538	1.081	1
eil76	75	715	18/715*	0.884	295	147	0	80	42	18/715	1.000	30
eil76	75	561	37/561*	0.908	5231	2610	5	266	120	39/560	1.054	54
eil76	75	538	49/538*	0.929	41	20	0	17	0	49/538	1.000	0
eil76	100	658	25/658*	0.920	8921	4457	3	674	413	29/614	1.160	60
eil76	100	540	50/540*	0.960	61	30	0	21	1	50/540	1.000	1
eil76	100	538	52/538*	0.971	23	11	0	13	0	53/538	1.019	0
pr76	25	184776	6/184 776*	0.832	3	1	0	0	22	6/184 776	1.000	20
pr76	25	129347	12/129 347*	0.863	439	218	1	192	35	13/124 554	1.083	17
pr76	25	111644	18/111 644*	0.933	371	124	61	95	15	19/110 583	1.056	10
pr76	50	166902	12/166 902*	0.854	2291	1145	0	441	220	13/161 063	1.083	31
pr76	50	115897	25/115 897*	0.927	12003	3973	2028	467	320	28/113 538	1.120	40
pr76	50	109241	34/109 150	0.858	206693	71659	31706	571	3600	38/108 159	1.118	12
pr76	75	154707	18/154 707*	0.885	13713	6851	5	750	1213	19/150 358	1.056	60
pr76	75	111639	42/111 585	0.822	169607	57335	27489	616	3600	44/108 159	1.048	63
pr76	75	108689	43/108 662	0.871	173786	75772	11142	600	3600	44/108 159	1.023	35
pr76	100	131725	25/131 725*	0.952	7	3	0	14	31	25/131 725	1.000	29
pr76	100	109599	49/108 159	0.762	138892	65371	4095	586	3600	49/108 159	1.000	27
pr76	100	109326	49/108 159	0.768	125702	59722	3150	630	3600	49/108 159	1.000	47

number of labels used in the optimal tour will provide an upper bound. Also, if there is a need to iterate the process because no feasible MIP solution has been found or because the tour does not respect the length constraint, then this bound is modified as follows. Let L be the current bound. If a MIP solution was not found, L is fixed to $L-\lceil(L-l^*)/2\rceil$, where l^* is the optimal TSP length. If a MIP solution was found and the length l of the resulting TSP solution exceeds L, the latter is set equal to $L-(l-L)$. The rationale is that $l-L$ is the distance exceeding the bound on the length, and therefore we reduce L by this distance. The process is iterated until a feasible solution is found or until L is smaller than l^*.

No initial upper bound is computed for the MLHCP. These heuristics are used after the first constraint generation step of the branch-and-cut algorithm and not during Step 1. This is done to take advantage of the violated cuts identified during this first constraint generation step.

4. Computational results

The algorithm just described was coded in C and was run on a computer with an Intel Core 2 Duo E6550 2.33 Ghz CPU. Linear programs were solved by CPLEX 11.1.2 and the TSPs were solved using the Concorde library (http://www.tsp.gatech.edu/concorde/index.html).

Tests for the MLHCP were conducted on randomly generated instances for $p=50$, 100, 150, and 200 and $n=50$, 100, 150, and 200 and their results are reported in Table 1. For each pair (p, n), one instance is deterministically generated by assigning the labels to the edges by means of the following formula. An edge (i, j) is given the label $\lfloor p(ijA-\lfloor ijA\rfloor)\rfloor$, where $A=(\sqrt{5}-1)/2$. Vertices are numbered from 1 to n and labels are coded as integers between 0 and $p-1$. The table gives the optimum (Opt), the ratio between the lower bound and the optimum (lb/Opt) at the root of the search tree, the number of nodes in the search tree (#Nodes), the number of branchings on a

Table 3
Summary of computational results for the MLHCPLC (part 2).

Inst.	p	%	Opt	lb/Opt	#Nodes	#u	#x	#Cuts	Seconds	ub	ub/Opt	Seconds
kroA100	25	40602	6/40 602*	0.781	1027	507	6	318	315	7/38 855	1.167	31
kroA100	25	27381	12/27 381*	0.819	579	288	1	343	126	13/27 163	1.083	61
kroA100	25	22599	18/22 599*	0.892	35	17	0	56	29	18/22 599	1.000	27
kroA100	50	36423	13/34 895	0.762	10122	5064	0	1159	3600	13/34 895	1.000	61
kroA100	50	24812	25/24 812*	0.852	12911	6450	5	503	1045	27/24 529	1.080	61
kroA100	50	21639	37/21 639*	0.923	22683	6760	4581	217	386	38/21 548	1.027	35
kroA100	75	33811	19/32 908	0.803	15073	7542	0	1082	3600	19/32 908	1.000	62
kroA100	75	23842	35/23 801*	0.923	38583	19277	14	414	2112	44/23 494	1.257	62
kroA100	75	21432	57/21 427	0.853	195572	86627	11181	302	3600	58/21 282	1.018	61
kroA100	100	30056	27/30 034	0.804	24399	12179	29	992	3600	29/28 760	1.074	91
kroA100	100	21833	59/21 794	0.808	135467	58987	8773	314	3600	62/21 282	1.051	72
kroA100	100	21282	62/21 282	0.863	137787	67469	1442	360	3600	62/21 282	1.000	5
kroB100	25	39594	6/39 594*	0.764	21	10	0	1	36	6/39 594	1.000	31
kroB100	25	27234	12/27 234*	0.827	203	100	1	176	84	13/26 837	1.083	62
kroB100	25	23397	18/23 397*	0.918	197	77	21	97	46	20/22 932	1.111	40
kroB100	50	35882	12/35 848*	0.821	1357	678	0	240	278	12/35 848	1.000	30
kroB100	50	24912	25/24 912*	0.928	3047	1513	10	259	237	28/24 570	1.120	60
kroB100	50	22361	37/22 361*	0.936	31159	13152	2427	279	710	41/22 141	1.108	40
kroB100	75	35138	18/35 053	0.766	14753	7374	8	1140	3600	19/33 274	1.056	92
kroB100	75	23815	37/23 815*	0.930	15929	7924	40	311	702	41/23 660	1.108	43
kroB100	75	22186	59/22 141	0.808	152666	65311	11050	356	3600	59/22 141	1.000	35
kroB100	100	30764	27/30 634	0.819	27412	13703	11	955	3600	29/29 698	1.074	91
kroB100	100	23095	50/23 095*	0.938	42055	20489	538	395	1221	55/22 695	1.100	45
kroB100	100	22211	67/22 141	0.820	147998	71429	2583	412	3600	67/22 141	1.000	19
eil101	25	1002	6/1002*	0.757	579	289	0	145	169	7/964	1.167	30
eil101	25	750	12/750*	0.833	77	38	0	47	37	12/750	1.000	30
eil101	25	665	18/665*	0.861	249	99	25	142	73	19/659	1.056	63
eil101	50	931	12/931*	0.796	1597	798	0	138	326	12/931	1.000	30
eil101	50	693	25/693*	0.858	7801	3828	72	706	587	29/681	1.160	91
eil101	50	638	37/638*	0.940	865	428	4	139	56	39/635	1.054	39
eil101	75	869	18/869	0.829	20378	10193	0	897	3600	19/847	1.056	61
eil101	75	670	37/670*	0.913	11649	5819	5	515	576	40/663	1.081	61
eil101	75	629	54/629*	0.957	21	10	0	23	2	55/629	1.019	1
eil101	100	803	25/803	0.838	25375	12665	27	1070	3600	28/768	1.120	92
eil101	100	643	50/643*	0.956	901	448	2	108	58	52/641	1.040	42
eil101	100	629	63/629*	0.921	139	69	0	47	4	65/629	1.032	2
lin105	25	28024	6/28 024*	0.776	1623	746	65	508	562	7/27 039	1.167	31
lin105	25	18014	12/18 014*	0.885	15	7	0	38	20	12/18 014	1.000	17
lin105	25	15277	18/15 277*	0.928	3	1	0	33	5	18/15 277	1.000	3
lin105	50	23803	12/23 803*	0.860	1569	777	7	431	380	14/22 184	1.167	62
lin105	50	16208	25/16 208*	0.935	299	147	2	123	79	27/15 825	1.080	60
lin105	50	14635	37/14 635*	0.991	3	1	0	33	2	37/14 635	1.000	1
lin105	75	22677	18/22 641	0.833	13981	6985	6	928	3600	19/22 012	1.056	61
lin105	75	15460	37/15 460*	0.956	155	77	0	83	68	38/15 455	1.027	60
lin105	75	14379	54/14 379*	0.984	1	0	0	26	0	54/14 379	1.000	0
lin105	100	20420	25/20 180*	0.900	1507	753	0	164	211	25/20 180	1.000	31
lin105	100	15021	50/15 021*	0.972	529	260	4	114	39	51/15 008	1.020	25
lin105	100	14379	68/14 379*	0.985	3	1	0	33	1	68/14 379	1.000	1

variable u_k (#u) and on a variable x_e (#x), the number of identified cuts (#Cuts), and the computing time (Seconds).

Results show that instances with a large number of vertices and a large number of labels can be solved optimally. We observe that the optimal solution value tends to be very low in comparison with the number of available labels. Also, the computation time increases with p but is not necessarily linked to the number of nodes in the search tree. The difficulty comes from solving the linear program which tends to become time consuming. This is due to the symmetry of the problem. Indeed, several Hamiltonian cycles can have the same number of labels (even a low number of labels) but it is not possible to distinguish among them. It also appears that branching is mainly performed on the u_k variables.

To generate instances for the MLHCPLC and the LCTSP, we have used instances from the TSPLIB for the distance matrix, and labels were assigned using the same formula as for the MLHCP. Runs were limited to 1 h. The number of labels considered were 25, 50, 75, and 100. For the LCTSP, the limit on the number of labels was fixed to 25%, 50%, and 75% of the total number of labels. For the MLHCPLC, the limits for the length correspond to the best solutions found for the LCTSP. For instance, if the best solution for the LCTSP found for a TSPLIB instance i with n labels and a bound of 0.25% n is l, then we generate for the MLHCPLC an instance based on the TSPLIB instance i, with n labels, and l as the bound on the length of the tour. This way, each result for the MLHCPLC is linked to a corresponding result for the LCTSP. Results for the MLHCPLC and the LCTSP are reported in Tables 2, 3 and 4, 5, respectively. These tables give the optimum or the best found solution (Opt) with the value of the second aspect of the problem (number of labels in the case of the LCTSP and length of the tour in the case of the MLHCPLC), the ratio between the lower bound at the root and the optimum (lb/Opt), the number of nodes in the search tree (#Nodes), the number of branchings on a variable u_k (#u) and on a variable x_e (#x), the number of identified cuts (#Cuts) and the computing time (Seconds). They also provide the initial upper bound (ub), the ratio between the upper bound and the optimum (ub/Opt),

Table 4
Summary of computation results for the LCTSP (part 1).

Inst.	p	%	Opt	lb/Opt	#Nodes	#u	#x	#Cuts	Seconds	ub	ub/Opt	Seconds
eil51	25	25	697/6*	0.889	795	395	2	164	28	697/6	1.000	4
eil51	25	50	494/12*	0.955	353	169	7	99	4	494/12	1.000	1
eil51	25	75	432/18*	0.990	33	14	2	18	0	432/18	1.000	0
eil51	50	25	599/12*	0.944	2957	1412	66	458	126	672/10	1.122	57
eil51	50	50	435/25*	0.991	109	51	3	30	0	438/24	1.007	0
eil51	50	75	426/29*	0.992	483	217	24	39	0	428/28	1.005	0
eil51	75	25	519/18*	0.966	317	158	0	89	5	519/18	1.000	1
eil51	75	50	426/35*	0.992	383	191	0	48	1	429/32	1.007	0
eil51	75	75	426/35*	0.992	343	170	1	40	0	429/32	1.007	0
eil51	100	25	477/25*	0.979	1039	516	3	141	8	486/24	1.019	0
eil51	100	50	426/37*	0.992	103	51	0	17	0	426/37	1.000	0
eil51	100	75	426/37*	0.992	103	51	0	17	0	426/37	1.000	0
berlin52	25	25	11 330/6*	0.933	151	70	5	46	6	11 330/6	1.000	2
berlin52	25	50	8516/12*	0.975	139	65	4	88	2	8600/12	1.010	0
berlin52	25	75	7758/18*	0.994	53	25	1	37	0	7775/18	1.002	0
berlin52	50	25	10 074/12*	0.955	719	356	3	148	15	10 211/12	1.014	1
berlin52	50	50	7817/25*	0.998	11	5	0	22	0	7817/25	1.000	0
berlin52	50	75	7542/35*	1.000	1	0	0	5	0	0/0	0.000	0
berlin52	75	25	9058/18*	0.985	43	21	0	33	1	9058/18	1.000	1
berlin52	75	50	7604/37*	0.999	5	2	0	6	0	7604/37	1.000	0
berlin52	75	75	7542/41*	1.000	1	0	0	5	0	0/0	0.000	0
berlin52	100	25	8481/25*	0.989	1729	857	7	145	14	8715/25	1.028	0
berlin52	100	50	7542/42*	1.000	1	0	0	5	0	0/0	0.000	0
berlin52	100	75	7542/42*	1.000	1	0	0	5	0	0/0	0.000	0
eil76	25	25	837/6*	0.914	393	190	6	112	53	837/6	1.000	14
eil76	25	50	629/12*	0.967	2607	1057	246	651	131	674/10	1.072	26
eil76	25	75	565/18*	0.982	201	82	18	68	3	565/18	1.000	1
eil76	50	25	747/12*	0.935	11797	5865	33	1086	1136	768/12	1.028	30
eil76	50	50	577/25*	0.981	1351	657	18	207	27	586/25	1.016	2
eil76	50	75	539/37*	0.997	37	18	0	16	0	541/36	1.004	0
eil76	75	25	715/18*	0.952	12949	6471	3	747	847	715/18	1.000	28
eil76	75	50	561/37*	0.980	16311	8095	60	457	248	569/37	1.014	4
eil76	75	75	538/50*	0.998	3	1	0	4	0	538/50	1.000	0
eil76	100	25	658/25*	0.966	7495	3746	1	594	316	658/25	1.000	30
eil76	100	50	540/50*	0.997	413	206	0	54	4	543/50	1.006	1
eil76	100	75	538/53*	0.998	3	1	0	6	0	538/53	1.000	0
pr76	25	25	184 776/6*	0.911	225	110	2	73	37	184 776/6	1.000	9
pr76	25	50	129 347/12*	0.947	1341	567	103	388	62	133 331/12	1.031	6
pr76	25	75	111 644/18*	0.985	3491	344	1401	144	32	111 954/18	1.003	1
pr76	50	25	166 902/12*	0.913	6317	3156	2	757	719	168 410/12	1.009	30
pr76	50	50	115 897/25*	0.979	11089	3595	1949	296	174	116 896/24	1.009	2
pr76	50	75	109 241/37	0.966	335784	68141	99758	516	3600	109 410/36	1.002	1
pr76	75	25	154 707/18*	0.938	24183	12075	16	962	2349	154 910/18	1.001	30
pr76	75	50	111 639/37*	0.977	70899	27801	7648	487	1169	112 530/37	1.008	4
pr76	75	75	108 689/43	0.967	331298	103028	62628	464	3600	108 689/43	1.000	1
pr76	100	25	131 725/25*	0.976	4003	1988	13	373	183	135 599/24	1.029	22
pr76	100	50	109 599/49	0.959	183936	80261	11716	545	3600	109 604/45	1.000	6
pr76	100	75	109 326/52	0.962	251161	110314	15276	436	3600	109 417/50	1.001	2

and the time needed to compute it. An asterisk indicates that the solution is proven to be optimal.

As a first remark, if the algorithm is not always capable of proving the optimality of the solution, it can do so for a majority of instances: this happened in 78 instances out of 96 for the MLHCPLC, and in 75 instances out of 96 for the LCTSP. It should be noted that for the instances eil76 and pr76, which have the same number of vertices and therefore the same distribution of labels, pr76 seems to be considerably harder. This tends to indicate that computational difficulty is not only linked to the size of the instance or to the distribution of the labels. This can also be observed on the larger instances, with instances kroA100 and kroB100 being more difficult to solve than eil101 and lin105.

If we compare the performance of the branch-and-cut algorithm on the two MLHCP variants, it appears that the MLHCPLC is easier to solve than the LCTSP. First, optimality is proved on three more instances for the MLHCPLC. Of the 60 instances where both problems were solved optimally, if we compare related runs, the branch-and-cut algorithm is faster on the MLHCPLC in 36 cases. Moreover, on most instances, the difference is more important when it is faster for the MLHCPLC.

The heuristics are rather efficient for both problems. Their computation times remain reasonable with respect to the overall computing time of the algorithm, especially for the LCTSP. To illustrate the efficiency of the heuristic, for the 75 LCTSP instances that were solved to optimality, the initial upper bound was less than 1% of higher than the optimal solution value for 50 instances, between 1% and 2% for 11 instances, and between 2% and 5% for 9 instances. Only two instances have a solution value more than 5% away from the optimum.

Concerning the statistics of the search tree, it appears that the computation effort is mainly expended on the u_k variables. This can be explained by the fact that forbidding a label has major consequences on the solution structure. Moreover, connectivity cuts are efficient for the TSP aspect of the TSPLIB instances we solved. However, it should be noted that the presence of the labels makes the problem much more difficult to solve.

Table 5
Summary of computational results for the LCTSP (part 2).

Inst.	p	%	Opt	lb/Opt	#Nodes	#u	#x	#Cuts	Seconds	ub	ub/Opt	Seconds
kroA100	25	25	40 602/6*	0.879	3217	1477	131	796	1117	41 444/6	1.021	32
kroA100	25	50	27 381/12*	0.913	3951	1970	5	751	633	27 742/12	1.013	21
kroA100	25	75	22 599/18*	0.970	3873	378	1558	178	57	22 880/18	1.012	4
kroA100	50	25	36 423/12	0.901	11623	5815	0	1139	3600	36 423/12	1.000	31
kroA100	50	50	24 812/25	0.945	43106	21246	315	664	3600	25 538/25	1.029	31
kroA100	50	75	21 639/37*	0.984	72289	21890	14254	221	909	21 763/37	1.006	1
kroA100	75	25	33 811/18	0.918	15629	7820	0	1104	3600	33 811/18	1.000	30
kroA100	75	50	23 842/37	0.958	69588	34701	100	485	3600	24 709/37	1.036	31
kroA100	75	75	21 432/56	0.979	268668	91038	43306	229	3600	21 472/54	1.002	0
kroA100	100	25	30 056/25	0.933	20689	10350	0	957	3600	30 056/25	1.000	31
kroA100	100	50	21 833/50*	0.989	79747	38602	1271	290	2100	22 227/47	1.018	20
kroA100	100	75	21 282/62*	0.984	206817	78684	24724	212	1860	21 401/58	1.006	1
kroB100	25	25	39 594/6*	0.879	1687	840	3	502	522	39 594/6	1.000	31
kroB100	25	50	27 234/12*	0.939	2381	966	224	497	292	27 927/12	1.025	17
kroB100	25	75	23 397/18*	0.979	857	326	102	148	32	23 848/18	1.019	4
kroB100	50	25	35 882/12	0.908	12028	5986	31	1246	3600	36 061/12	1.005	31
kroB100	50	50	24 912/25*	0.976	14701	7255	95	447	1192	25 761/24	1.034	48
kroB100	50	75	22 361/37*	0.988	23503	6769	4982	207	379	22 756/34	1.018	15
kroB100	75	25	35 138/18	0.874	14302	7158	0	1334	3600	35 138/18	1.000	31
kroB100	75	50	23 815/37*	0.983	33189	16355	239	302	1388	24 341/37	1.022	8
kroB100	75	75	22 186/56*	0.986	185555	59793	32984	205	2638	22 375/51	1.009	4
kroB100	100	25	30 764/25	0.943	25196	12600	5	778	3600	30 764/25	1.000	31
kroB100	100	50	23 095/50	0.982	126400	61772	1436	344	3600	23 804/48	1.031	16
kroB100	100	75	22 211/68	0.983	256317	98252	29918	236	3600	22 214/65	1.000	1
eil101	25	25	1002/6*	0.895	1311	649	6	260	408	1008/6	1.006	31
eil101	25	50	750/12*	0.946	1547	770	3	500	230	750/12	1.000	30
eil101	25	75	665/18*	0.971	1337	516	152	326	62	667/18	1.003	6
eil101	50	25	931/12	0.899	13084	6503	43	1158	3600	937/12	1.006	31
eil101	50	50	693/25*	0.968	23581	11759	31	1104	2174	697/25	1.006	30
eil101	50	75	638/37*	0.994	431	214	1	88	12	639/37	1.002	2
eil101	75	25	869/18	0.929	18395	9191	11	1031	3600	876/18	1.008	30
eil101	75	50	670/37	0.983	50128	24949	118	1137	3600	695/37	1.037	30
eil101	75	75	629/56*	0.998	29	13	1	23	0	630/55	1.002	0
eil101	100	25	803/25	0.936	27371	13661	31	897	3600	806/25	1.004	30
eil101	100	50	643/50*	0.994	1059	528	1	125	30	646/49	1.005	11
eil101	100	75	629/66*	0.998	12459	5598	631	208	227	640/54	1.017	0
lin105	25	25	28 024/6*	0.869	1547	714	59	493	533	28 029/6	1.000	31
lin105	25	50	18 014/12*	0.952	223	111	0	174	48	18 014/12	1.000	13
lin105	25	75	15 277/18*	0.986	59	29	0	57	6	15 277/18	1.000	3
lin105	50	25	23 803/12*	0.921	2153	1071	5	470	558	23 803/12	1.000	31
lin105	50	50	16 208/25*	0.980	1885	902	40	228	136	16 523/24	1.019	27
lin105	50	75	14 635/37*	0.999	7	3	0	37	1	14 635/37	1.000	1
lin105	75	25	22 677/18	0.903	14424	7157	61	1137	3600	23 601/18	1.041	31
lin105	75	50	15 460/37*	0.990	2193	1077	19	145	99	15 876/35	1.027	24
lin105	75	75	14 379/54*	0.999	3	1	0	36	0	14 379/54	1.000	0
lin105	100	25	20 420/25	0.946	23169	11583	5	772	3600	20 568/25	1.007	31
lin105	100	50	15 021/50*	0.994	10971	5440	45	135	248	15 146/49	1.008	8
lin105	100	75	14 379/68*	0.999	3	1	0	33	0	14 379/68	1.000	0

5. Conclusion

We have proposed mathematical models, valid inequalities, and a branch-and-cut for the MLHCP and two of its variants where tour length is considered either as a constraint or as an objective. Computational experiments were conducted for the two variants on newly generated instances for the MLHCP and on modified TSPLIB instances. Results show that the branch-and-cut algorithm is capable of solving a majority of instances. The initial heuristic yields good results within a reasonable computing time when compared to the overall computational time.

Acknowledgements

This research was partly supported by the Canadian Natural Sciences and Engineering Research Council under grant 39682-10, the International Campus on Safety and Intermodality in Transportation, the Nord-Pas-de-Calais Region, the European Community, the Regional Delegation for Research and Technology, the French Ministry of Higher Education and Research, and the National Center for Scientific Research. The authors gratefully acknowledge the support of these institutions. Thanks are also due to the referees for their valuable comments.

References

[1] Xiong Y, Golden BL, Wasil EA. The colorful traveling salesman problem. In: Baker E, Joseph A, Mehrotra A, Trick M, editors. Extending the horizons: advances in computing, optimization, and decision techniques. Boston: Springer; 2007. p. 115–23.
[2] Chang R-S, Leu S-J. The minimum labeling spanning trees. Information Processing Letters 1997;63:277–82.
[3] Gendreau M, Labbé M, Laporte G. Efficient heuristics for the design of ring networks. Telecommunication Systems 1995;4:177–88.
[4] Cerulli R, Dell'Olmo P, Gentili M, Raiconi A. Heuristic approaches for the minimum labelling Hamiltonian cycle problem. Electronic Notes on Discrete Mathematics 2006;25:131–8.
[5] Dantzig GB, Fulkerson DR, Johnson SM. Solution of a large-scale traveling-salesman problem. Operations Research 1954;2:393–410.
[6] Dirac GA. Some theorems on abstract graphs. Proceedings of the London Mathematical Society 1952;2:69–81.
[7] Grötschel M, Padberg MW. On the symmetric travelling salesman problem I: inequalities. Mathematical Programming 1979;16:265–80.
[8] Padberg M, Rinaldi G. An efficient algorithm for the minimum capacity cut problem. Mathematical Programming 1990;47:19–36.

Journal of the Operational Research Society (2013) 64, 384–395

www.palgrave-journals.com/jors/

A heuristic approach based on shortest path problems for integrated flight, aircraft, and passenger rescheduling under disruptions

N Jozefowiez[1,2], C Mancel[3*] and F Mora-Camino[3]

[1]LAAS-CNRS, Toulouse, France; [2]Université de Toulouse, INSA, Toulouse, France; and [3]École Nationale de l'Aviation Civile, Toulouse, France

In this paper, we present a heuristic method to solve an airline disruption management problem arising from the ROADEF 2009 challenge. Disruptions perturb an initial flight plan such that some passengers cannot start or conclude their planned trip. The developed algorithm considers passengers and aircraft with the same priority by reassigning passengers and by creating a limited number of flights. The aim is to minimize the cost induced for the airline by the recovery from the disruptions. The algorithm is tested on real-life-based data, as well as on large-scale instances and ranks among the best methods proposed to the challenge in terms of quality, while being efficient in terms of computation time.

Journal of the Operational Research Society (2013) **64,** 384–395. doi:10.1057/jors.2012.20
Published online 16 May 2012

Keywords: air transport; planning; networks and graphs

1. Introduction

Airlines operate their fleet according to flight schedules, aircraft rotations, and crew rotations. Nevertheless several kinds of disruptions happen quite frequently that prevent the airline from executing the expected schedule. These disruptions may cause considerable costs for the companies (extra operations, extra catering, lodging if necessary, ticket refund, and financial compensation in case of cancellation or long delay).

Airlines are thus more and more interested in getting efficient systems allowing to return to normal operations after disruptions in a short time and with a minimum induced cost. Since the mid-1980s, several studies have been devoted to airline schedule recovery (see for instance the surveys by Filar *et al,* 2001; Ball *et al,* 2007; and Clausen *et al,* 2010). In particular, Clausen *et al* (2010) provide detailed descriptions and comparisons of the different approaches. It appears that the airline schedule recovery problem is usually decomposed according to the natural hierarchy of resources: the aircraft recovery problem (associated with flight rescheduling), the crew recovery problem, and the passenger itinerary recovery problem. This decomposition can lead to sub-optimal solutions particularly considering financial costs due to the delayed passengers.

*Correspondence: C Mancel, Ecole Nationale de l'Aviation Civile, Laboratoire MAIAA, 7 avenue Edouard Belin, BP 54005, 31055 Toulouse, France.
E-mail: catherine.mancel@enac.fr

Most of the proposed methods for the aircraft recovery problem are based on network models where nodes are associated to the flights. Teodorovic and Guberinic (1984) proposed the first branch-and-bound algorithm. It was able to solve only small-size problems. Since 1990, several heuristic methods to solve large-scale problems (Thengvall *et al*, 2001) have been proposed, ranging from greedy algorithms (Argüello *et al*, 1997; Stojkovic *et al*, 1998) to column generation-based methods (Clarke *et al*, 1997; Eggenberg *et al*, 2007).

The crew recovery problem is mainly modelled as a set covering problem and is often solved by means of branch-and-bound-based methods (Wei *et al*, 1997; Lettovsky *et al*, 2000; Medard and Sawhney, 2007).

The passenger itinerary recovery problem is mainly modelled as a multi-commodity flow network problem (Barnhart *et al*, 2002; Clarke, 2005). Relevant works addressing this problem consider it as part of an integrated disruption management problem. For instance, Lettovsky (1997) presented the first fully integrated approach for airline disruption management, although only parts of it were implemented. In his framework, an aircraft recovery model, a crew recovery model, and a passenger flow model are simultaneously considered. The proposed solution algorithms are based on Bender's decomposition.

Bratu and Barhnart (2006) proposed two models that solve the integrated aircraft and crew recovery problem while considering the impact on passenger delays in the objective function. The models have been solved using

OPL Studio in a simulation framework using data from domestic operations of a major US airline.

In this paper, the problem is the so-called disruption management problem for commercial aviation as described by Palpant *et al* (2009) in the context of the ROADEF 2009 challenge. In this context, we consider the problem of rescheduling aircraft and passengers under disruptions. Several algorithms have been proposed in the competition, see Artigues *et al* (forthcoming) for an overview of these solution approaches. Methods used in the challenge include a large neighbourhood search heuristic (Bisaillon *et al*, 2009) and a mathematical programming approach using statistical analysis (Acuna Agost *et al*, 2009).

The main contribution of this paper is the proposition of a fast and efficient heuristic method for the problem. The problem is stated in Section 2. Section 3 describes the heuristic method. Computational tests are reported in Section 4. Finally, conclusions are drawn and future research directions are discussed in Section 5.

2. Problem description

Here, we consider the integrated problem of aircraft rotation and passenger itinerary recovery as defined in Palpant *et al* (2009). All the parameters introduced hereafter are summarized in the appendix.

2.1. Planning horizon and recovery time window (*RTW*)

The data are defined over a planning horizon. However, the disruptions occur and modifications can only be made during a part of the planning horizon called the *RTW*. The start (respectively, the end) of the *RTW* is denoted as RTW_s (respectively, RTW_e). The *RTW* is divided into smaller time windows equal to 1 h, denoted $h \subseteq RTW$.

2.2. Airport

The airports form a Set A. For each airport $a \in A$ and for each time window $h \subseteq RTW$, the value c_a^{lh} is the maximum number of landings that can occur during the time window h at airport a and c_a^{th} the maximum number of takeoffs. The capacities are hard constraints. For each pair $a_1, a_2 \in A$, $d_{a_1 a_2}$ is the distance between a_1 and a_2.

2.3. Aircraft

The aircraft fleet forms a set P. An aircraft $p \in P$ has a maximum capacity in terms of seats $c_p^{\max}$. In the ROADEF 2009 challenge, three capacities, which correspond to the number of seats in the first class, the business class, and the economic class, respectively, were defined. However, to avoid complications in the explanations, only one class can be considered without loss of generality.

An aircraft $p \in P$ performs a rotation, which is a sequence σ_p of flight legs (or legs) starting from an origin airport O_p. The ith leg in the rotation is $\sigma_p(i)$ and $|\sigma_p|$ is the number of legs in the rotation. The leg $\sigma_p(i)$ is defined by an origin airport $\sigma_p^o(i)$, a departure time $\sigma_p^d(i)$, a destination airport $\sigma_p^f(i)$, an arrival time $\sigma_p^a(i)$, and a remaining seat capacity $\sigma_p^c(i)$. An aircraft P cannot operate flight legs that are longer than its maximum range $r_p^{\max}$. To be correct, σ_p must respect the following constraints:

1. The rotation σ_p must start at O_p, that is, $\sigma_p^o(1) = O_p$.
2. An aircraft p must respect a turnaround duration tr between two consecutive legs, that is, $\forall i \in [1, |\sigma_p|]$, $\sigma_p^a(i) + tr \leq \sigma_p^d(i+1)$.
3. The rotation must be *connected*, that is, $\forall i \in [1, |\sigma_p|], \sigma_p^f(i) = \sigma_p^o(i+1)$.

Alterations can be made only on the part of the rotation σ_p of an aircraft p taking place during the *RTW*. To take this into account in our algorithm seamlessly, we need to redefine O_p and to define t_p^r the earliest possible takeoff time for each $p \in P$. If the rotation σ_p of an aircraft p is empty or starts after RTW_s, then O_p is unchanged and t_p^r is equal to RTW_s. Otherwise, let k be the index of the last leg in σ_p taking off before the start of the *RTW* (ie, $\nexists k' \in [k+1, |\sigma_p|], \sigma_p^d(k) < \sigma_p^d(k') < RTW_s$). Then, O_p is equal to $\sigma_p^f(k)$ and t_p^r to $\max(RTW_s, \sigma_p^a(k) + tr)$. In the remaining part of the paper, σ_p will refer to the sub-rotation $\sigma_p(k+1, |\sigma_p|)$, which may be empty if the initial rotation is over before the beginning of the *RTW*.

For a subset $P_m \subseteq P$, each aircraft $P \in P_m$ must undergo a maintenance during the *RTW*. For each aircraft, the maintenance is defined by a maintenance airport, a duration, and a maximum range limiting the flying time allowed before the maintenance. Checking this constraint is straightforward and it will not be discussed in the paper. A maintenance can be treated seamlessly as a leg in σ_p.

We set $\sum = \{\sigma_p | p \in P\}$.

2.4. Groups of passengers

The groups of passengers form a set G. A group $g \in G$ is defined by a size s_g, an origin airport O_g, a ticket price p_g, and a status w_g, which indicates if the group is on an *inbound* or an *outbound* trip. A group $g \in G$ follows an itinerary γ_g, which is a sequence of legs not necessarily belonging to the same rotation. The ith leg in the itinerary is $\gamma_g(i)$ and $|\gamma_g|$ is the number of legs in the itinerary. The leg $\gamma_p(i)$ is defined by an origin airport $\gamma_p^o(i)$, a departure time $\gamma_p^d(i)$, a destination airport $\gamma_p^f(i)$, and an arrival time $\gamma_p^a(i)$. To be correct, γ_g must respect the following constraints:

1. The itinerary γ_p must start at O_g, that is, $\gamma_g^o(1) = O_g$.
2. A group g must respect a connection delay cd between two consecutive legs, that is, $\forall i \in [1, |\gamma_g|[, \gamma_g^a(i) + cd \leqslant \gamma_g^d(i+1)$.

3. The itinerary must be *connected*, that is, $\forall i \in [1, |\gamma_g|[$, $\gamma_g^f(i) = \gamma_g^o(i+1)$.

As with the aircraft, the fact that only the part of an itinerary γ_g taking place inside the *RTW* can be modified leads us to redefine O_g if the itinerary starts before the *RTW*. For a group g such that $\gamma_g^d(1) < RTW_s$, let k be the index of the last leg in γ_g happening before the start of the *RTW* (ie, $\nexists k' \in [k+1, |\gamma_g|], \gamma_g^d(k) < \gamma_g^d(k') < RTW_s$). Then, O_g is set to $\gamma_g^f(k)$. In the remaining part of the paper, γ_g will refer to the sub-itinerary $\gamma_g(k+1, |\gamma_g|)$.

For each group g, we also associate an earliest possible time of departure $t_g^r = \gamma_g^d(1)$, a destination airport $D_g = \gamma_g^f(|\gamma_g|)$, and a latest possible time of arrival $t_g^d = \gamma_g^a(|\gamma_g|) + ml$ with ml a constant representing the maximum allowed lateness.

We set $\Gamma = \{\gamma_g | g \in G\}$.

2.5. Sets of disruptions

The sets of disruptions are the following:

- set of flight delays $\mathcal{D}$: A delay $d \in \mathcal{D}$ is defined by a triplet $(p, i, t) \in P \times \mathbb{N}^+ \times \mathbb{N}^+$ with p the affected aircraft, i the index of the affected leg in σ_p, and t the delay in minutes;
- set of flight cancellations $\mathcal{C}$: A cancellation $c \in \mathcal{C}$ is defined by a couple $(p, i) \in P \times \mathbb{N}^+$ with p the affected aircraft and i the index of the affected leg in σ_p;
- set of aircraft breakdowns $\mathcal{B}$: A breakdown $b \in \mathcal{B}$ is defined by a triplet $(p, s, e) \in P \times \mathbb{N}^+ \times \mathbb{N}^+$ with p the affected aircraft, s the start time of the breakdown, and e the end time of the breakdown;
- set of airport capacity reductions $\mathcal{R}$: A reduction $r \in \mathcal{R}$ is defined by a quadruplet $(a, h, z, c) \in A \times RTW \times \{t, l\} \times \mathbb{N}^+$ with a the affected airport, h the time window during which the reduction happens, z the affected activity (takeoff t or landing l), and c the new capacity.

2.6. The problem

Given an initial plan $S_0 = (A, P, \sum_0, G_0, \Gamma_0)$ and sets of disruptions $(\mathcal{D}, \mathcal{C}, \mathcal{B}, \mathcal{R})$ the problem consists in providing an alternate feasible plan $S_f = (A, P, \sum_f, G_f, \Gamma_f)$ by modifying aircraft rotations and passenger itineraries during the *RTW*.

The precise objective function used for the ROADEF 2009 challenge is too complex to be completely expressed here (see, Artigues *et al*, forthcoming, for a detailed description). It is also not trivial to compute. As a consequence, in the heuristic method, which is described in the next section, it is not explicitly computed, but the following *soft constraints*, that contribute to a penalty in the objective function if they are not fulfilled, are considered:

- As much as possible, passengers should not be delayed or cancelled.
- As much as possible, the maximum delay for passengers at their destination should not exceed 18 h for domestic and continental flights, and 36 h for intercontinental flights.
- As much as possible, passengers should not be downgraded to a lower cabin class.
- As much as possible, flights should not be delayed or cancelled.
- As much as possible, by the end of the *RTW* each aircraft should be at its initially planned position.

3. The new connections and flights (NCF) heuristic method

Our heuristic, called NCF heuristic method, works in three phases. During the first phase (Section 3.1), the disruptions are integrated into the initial plan $S_0 = (A, P, \sum_0, G_0, \Gamma_0)$. They are treated in a straightforward fashion in order to return as fast as possible to a new feasible plan $S_1 = (A, P, \sum_1, G_0, \Gamma_1)$. During this phase, legs of some aircraft rotations may be removed to respect rotation connectivity or airport capacities. If a leg σ is removed, the itineraries of the groups $g \in G_0$ such that $\sigma \in \gamma_g$ are cancelled. An itinerary can also be cancelled if a change in the departure and arrival times of its legs violates the connection delay. Cancelling the itinerary of a group g means that γ_g is set to $\emptyset$. If g had in fact started its trip before the beginning of the *RTW*, we still consider its itinerary to be empty but its status w_g is set to *truncated*. At the end of this phase, we build the set $G_p = \{g \in G_0 | \gamma_g = \emptyset\}$. The goal of the following phases is to find itineraries for the groups $g \in G_p$.

The goal of the second phase (Section 3.2) is to reassign to the existing set of rotations $\sum_1$ as many passenger groups $g \in G_p$ as possible. This phase produces a new plan $S_2 = (A, P, \sum_1, G_1, \Gamma_2)$. During this phase, G_0 may be modified as some groups may be split in order to respect the aircraft seat capacities.

If G_p is not empty at the end of the second phase, we try to extend the aircraft rotations to build itineraries for the groups $g \in G_p$ in the third phase (Section 3.3). This step produces a plan $S_3 = (A, P, \sum_2, G_2, \Gamma_3)$, which is returned as the solution of the heuristic algorithm.

3.1. Phase 1: integration of the disruptions

Starting from the initial plan, the set of disruptions are considered sequentially. This phase is composed of the following steps: (i) cancelled flights; (ii) delayed flights; (iii) aircraft breakdowns; (iv) airport capacity drops; (v) airport capacity overflow repair; (vi) connectivity repair.

Cancelled flights: For each $c = (p, i) \in \mathcal{C}$, we simply remove $\sigma_p(i)$ from σ_p. The itineraries of the groups $g \in G_0 \setminus G_p$ such that $\sigma_p(i) \in \gamma_g$ are cancelled.

Algorithm 1 Delay propagation

$\sigma_p^d(i) \leftarrow \sigma_p^d(t) + d$
$\sigma_p^a(i) \leftarrow \sigma_p^a(i) + d$
while $i < |\sigma_p|$ and $\sigma_p^a(i) + tr > \sigma_p^d(i+1)$ **do**
  $d \leftarrow \sigma_p^a(i) + tr - \sigma_p^d(i+1)$
  $i \leftarrow i+1$
  $\sigma_p^d(i) \leftarrow \sigma_p^d(i) + d$
  $\sigma_p^a(i) \leftarrow \sigma_p^a(i) + d$
end while

Delayed flights: For each $d=(p,i,t) \in D$, Algorithm 1 is used to propagate the delay on $\sigma_p(i)$ to the following legs in σ_p. If a leg σ is modified by Algorithm 1, we check if the itineraries of the groups $g \in G_0 \setminus G_p$ such that $\sigma \in \gamma_g$ still respect the connection delay and cancel them if necessary.

Aircraft breakdowns: For each $b=(p,s,e) \in B$, the first step is to cancel the legs in σ_p, which overlap with the breakdown. The second step consists in inserting the breakdown in σ_p as a fictitious leg. Let i be the index of the leg $\sigma_p(i)$, such that $\nexists j, \sigma_p^a(i) < \sigma_p^a(j) < s$. The breakdown is then considered as a new leg $\sigma_p(i+1)$ with $\sigma_p^o(i+1) = \sigma_p^f(i)$, $\sigma_p^a(i+1) = s$, $\sigma_p^f(i+1) = \sigma_p^a(i)$, $\sigma_p^d(i+1) = e$. In the algorithm, the breakdown is considered as a standard leg with the exceptions that its origin airport and its arrival airport are dynamically adjusted to reflect the airport reached by the leg directly preceding the breakdown and that no turnaround delay is required. In the remaining part of the paper, this special case will not be treated in order not to obfuscate the explanations but these points are considered in the implementation.

Airport capacity reductions: For each $r = (a, h, t, c) \in \mathcal{R}$, we set c_a^{th} to c.

Airport capacity overflow repair: The following procedure is used to return to a solution respecting these capacities. The airports are sequentially considered. For each airport $a \in A$, we consider the time windows $h \in RTW$ chronologically. For a time window h, if the number of landings is $> c_a^{lh}$, a leg of an aircraft landing at a during h must be cancelled. To choose the leg to cancel, we consider the following rules. We cancel the leg of an aircraft $p \in P/P_m$ that lands at a during h such that there is a delayed flight $d \in \mathcal{D}$ operated by p. If there is no such aircraft, we select an aircraft $p \in P_m$ that has suffered from a delay. If there is still no such aircraft, we select a non-delayed aircraft landing at a during h. This is iterated until there is no more landing capacity overflow. The same process is used if there is a takeoff capacity overflow.

Connectivity repair: The previous steps can lead to rotations that do not respect the connectivity constraints. Algorithm 2 is used to repair the connectivity of the rotation of an aircraft $p \in P$.

Algorithm 2 Connectivity repair algorithm

— First phase
if $\sigma_p^o \neq O_p$ **then**
  while $|\sigma_p| \neq 0$ and $\sigma_p^o(1) \neq O_p$ **do**
    $\sigma \leftarrow$ search_sub-rotation$(p, O_p, t_p^r, \sigma_p^o(1), \sigma_p^d(1) - tr)$ (Algorithm 3) //σ is a sequence of legs
    if $|\sigma| \neq 0$ **then**
      $\sigma_p \leftarrow \sigma.\sigma_p$ //"." operator is the concatenation of two sequences of legs
      $i \leftarrow |\sigma| + 1$
    else
      Cancel $\sigma_p(1)$
    end if
  end while
else
  $i \leftarrow 1$
end if
— Second phase
while $|\sigma_p| \neq 0$ and $i < |\sigma_p|$ **do**
  if $\sigma_p^f(i) \neq \sigma_p^o(i+1)$ **then**
    $\sigma \leftarrow$ search_sub-rotation$(p, \sigma_p^f(i), \sigma_p^a(i) + t_r, \sigma_p^o(i+1), \sigma_p^d(i+1) - tr)$ (Algorithm 3)
    if $|\sigma| \neq 0$ **then**
      $\sigma_p \leftarrow \sigma_p(1,i).\sigma.\sigma_p(i+1, |\sigma_p|)$
      $i \leftarrow i + |\sigma| + 1$
    else
      if $\sigma_p(i+1)$ is not a maintenance **then**
        Cancel $\sigma_p(i+1)$
      else
        Cancel $\sigma_p(i)$
        if $i \neq 1$ **then**
          $i \leftarrow i-1$
        end if
      end if
    end if
  else
    $i \leftarrow i+1$
  end if
end while

In a first phase, we check if there is a connection problem between O_p and the departure airport of the first leg of the rotation. If there is a problem, a sub-rotation connecting O_p and $\sigma_p^o(1)$ in a time window $[t_p^r, \sigma_p^d(1) - tr]$ is searched using Algorithm 3, described below. If a sub-rotation is not found, the first leg is cancelled and the process is iterated; otherwise the sub-rotation is inserted at the head of σ_p and the algorithm moves to the second phase.

During the second phase, the connectivity between all couples of legs $\sigma_p(i)$ and $\sigma_p(i+1)$ with $1 \leqslant i < |\sigma_p|$ is checked. Starting with $i=1$, the following process is iterated until $i=|\sigma_p|$. If $\sigma_p^f(i) = \sigma_p^o(i+1)$, we move to the next link in the rotation. Otherwise, a sub-rotation connecting $\sigma_p^f(i)$ and $\sigma_p^o(i+1)$ in a time window $[\sigma_p^a(i), \sigma_p^d(i+1)-tr]$ is searched using Algorithm 3. If a sub-rotation is found, it is inserted between $\sigma_p(i)$ and $\sigma_p(i+1)$ and i is modified such that $\sigma_p(i+1)$ is the next leg to be considered, otherwise two cases can happen. The first case occurs if $\sigma_p^f(i+1)$ is not a maintenance, then $\sigma_p(i+1)$ is cancelled and the process is iterated. The second case arises when $\sigma_p^f(i+1)$ is a maintenance. Therefore, $\sigma_p(i+1)$ cannot be cancelled and the leg $\sigma_p(i)$ is cancelled instead. The algorithm moves back to $\sigma_p(i-1)$.

Algorithm 3 inputs are: (i) an origin airport O; (ii) a ready time t_r; (iii) a destination airport D; (iv) a due time t_d; (v) an aircraft p. The algorithm returns a new rotation from O to D minimizing the time of arrival t^* at D. Note that such a rotation may not exist. The algorithm follows closely the Dijkstra algorithm.

Algorithm 3 Sub-rotation search

for all $a \in A$ **do**
  $\pi_a \leftarrow$ nil
  $eta_a \leftarrow +\infty$
end for
$eta_O \leftarrow t_r$
$Q \leftarrow A \backslash \{O\}$
repeat
  Extract from Q the node a with the smallest eta_a
  if $a \neq D$ **then**
    for all $a' \in A$ adjacent to a **do**
      Perform a relaxation on (a, a') (Algorithm 4)
    end for
  end if
until $a = D$

The algorithm works on the graph (A, E) with $E = \{(a_1, a_2) \in A \times A | d_{a_1 a_2} \leqslant r_p^{\max}\}$. For a node $a \in A$, π_a is the predecessor of a and eta_a the estimated time of arrival at a.

Each $e=(a_1, a_2) \in E$ has a weight $w_e = f_e^p + tr + s_e$ with f_e^p the flight time for p from a_1 to a_2 and s_e the waiting time at airport a_1 before departure. This waiting time is induced by the need to respect the airport landing and takeoff capacities. This waiting time is computed dynamically by the algorithm during the relaxation phase of an arc (a, a') described in Algorithm 4. The estimated time of departure from a is etd and the estimated time of arrival at a' is eta. We define etd_h the corresponding time window $h \in RTW$ and etd_r the remaining time until the end of h. The same information is defined for eta. For each $h \in RTW$ and an airport a, $\#_a^{lh}$ is the number of scheduled landings at a during h and $\#_a^{th}$ the number of takeoffs.

Algorithm 4 Relaxation between two nodes a and a'

$etd \leftarrow eta_a + tr$
$eta \leftarrow etd + f_{aa'}^p$
while $(eta \leqslant t_d)$ and $((\#_a^{tetd_h} = c_a^{tetd_h})$ or $(\#_{a'}^{leta_h} = c_a^{leta_h}))$ **do**
  if $\#_a^{tetd_h} = c_a^{tetd_h}$ **then**
    $eta \leftarrow eta + etd_r$
    $etd \leftarrow etd + etd_r$
  else
    $eta \leftarrow eta + eta_r$
    $etd \leftarrow etd + eta_r$
  end if
end while
if $etd \leqslant t_d$ and $eta < eta_{a'}$ **then**
  $\pi_{a'} \leftarrow a$
  $eta_{a'} \leftarrow eta$
end if

3.2. Phase 2: passenger assignment

In this phase, we try to assign groups of passengers from G_p to itineraries so that they can reach their destination. The assignment heuristic (Algorithm 5) is presented in Section 3.2.1. The search for an itinerary for a given group of passengers is modelled as a shortest path problem solved by Algorithm 6 described in Section 3.2.2.

Algorithm 5 Assignment of passengers to existing rotations.

$G_1 \leftarrow G_O$
for $G_x = G_t, G_o, G_i$ **do**
  while $G_x \neq \emptyset$ **do**
    $G \leftarrow g \in G_x$ such that $\nexists g' \in G_x, p_g S_g < p_{g'} s_{g'}$
    $G_x \leftarrow G_x \setminus \{g\}$
    $\gamma \leftarrow$ search_itinerary(O_g, t_g^r, D_g, t_g^d) (Algorithm 6)
    if $|\gamma| \neq 0$ **then**
      if $\min_{1 \leqslant i \leqslant |\gamma|} \gamma^c(i) < s_g$ **then**
        $g' \leftarrow g$
        $s_g \leftarrow \min_{1 \leqslant i \leqslant |\gamma|} \gamma^c(i)$
        $s_{g'} \leftarrow s_g - \min_{1 \leqslant i \leqslant |\gamma|} \gamma^c(i)$
        $\gamma_{g'} \leftarrow \emptyset$
        $G_1 \leftarrow G_1 \cup \{g'\}$
        $G_x \leftarrow G_x \cup \{g'\}$
      end if
      $\gamma_g \leftarrow \gamma$
    end if
  end while
  end for

Algorithm 6 Passenger itinerary search

$\gamma^* \leftarrow \emptyset, t^* \leftarrow +\infty$, $c^* \leftarrow 0$
$L \leftarrow \{(O, \emptyset, t_r, +\infty)\}$ // L is a list of labels
while $L = \emptyset$ **do**
  $n \leftarrow n = (d, \gamma, t_r, c) \in L$ such that $\nexists n' = (d', \gamma'', t'_r, c') \in L$, $(t'_r < t_r)$ or $((t'_r = t_r)$ and $(c' > c))$
  $L \leftarrow L \setminus \{n\}$
  if $(d = D)$ and $((c > c^*)$ or $((c = c^*)$ and $(t < t^*))$ **then**
    $\gamma^* \leftarrow \gamma, t^* \leftarrow t, c^* \leftarrow c$
  else
    for all $p \in P$ **do**
      for all $i \in [1, |\sigma_p|]$ such that $\sigma_p^o(i) = d$ and $t + cd < \sigma_p^d(i)$ **do**
        if $\sigma_p^a(i) < t_d$ **then**
          $n_a \leftarrow (\sigma_p^f(i), \gamma.\sigma_p(i), \sigma_p^a(i), \min(\sigma_p^c(i), c))$
          for all $n' \in L$ **do**
            if n_a dominates n' **then**
              $L \leftarrow L \setminus \{n'\}$
            end if
          end for
          If $\nexists n' \in L$ such that n' dominates n_a **then**
            $L \leftarrow L \cup \{n_a\}$
          end if
        end if
      end for
    end for
  end if
end while

3.2.1. Assignment heuristic. First, G_p is divided into three subsets:

1. $G_t = \{g \in G_p | w_g = truncated\}$,
2. $G_o = \{g \in G_p | w_g = outbound\}$,
3. $G_i = \{g \in G_p | w_g = inbound\}$.

The subset G_t has a higher priority than G_o and G_i because the itineraries of passengers from G_t are started but not completed and these passengers cost more to compensate. G_o has a higher priority than G_i.

Then, the groups are considered as shown in Algorithm 5 according to the priority of the sets and to a score $f_g = s_g \times p_g (\forall g \in G_p)$. For each group g, an itinerary from O_g to D_g is searched in the time window $[t_g^r, t_g^d]$ by means of Algorithm 6 (see Section 3.2.2). If an itinerary is found, the group is assigned to the new itinerary. If the seat capacity of the itinerary is strictly smaller than s_g, g is split into two, with the remining of the passengers forming a new group, which is inserted in G_p.

3.2.2. Passenger itinerary search. Algorithm 6 inputs are: (i) an origin airport O; (ii) a ready time t_r; (iii) a destination airport D; (iv) a due time t_d. The algorithm searches for an itinerary from O to D in the time window $[t_r, t_d]$. If such an itinerary exists, the algorithm returns the one being able to transport the greatest number of passengers, using the time of arrival at D to break ties.

The algorithm works on a dynamically built graph in which a node is defined by a label (d, γ, t, c) with d an airport, γ an itinerary from O to d, t the time of arrival at d, and c the maximum number of passengers that can be transported. The best solution is stored as a triplet: γ^* the itinerary from O to D, t^* the time of arrival at D, and c^* the maximum number of passengers that can be transported.

At each step, the algorithm selects the node $n = (d, \gamma, t, c)$ with the smallest t. The capacity is used to break ties. If the airport reached at this node is D, it is tested as a candidate for the best solution. Otherwise, we consider the set of aircraft leaving D after t plus the time needed to allow the connection. Each aircraft leads to the creation of a new node. Not all the nodes are kept as a dominance relation exists between two nodes. A node $n_1 = (d_1, p_1, t_1, c_1)$ dominates another node $n_2 = (d_2, p_2, t_2, c_2)$ if the three following criteria are verified: (i) $d_1 = d_2$; (ii) $t_1 \leqslant t_2$; (iii) $c_1 \geqslant c_2$.

3.3. Phase 3: Flight leg creation

In this step, new sub-rotations are inserted to existing aircraft rotations to allow the transportation of passengers from G_p.

First, for each airport pair (O, D), a meta-group is created. A meta-group $m \subseteq G_p$ associated to a pair (O_m, D_m) is the set of the groups of passengers in G_p wishing to go from O_m to D_m. The meta-group m is defined by its size $s_m = \sum_{g \in m} s_g$, its ready time $t_m^r = \max_{g \in m} t_g^r$, and its due time $t_m^d = \max_{g \in m} t_g^d$. The meta-groups form a set M.

Algorithm 7 is applied to each $m \in M$ starting with the largest meta-group until all the meta-groups have been considered. In this algorithm, the aircraft $p \in P$ are considered one after the other. We investigate the possibility to include a new sub-rotation before $\sigma_p(1)$ if $O_p = O_m$ and $t_p^r \leqslant t_m^r$ or between two legs $\sigma_p(i)$ and $\sigma_p(i+1)$ if $\sigma_p^f(i) = O_m$ and $\sigma_p^a(i) + tr \geqslant t_m^r$. A return trip from D_m to O_m may be necessary to respect the connectivity constraint of σ_p. The search for the sub-rotations is done by Algorithm 3, previously described in Section 3.1. If a sub-rotation is found, it is inserted in σ_p and as many passengers as possible from m are assigned to it. The process is iterated until an itinerary has been found or until all the possibilities have been exhausted.

Algorithm 7 Itinerary creation for a meta-group.

$\sigma \leftarrow \emptyset$
for all $p \in P$ **while** $\sigma = \emptyset$ **do**
  if $O_p = O_m$ **then**
    $i \leftarrow 0$
    $\sigma \leftarrow$ search_sub-rotation($O_m, t^r_m, D_m, \min(t^d_m, \sigma^{\mathrm{d}}_p(1) - t_r, p)$ (Algorithm 3)
    if $\sigma_p \neq \emptyset$ and $\sigma \neq \emptyset$ **then**
      $\sigma^{\mathrm{r}} \leftarrow$ search_sub-rotation($D_m, \sigma^{\mathrm{a}}(|\sigma|) + t_r, O_m, \sigma^d_p(1) - tr, p$)
      if $\sigma^r = \emptyset$ **then**
        $\sigma \leftarrow \emptyset$
      end if
    end if
  end if
  if $\sigma = \emptyset$ **then**
    for all $i \in [1, |\sigma_p|]$ such that $\sigma^f_p(i) = O_m$ and $t^r_m \leqslant \sigma^d_p(i) + tr \leqslant t^d_m$ **while** $\sigma = \emptyset$ **do**
      $\sigma \leftarrow$ search_sub-rotation($O_m, \sigma^d_p(i) + tr, D_m, \min(t^d_m, \sigma^d_p(i+1) - tr, p)$) (Algorithm 3)
      if $\sigma \neq \emptyset$ and $i \neq |\sigma_p|$ **then**
        $\sigma^r \leftarrow$ search_sub-rotation($D_m, \sigma^a(|\sigma|) + tr, O_m, \sigma^d_p(i+1) - tr, p$) (Algorithm 3)
        if $\sigma^{\mathrm{r}} \neq \emptyset$ **then**
          $\sigma \leftarrow \emptyset$
        end if
      end if
    end for
  end if
end for
if $\sigma \neq \emptyset$ **then**
  $c \leftarrow c^{\max}_p$
  while $c \neq 0$ and $m \neq \emptyset$ **do**
    $g \leftarrow g \in m$ such that $\nexists g' \in m, s_g > s_g$
    $m \leftarrow m \setminus \{g\}$
    $G_p \leftarrow G_p \setminus \{g\}$
    if $s_g \leqslant c$ **then**
      $\gamma_g \leftarrow \sigma$
      $c \leftarrow c \leftarrow s_g$
    else
      $g' \leftarrow g$
      $s_g \leftarrow c$
      $\gamma_g \leftarrow \sigma$
      $s_{g'} \leftarrow s_{g'} \leftarrow c$
      $m \leftarrow m \cup \{g'\}$
      $M \leftarrow M \cup \{m\}$
      $c \leftarrow 0$
    end if
  end while
    $\sigma_p \leftarrow \sigma_p(1, i).\sigma.\sigma^r.\sigma_p(i+1, |\sigma_p|)$
end if

4. Computational results

The algorithm was coded in C and was run on an Intel Core 2 Duo E6550 2.33 Ghz CPU.

It was tested on instances provided by Amadeus and it was evaluated with the objective function used for the ROADEF 2009 challenge, also provided by Amadeus as a black box executable. They were divided into three sets (A, B, and X). Globally, the instances comprise a maximum of 2000 flights connecting 150 airports for a recovery period of at most 3 days. Detailed information concerning the number of flights, the number of aircraft, the number of airports, the number of passenger itineraries, the number of disrupted flights, the number of disrupted aircraft, the number of disrupted airports and the length of the recovery period are provided in Tables 1–3.

Computational times were limited to 10 min on a reference computer (our test computer is slightly slower than the reference computer). Table 4 gives the average score of the participants of the ROADEF 2009 challenge obtained with the same objective function. This average score is computed on all instances of Set B and on instances of Set X with the exception of $X01$, $X02$, $X03$, and $X04$ instances (because most of participants obtained no solution on these instances). To compute the average score, normalized scores were used. Let $z(M, I)$ denote the objective function value obtained by Method M on Instance I. Let $zb(I)$ and $zw(I)$ denote the best and worst objective function values found by all methods on instance I, respectively. The normalized score obtained by Method M on Instance I is given by $(zw(I) - z(M, I))/(zw(I) - zb(I))$.On this subset of instances, we obtain the second best score behind Bisaillon *et al.* However, if the instances $X01$, $X02$, $X03$, $X04$ are considered, NCF obtains a better average score (93.47) than Bisaillon *et al* (91.46), which is the only team reporting results for these instances. Globally, it can be concluded that our method provides very good results for these instances.

Tables 5–7 report detailed results on all instances. It provides our results and the computational times used to obtain them, as well as the results obtained by the three best participant teams except for set A for which their final results are unknown. Results in bold font indicate that it is the best found solution, INF indicates that no result was returned in the time limit. On instances of Sets B and X, NCF found 13 best found solutions out of 22 instances; in comparison, Bisaillon *et al* found 1 out of 22, Hanafi *et al* found 6 out of 22. It is another indicator of the quality of the results we obtain.

In order to compare ourselves to the other eight methods submitted to the challenge, Figures 1–3 indicate for the instances of Sets B, XA, XB the worst and the best obtained scores, as well as the median score and the score of NCF. Note that for instances XA and XB, as not all

Table 1 Instance Set *A* characteristics

	*A*1	*A*2	*A*3	*A*4	*A*5	*A*6	*A*7	*A*8	*A*9	*A*10
#flights	608	608	608	608	608	608	608	608	608	608
#aircraft	85	85	85	85	85	85	85	85	85	85
#airports	35	35	35	35	35	35	35	35	35	35
#itineraries	1943	1943	1943	1943	3959	1872	1872	1872	1872	3773
#disrupted flight	63	107	83	41	0	63	107	83	41	0
#disrupted aircraft	0	0	1	0	0	0	0	1	0	0
#disrupted airport	0	0	0	2	35	0	0	0	2	35
Recovery period	1	1	1	1	2	1	1	1	1	2

Table 2 Instance Set *B* characteristics

	*B*1	*B*2	*B*3	*B*4	*B*5
#flights	1422	1422	1422	1422	1422
#aircraft	255	255	255	255	255
#airports	44	44	44	44	44
#itineraries	11 214	11 214	11 214	11 214	11 214
#disrupted flight	229	254	228	229	0
#disrupted aircraft	0	0	1	0	0
#disrupted airport	0	0	0	1	2
Recovery period	2	2	2	2	2

	*B*6	*B*7	*B*8	*B*9	*B*10
#flights	1422	1422	1422	1422	1422
#aircraft	255	255	255	255	255
#airports	44	44	44	44	44
#itineraries	11 565	11 565	11 565	11 565	11 565
#disrupted flight	229	254	228	229	0
#disrupted aircraft	0	0	1	0	0
#disrupted airport	0	0	0	1	2
Recovery period	2	2	2	2	2

Table 3 Instance Set *X* characteristics

	*X*01	*X*02	*X*03	*X*04	*XA*01	*XA*02
#flights	2178	2178	2178	2178	608	608
#aircraft	618	618	618	618	85	85
#airports	168	168	168	168	35	35
#itineraries	28 308	28 308	29 151	29 151	1943	3959
#disrupted flight	0	0	0	0	82	0
#disrupted aircraft	1	1	1	1	3	3
#disrupted airport	1	0	1	0	0	35
Recovery period	3	3	3	3	2	2

	*XA*03	*XA*04	*XB*01	*XB*02	*XB*03	*XB*04
#flights	608	608	1422	1422	1422	1422
#aircraft	85	85	255	255	255	255
#airports	35	35	44	44	44	44
#itineraries	1872	3773	11 214	11 214	11 565	11 565
#disrupted flight	82	0	228	0	227	0
#disrupted aircraft	3	3	3	1	4	3
#disrupted airport	0	35	0	2	0	2
Recovery period	2	2	2	2	2	2

competitors found a solution during the allowed time for each instance, the charts indicate for each instance the number of competitors that found a solution. We do not provide a chart for instances *X*01–*X*04 because only the method proposed by Bisaillon *et al* (2009) and NCF found a solution.

On Set *B*, it appears that for the two instances where we did not obtain the best results (B5 and B6), we are not far from the median and well ahead of the worst solutions. In these instances, the only disruptions are large airport capacity reductions. More precisely, it is the closing of the two main airports (which are hubs) during most of the *RTW*. It leads to a large number of passengers who have to be rerouted. The average results of our method on these two instances can be explained as follows. NCF modifies as little as possible the initial plan in terms of aircraft rotations and passenger itineraries. We mean that NCF never modifies an itinerary from the set of passengers G/G_p. Therefore, passengers having these hubs as destination airport and supposed to reach them during their capacity disruption, could not be flown there, even after the end of the disruption. Indeed, the existing flights that have not been affected by any disruption may already be full and thus will not be able to fly the stranded passengers. Whereas other methods, like Bisaillon *et al* (2009) for instance, allow modification of any passenger itinerary.

Table 4 Final scores for participant teams

Team	*Average score (%)*
Bisaillon, Cordeau, Laporte, Pasin	95.80
Jozefowiez, Mancel, Mora-Camino	92.87
Hanafi, Wilbaut, Mansi, Clautiaux	92.63
Acuna-Agost, Michelon, Feillet, Gueye	74.26
Eggermont, Firat, Hurkens, Modelski	72.00
Darlay, Kronek, Schrenk, Zaourar	70.62
Peekstok, Kuipers	70.31
Dickson, Smith, Li	42.02
Eggenberg, Salani	20.43

Table 5 Results on Set *A*

Instances	*A*01	*A*02	*A*03	*A*04	*A*05
NCF	150 095.70	377 992.90	473 992.15	2 520 586.00	13 640 667.40
Time	< 1 s	< 1 s	< 1 s	< 1 s	25 s
Instances	*A*06	*A*07	*A*08	*A*09	*A*10
NCF	111 540.50	623 236.80	997 137.80	6 163 295.90	23 840 247.85
Time	< 1 s	< 1 s	< 1 s	< 1 s	20 s

Table 6 Results on Set *B*

Instances	*B*01	*B*02	*B*03	*B*04	*B*05
NCF	**971 182.50**	**1 220 708.30**	**1 007 565.70**	**1 101 394.80**	25 302 036.95
Time	28 s	39 s	28 s	30 s	2 min 06 s
Bisaillon *et al*	983 731.75	1 522 452.75	1 031 825.30	1 192 519.20	15 639 190.80
Hanafi *et al*	5 813 896.95	9 950 888.70	5 569 623.95	5 775 277.70	13 139 974.30
Acuna-Agost *et al*	1 540 123.55	2 656 393.25	1 572 754.95	1 629 491.90	14 042 563.85
Instances	*B*06	*B*07	*B*08	*B*09	*B*10
NCF	**3 218 000.10**	**5 039 744.20**	**3 509 318.00**	**3 967 344.70**	59 289 841.80
Time	24 s	34 s	24 s	25 s	1 min 21 s
Bisaillon *et al*	3 789 254.05	5 488 693.00	4 069 557.35	5 906 239.15	52 355 192.80
Hanafi *et al*	9 095 248.10	19 144 460.30	10 099 607.00	10 176 173.55	**34 523 605.00**
Acuna-Agost *et al*	4 926 204.05	8 381 142.30	5 092 952.60	5 414 178.30	40 080 949.40

Table 7 Results on Set *X*

Instances	*XA*01	*XA*02	*XA*03	*XA*04
NCF	150 857.60	4 787 273.45	404 964.20	9 352 557.15
Time	< 1 s	25 s	< 1 s	20 s
Bisaillon *et al*	462 571.10	2 238 311.75	959 080.90	5 480 962.75
Hanafi *et al*	**116 195.20**	**1 475 322.10**	**285 287.05**	**4 112 262.60**
Acuna-Agost *et al*	214 321.95	2 010 576.25	433 172.00	6 575 537.15
Instances	*XB*01	*XB*02	*XB*03	*XB*04
NCF	**1 194 006.65**	24 885 515.20	**4 251 062.90**	57 588 009.55
Time	29 s	2 min 06 s	25 s	1 min 19 s
Bisaillon *et al*	1 352 823.05	17 064 421.50	6 463 354.30	53 543 381.45
Hanafi *et al*	5 985 772.05	12 716 512.00	11 124 244.55	**34 331 225.80**
Acuna-Agost *et al*	INF	INF	INF	INF
Instances	*X*01	*X*02	*X*03	*X*04
NCF	**283 033.85**	**135 872.00**	**1 835 571.95**	590 774.35
Time	3 min 21 s	29 s	3 min 50 s	23 s
Bisaillon *et al*	1 116 142.85	806 011.20	2 682 125.00	**485 904.75**
Hanafi *et al*	INF	INF	INF	INF
Acuna-Agost *et al*	INF	INF	INF	INF

The same conclusions can be reached for the instances *XA* and *XB*, which are built on the same structure as the instances *A* and *B*, that is, they have the same size in terms of number of aircraft, number of passengers and number of airports, and they have the same kind of perturbations, but in a larger scale.

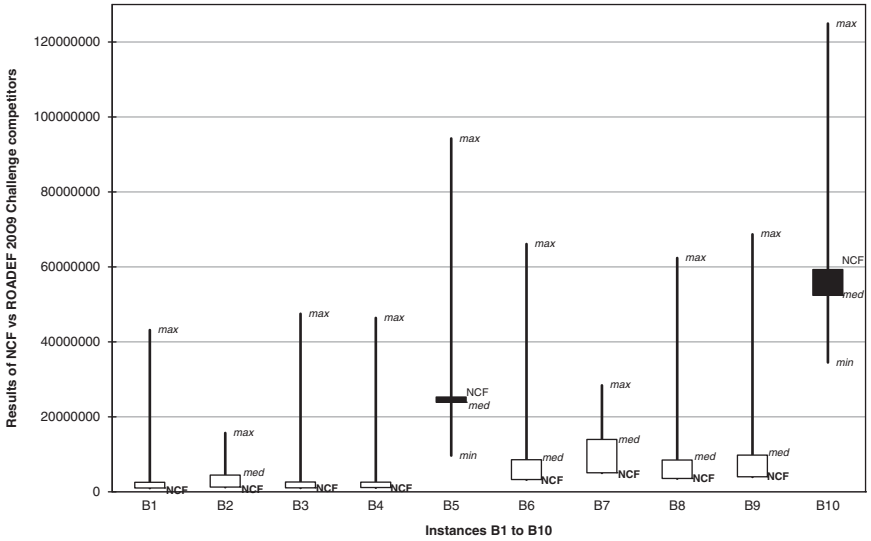


Figure 1 Comparison of all the methods on the instance Set *B*.

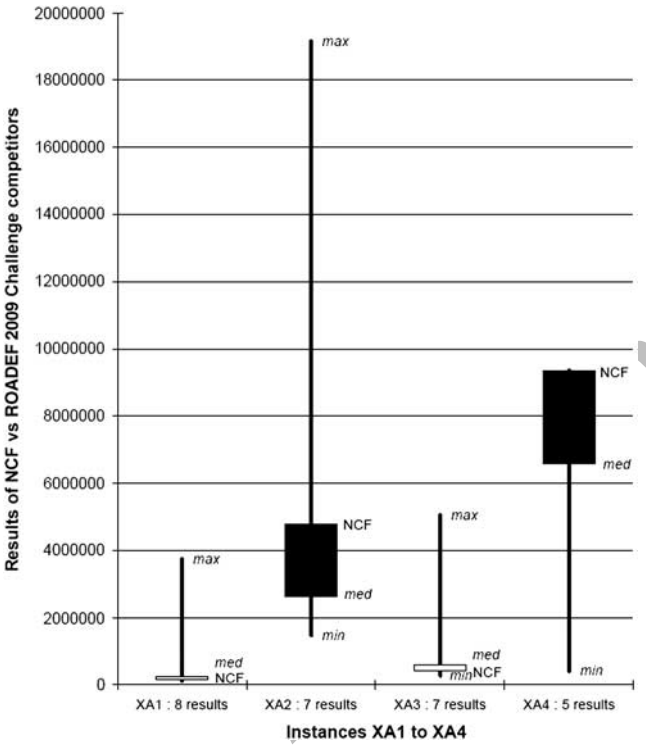


Figure 2 Comparison of all the methods on the instance Set *XA*.

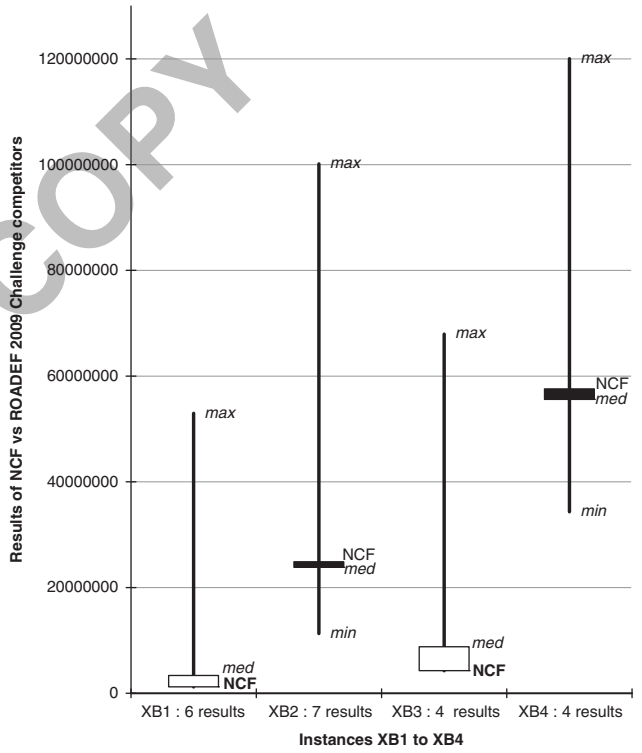


Figure 3 Comparison of all the methods on the instance Set *XB*.

Our method largely outperforms the one by Bisaillon *et al* (2009) on the instances *X* except for *X*04, their algorithm being the only one to report results on these. Unlike instances *XA* and *XB*, the main difficulty of instances *X*01–*X*04 does not come from a large number of disruptions but from the size of the instances. Therefore, the graph to search for a new itinerary among the existing flights (NCF's Phase 2) can be quite large. However, our method following simple and deterministic rules is able to treat all the passengers while Bisaillon *et al*'s method relies on stochastic searches and must iterate this search. We believe that the size of the instances prevents Bisaillon *et al*'s method to perform enough searches to converge. Concerning instance *X*04, given the size of the instance, the

cost associated to not being able to fly a group of passengers to its destination, and the size of the groups, a difference of 100 000 in the solution cost may not be relevant. However, as the details of the solutions of the other teams are unknown, it is not possible to investigate the variation in more details.

Finally, another advantage of NCF is its efficiency in terms of computational time. On a majority of instances, it does not pass over 1 min, and it never passes over 4 min, remaining far below the 10 min time limit imposed by the challenge.

5. Conclusion

We have proposed a simple and efficient heuristic method for the integrated flight, aircraft and passenger rescheduling problem. This work was done in the context of the ROADEF 2009 challenge. This paper shows the quality of the algorithm. When the complete set of instances is considered, the method obtains the best average score.The NCF heuristic also obtains the best found solution on more than half the instances. Additionally, it is worthy to underline the speed of the NCF method. This work is continued in a collaboration with Amadeus.

Acknowledgements—Thanks are due to the referees for their valuable comments.

References

Acuna Agost R, Feillet D, Michelon P and Gueye S (2009). Rescheduling flights, aircraft and passengers simultaneously under disrupted operations—a mathematical programming approach based on statistical analysis. Submitted to the Anna Valicek Medal 2009, http://www.agifors.org/award_home.jsp, accessed 22 February 2012.

Argüello MF, Bard JF and Yu G (1997). A grasp for aircraft routing in response to groundings and delays. *Journal of Combinatorial Optimization* **1**(3): 211–228.

Artigues C, Bourreau E, Afsar HM, Briant O and Boudia M (forthcoming). Disruption management for commercial airlines: Overview of methods and official results for the ROADEF 2009 challenge. *European Journal of Industrial Engineering* (accepted).

Ball M, Banrhart C, Nemhauser GL and Odoni A (2007). Air transportation: Irregular operations and control. In: Barnhart C and Laporte G (eds). *Handbooks in Operations Research and Management Science: Transportation*. Elsevier: North-Holland, pp. 1–67.

Barnhart C, Kniker T and Lohatepanont M (2002). Itinerary-based airline fleet assignment. *Transportation Science* **v**: 199–217.

Bisaillon S, Cordeau JF, Laporte G and Pasin F (2009). *A large neighborhood search heuristic for the aircraft and passenger recovery problem*. Tech. Rep. CIRRELT-2009-42. CIRRELT, Montreal, Quebec, Canada.

Bratu S and Barhnart C (2006). Flight operations recovery: New approaches considering passenger recovery. *Journal of Scheduling* **9**(3): 279–298.

Clarke L, Johnson E, Nemhauser GL and Zhu Z (1997). The aircraft rotation problem. *Annals of Operations Research* **69**(0): 33–46.

Clarke M (2005). Passenger reaccommodation a higher level of customer service. *Airline Group of the International Federation of Operational Research Societies (AGIFORS) Airline Operations Study Group Meeting*.

Clausen J, Larsen A, Larsen J and Rezanova NJ (2010). Disruption management in the airline industry—Concepts, models and methods. *Computers and Operations Research* **37**(5): 809–821.

Eggenberg N, Salani M and Bierlaire M (2007). *A column generation algorithm for disrupted airline schedules*. Tech. Rep. TRANSP-OR071203, Transport and Mobility Laboratory of Ecole Polytechnique Fédérale de Lausanne, Lauzanne, Switzerland.

Filar JA, Manyem P and White K (2001). How airlines and airports recover from schedule perturbations: A survey. *Annals of Operations Research* **108**(4): 315–333.

Lettovsky L (1997). *Airline operations recovery: An optimization approach*. PhD Thesis, Georgia Institute of Technology, Atlanta, USA.

Lettovsky L, Johnson EL and Nemhauser GL (2000). Airline crew recovery. *Transportation Science* **34**(4): 337–348.

Medard CP and Sawhney N (2007). Airline crew scheduling from planning to operations. *European Journal of Operational Research* **183**(3): 1013–1027.

Palpant M, Boudia M, Robelin CA, Gabteni S and Laburthe F (2009). ROADEF 2009 challenge: Disruption management for commercial aviation. http://challenge.roadef.org/2009/files/challenge_en.pdf, accessed 22 February 2012.

Stojkovic M, Soumis F and Desrosiers J (1998). The operational airline crew scheduling problem. *Transportation Science* **32**(3): 232–245.

Teodorovic D and Guberinic S 1984. Optimal dispatching strategy on an airline network after a schedule perturbation. *European Journal of Operational Research* **15**(2): 178–182.

Thengvall B, Bard J and Yu G (2001). Multiple fleet aircraft schedule recovery following hub closures. *Transportation Research Part A* **35**(4): 289–308.

Wei G, Yu G and Song M (1997). Optimization model and algorithm for crew management during airline irregular operations. *Journal of Combinatorial Optimization* **1**(3): 305–321.

Appendix

Table A1 List of notations introduced in Section 2

RTW	recovery time window; $RTW=[RTW_s;\ RTW_e]$
A	set of airports
c_a^{lh}	maximum number of landings that can occur during the time window $h\subseteq RTW$ at airport $a\in A$
c_a^{th}	maximum number of takeoffs that can occur during the time window $h\subseteq RTW$ at airport $a\in A$
P	set of aircraft
$c_p^{\max}$	seat capacity of aircraft $p\in P$
σ_p	rotation of aircraft $p\in P$
$\sum$	set of rotations; $\sum=\{\sigma_p \mid p\in P\}$
O_p	origin airport of σ_p; $O_p\in A$
$\sigma_p(i)$	ith leg in σ_p
$\sigma_p^o(i)$	origin airport of $\sigma_p(i)$; $\sigma_p^o(i)\in A$ and $\sigma_p(1)=O_p$
$\sigma_p^f(i)$	destination airport of $\sigma_p(i)$; $\sigma_p^f(i)\in A$
$\sigma_p^d(i)$	departure time of $\sigma_p(i)$; $\sigma_p^d(i)\in RTW$
$\sigma_p^a(i)$	arrival time of $\sigma_p(i)$; $\sigma_p^a(i)\in RTW$
$\sigma_p^c(i)$	remaining seat capacity of $\sigma_p(i)$
$r_p^{\max}$	maximum range of aircraft P
tr	turnaround delay
t_p^r	earliest possible takeoff time for aircraft $p\in P$
P_m	set of aircraft that must undergo a maintenance during RTW; $P_m\subseteq P$
G	set of passenger groups
s_g	size of passenger group $g\in G$
O_g	origin airport of passenger group g; $O_g\in A$
D_g	destination airport of passenger group g; $D_g\in A$
P_g	ticket price of passenger group g
w_g	'inbound trip' or an 'outbound trip' status of passenger group g
γ_g	itinerary of passenger group g
Γ	set of itineraries of all passenger groups; $\Gamma=\{\gamma_g \mid g\in G\}$
$\gamma_g(i)$	ith leg in itinerary γ_g
$\lvert\gamma_g\rvert$	number of legs in γ_g
$\gamma_p^o(i)$	origin airport of $\gamma_g(i)$; $\gamma_p^o(i)\in A$ and $\gamma_p^o(1)=O_p$
$\gamma_p^f(i)$	destination airport of $\gamma_g(i)$; $\gamma_p^f(i)\in A$
$\gamma_p^d(i)$	departure time of $\gamma_g(i)$
$\gamma_p^a(i)$	arrival time of $\gamma_g(i)$
cd	connection delay between two consecutive legs of an itinerary
t_g^r	earliest possible time of departure of passenger group g; $t_g^r=\gamma_g^d(1)$
t_g^d	latest possible time of arrival of passenger group g
D	set of flight delays
C	set of flight cancellations
B	set of aircraft breakdowns
R	set of airport capacity reductions

Received February 2010;
accepted January 2012 after three revisions

RAIRO-Oper. Res. 45 (2011) 353–364
DOI: 10.1051/ro/2012004

RAIRO Operations Research
www.rairo-ro.org

A BRANCH-AND-PRICE ALGORITHM FOR THE WINDY RURAL POSTMAN PROBLEM

Hasan Murat Afsar[1], Nicolas Jozefowiez[2,3] and Pierre Lopez[2,4]

Abstract. In this paper, we propose an exact solution method for the windy rural postman problem (WRPP). The motivation to study this problem comes from some real-life applications, such as garbage collecting in a predefined sector with hills, where the traversing or the servicing speed can change following the direction. We present a Dantzig-Wolfe decomposition and a branch-and-price algorithm to solve the WRPP. To the best of our knowledge, Dantzig-Wolfe decomposition has never been used to solve that problem. The numerical results show that optimal solutions are found in a very reasonable amount of time on instances with up to 100 nodes and 180 edges.

Keywords. Branch-and-price, windy rural postman problem.

Mathematics Subject Classification. 90B06, 90C10.

1. Introduction

The purpose of this paper is to investigate the use of a Dantzig-Wolfe decomposition and a branch-and-price algorithm to solve the windy rural postman problem (WRPP). The WRPP is an asymmetric variant of the rural postman problem (RPP) [1]. The RPP is an arc routing problem that arises when a subset of the arcs must be visited. It is a general case of the Chinese Postman Problem (CPP)

Received December 6, 2011. Accepted January 31, 2012.

[1] Université de Technologie de Troyes (UTT), Institut Charles Delaunay, LOSI, 12 rue Marie Curie, 10010 Troyes, France. murat.afsar@utt.fr

[2] CNRS, LAAS, 7 avenue du Colonel Roche, 31077 Toulouse Cedex 4, France.

[3] Université de Toulouse, INSA, LAAS, 31400 Toulouse, France.

[4] Université de Toulouse, LAAS, 31400 Toulouse, France.

Article published by EDP Sciences

in which a circuit that visits all the arcs at least once must be found [1,14]. In the WRPP, the cost of traversing an edge depends on the traveling direction.

The problem can be described as follows: given an undirected graph $G = (V, E \cup E_R)$, a cost function is associated with every edge/direction (c_{ij} and c_{ji} are the costs of traversing edge $[i, j]$ in either directions) and E_R is the subset of required edges (*i.e.*, needing service), which are supposed to be visited at least once. The WRPP then consists in finding a minimum cost circuit traversing every edge of E_R at least once. Note that any edge can be visited many times without serving, but each required edge is served only once.

First introduced by Orlof [15], the RPP has been shown to be NP-hard by Lenstra and Rinnooy Kan, in the general case [12]. As the RPP is a special case of the WRPP, the WRPP is also NP-hard. Christofides *et al.* [3] proposed a branch-and-price method for the RPP. Branch-and-cut methods were also considered by Sanchis [16], Corberán and Sanchis [5], Letchford [13], and more recently by Ghiani and Laporte [8]. Benavent *et al.* [1] were inspired by a property obtained by Win [17] for the windy version of the CPP to solve the WRPP. Win showed that if the graph is Eulerian, the windy CPP is polynomial. A feasible solution for the WCPP is also feasible for the WRPP, since the windy CPP is a special case of the WRPP. By using a stochastic method to duplicate the edges or changing their costs in a given range, Benavent *et al.* then present a multi-start scatter search algorithm. They also detail a cutting-plane procedure to exactly solve or to obtain lower bounds of the WRPP [2].

In our work, during a field analysis for a waste collection project, the WRPP was used as a key component in order to find an optimal rotation of a truck. Our main contribution is the study of the WRPP by means of a Dantzig-Wolfe decomposition and its solution by means of a branch-and-price algorithm. The proposed algorithm is shown to be competitive with a state-of-the-art branch-and-cut algorithm from the literature [2] and appears as a good stepping stone for further improvement.

The paper is organized as follows. Section 2 introduces mathematical models obtained as results of the Dantzig-Wolfe decomposition and a mixed-integer model for the sub-problem. Section 3 details the branch-and-price algorithm and algorithms to solve the sub-problem. Computational results and conclusions follow in Sections 4 and 5, respectively.

2. Mathematical models

Starting from the edge-based model proposed by Benavent *et al.* [2], we propose a Dantzig-Wolfe decomposition into a master problem and a sub-problem. We also present a lower bound based on the Lagrangian relaxation of required edge service constraints.

2.1. Edge-based model

Benavent *et al.* [2] give an extension to the WRPP of the mixed integer programming (MIP) model proposed by Grötschel and Win [9] for the windy postman problem. The integer decision variable u_{ij} counts the number of times the edge (i, j) is traversed from i to j. The set of the incident edges of vertex i is denoted by $\delta(i)$. The sub-graph G_R induced by E_R is not necessarily connected. We denote by $V_1, V_2, \ldots, V_p$ the connected components of G_R; they are called R-connected components in [6].

Model 1

$$C^{*,(1)}_{WRPP} \triangleq \min \sum_{(i,j)\in E} (c_{ij}u_{ij} + c_{ji}u_{ji}) \tag{2.1}$$

s.t.:

$$u_{ij} + u_{ji} \geq 1 \quad \forall (i, j) \in E_R \tag{2.2}$$

$$\sum_{(i,j)\in\delta(i)} (u_{ij} - u_{ji}) = 0 \quad \forall i \in V \tag{2.3}$$

$$\sum_{(i,j)|i\in S, j\in V\setminus S} u_{ij} \geq 1 \quad \forall S = \bigcup_{k\in Q} V_k,\ Q \subset \{1, \ldots, p\} \tag{2.4}$$

$$u_{ij}, u_{ji} \in \mathbb{N} \quad \forall (i, j) \in E. \tag{2.5}$$

In Model 1, the objective is to minimize the total travel cost defined by equation (2.1). Constraints (2.2) impose that every required edge is visited at least once. Constraints (2.3) are the flow conservation constraints. The connectivity between connected components is ensured by constraints (2.4): there must be at least one edge between any combination of connected components and the rest of the nodes. Note that constraints (2.4) are binding constraints and they significantly complicate the solution of the problem. Dantzig and Wolfe [7] propose a special reformulation and decomposition of the problem into a master and a sub-problem. Constraints (2.3) and (2.4) are considered by the sub-problem. The master problem takes into account only the set covering constraints (2.2).

2.2. Mathematical model of the restricted master problem

The master problem (MP) obtained by the Dantzig-Wolfe decomposition is a path-based model in which only the constraints related to the necessity of visiting all the required arcs are kept. Now, each variable represents a feasible *closed-walk*, *i.e.*, a circuit passing at least once through each edge of E_R. We will denote by P the set of all the feasible closed-walks. The length of a closed-walk $p \in P$ will be denoted by c_p. In spirit with the column generation approach we have adopted, we will only consider at a time a subset $P' \subset P$. The restricted master problem (RMP) is formulated as follows. For each $p \in P'$, let $\lambda_p \geq 0$ equal to 1 if and only if p is used. For each edge $(i, j) \in E_R$, and γ_p^{ij} is equal to 1 if and only if

the closed-walk $p \in P'$ serves (i, j) by traveling from i to j. Then, the restricted master problem (RMP) is:

$$C^*_{RMP} \triangleq \min \sum_{p \in P'} c_p \lambda_p \tag{2.6}$$

s.t.:

$$\sum_{p \in P'} \lambda_p = 1 \tag{2.7}$$

$$\sum_{p \in P'} \left(\gamma_p^{ij} + \gamma_p^{ji}\right) \lambda_p = 1 \quad \forall [i, j] \in E_R \tag{2.8}$$

$$\lambda_p \geq 0 \quad \forall p \in P'. \tag{2.9}$$

The objective function (2.6) minimizes the total cost of the chosen closed-walks. Only one closed-walk should be chosen (constraint (2.7)). Constraints (2.8) guarantee that every required edge in E_R is used only once and in a unique direction.

2.3. Mixed integer model of the sub-problem

First, we must underline that, even if the graph is asymmetric and the cost of an edge depends on the travel direction, an edge has a unique dual variable independent from the direction. This is the main reason why we use edge notation instead of arc notation.

In the sub-problem, we are looking for closed-walks with negative reduced costs. Let π_0 and π_{ij} be the dual variables associated to the constraints (2.7) and (2.8), respectively. We define the reduced cost $\bar{c}_p$ of a closed-walk p as follows:

$$\bar{c}_p = c_p - \sum_{[i,j] \in E} \left(\gamma_p^{ij} + \gamma_p^{ji}\right) \pi_{ij} - \pi_0.$$

We introduce a parameter β_p^{ij} which counts the number of travels through edge $[i, j]$, from i to j, with or without service, on a closed-walk p. Hence $\beta_p^{ij} - \gamma_p^{ij}$ is the number of times the vehicle traverses the edge $[i, j]$ without servicing. The cost of a closed-walk p can be rewritten as:

$$c_p = \sum_{[i,j] \in E} \left(\beta_p^{ij} c_{ij} + \beta_p^{ji} c_{ji}\right).$$

For a given closed-walk p, π_0 is constant. Thus, the reduced cost can be reformulated as:

$$\begin{aligned}
\bar{c}_p &= \sum_{[i,j] \in E} \left(\beta_p^{ij} c_{ij} + \beta_p^{ji} c_{ji}\right) - \sum_{[i,j] \in E_R} \left(\gamma_p^{ij} + \gamma_p^{ji}\right) \pi_{ij} - \pi_0 \\
&= \sum_{[i,j] \in E_R} \left[\gamma_p^{ij} \left(c_{ij} - \pi_{ij}\right) + \gamma_p^{ji} \left(c_{ji} - \pi_{ij}\right)\right] \\
&\quad + \sum_{[i,j] \in E_R} \left[\left(\beta_p^{ij} - \gamma_p^{ij}\right) c_{ij} + \left(\beta_p^{ji} - \gamma_p^{ji}\right) c_{ji}\right] + \sum_{[i,j] \in E_{NR}} \left(\beta_p^{ij} c_{ij} + \beta_p^{ji} c_{ji}\right) - \pi_0.
\end{aligned}$$

To formulate the sub-problem, we replace the parameters γ_p^{ij} and $\beta_p^{ij} - \gamma_p^{ij}$ by the binary variables y_{ij} which indicate whether edge $[i, j]$ is served on this closed-walk, and the integer variables z_{ij} which count the traversing of $[i, j]$ without servicing. The integer variable x_{ij} is the total number of traversing. An artificial variable f_{ij} for each $(i, j) \in E$ forces the walks to be connected. As previously, the set of the incident edges of vertex i is denoted by $\delta(i)$. Each edge has two costs (distances) following the directions c_{ij} and c_{ji}. A large number M is used as well in the formulation of the sub-problem.

Model 2

$$C_{WRPP}^{*,(2)} \triangleq \min \sum_{[i,j]\in E_R} \{y_{ij}(c_{ij}-\pi_{ij})+y_{ji}(c_{ji}-\pi_{ij})\}+\sum_{[i,j]\in E} (z_{ij}c_{ij}+z_{ji}c_{ji}) \quad (2.10)$$

s.t.:

$$y_{ij} + y_{ji} \leq 1 \quad \forall [i, j] \in E_R \quad (2.11)$$

$$y_{ij} + z_{ij} = x_{ij} \quad \forall [i, j] \in E \quad (2.12)$$

$$y_{ji} + z_{ji} = x_{ji} \quad \forall [i, j] \in E \quad (2.13)$$

$$\sum_{[i,j]\in\delta(i)} (x_{ij} - x_{ji}) = 0 \ \ \forall i \in V \quad (2.14)$$

$$f_{ij} \leq M x_{ij} \quad \forall [i, j] \in E \quad (2.15)$$

$$f_{ji} \leq M x_{ji} \quad \forall [i, j] \in E \quad (2.16)$$

$$\sum_{[i,j]\in\delta(i)} (f_{ij} - f_{ji}) = 1 \ \ \forall i \in V \setminus \{s\} \quad (2.17)$$

$$\sum_{(s,j)\in\delta(s)} f_{sj} \geq 1 \quad (2.18)$$

$$x_{ij}, x_{ji} \in \mathbb{R}, \quad z_{ij}, z_{ji} \in \mathbb{Z} \quad \forall [i, j] \in E$$
$$y_{ij}, y_{ji} \in \{0, 1\}, \quad f_{ij}, f_{ji} \in \mathbb{R} \quad \forall [i, j] \in E. \quad (2.19)$$

The objective function (2.10) minimizes the reduced cost of the closed-walk. As π_0 is constant at a given iteration for all the closed-walks, it is dropped from the objective function which minimizes now the difference between the total distance and the sum of the dual values. Constraints (2.11) allow serving a required edge at most once, only in one direction. Total times of traveling through edges in one or other direction are defined by constraints (2.12) and (2.13). Every time the postman comes into a vertex, he must go out of it (flow conservation, constraints (2.14)). Constraints (2.15) and (2.16) put in relation flow of type x and flow of type f: If there is a flow on an edge in one direction, then traversing counter direction should be greater than zero. There is a consumption of the flow at each vertex, except the source s (constraints (2.17) and (2.18)); thanks to these constraints, the walk is connected. If an edge is not used in the optimal solution, corresponding

counting and flow variables are equal to zero but there is always at least a couple of incident edges for all vertices. As the variables y_{ij} and z_{ij} are defined as binary and integer, respectively, x_{ij} and f_{ij} can be defined as real numbers (constraints (2.19)).

2.4. Lagrangian bound of the master problem

Lagrangian relaxation is a technique that works by removing hard constraints and putting them into the objective function, assigned with weights called Lagrangian multipliers. Each weight represents a penalty on the objective function if the particular constraint is not satisfied.We obtain the lower bound of the MP by pushing the constraints (2.8) to the objective function (2.6) with Lagrangian multipliers (π_{ij}):

$$\theta(\pi) = \min_{p \in P} \left\{ c_p + \sum_{[i,j] \in E_R} \pi_{ij} (1 - (\gamma_p^{ij} + \gamma_p^{ji})) \right\}. \tag{2.20}$$

We denote by $\pi = (\pi_{ij})_{[i,j] \in E}$ the vector of dual values associated with constraints (2.8).

By reformulating equation (2.20), we obtain:

$$\theta(\pi) = \min_{p \in P} \left\{ \bar{c}_p + \sum_{e \in E_R} \pi_{ij} + \pi_0 \right\}.$$

The expression $\sum_{e \in E_R} \pi_{ij} + \pi_0$ is constant for a given Lagrangian vector π. Therefore, every time the minimum reduced cost ($\bar{c}_p^* = \min_{p \in P} \bar{c}_p$) is found, *i.e.*, the sub-problem is solved to optimality, the Lagrangian lower bound is easily calculated. Having a valid lower bound of the MP can be useful to trigger an early stop of the column generation procedure. At each iteration, we can find out the gap between the best feasible solution found so far and the best Lagrangian lower bound (θ^*), which can be written as follows:

$$\theta^* = \max_{\pi} \theta(\pi) = \max_{\pi} \left\{ \bar{c}_p^* + \sum_{e \in E_R} \pi_{ij} + \pi_0 \right\}.$$

At each iteration, if the sub-problem is solved to optimality and the minimum reduced cost calculated, the best Lagrangian lower bound (θ^*) is updated if the new Lagrangian bound is better than the actual best Lagrangian lower bound. This update is in constant time and the lower bound is used as an indicator. It must be underlined that, when the column generation procedure terminates, the optimal solution of the linear relaxation of the restricted master problem is equal to the Lagrangian lower bound.

3. Column generation heuristics for the WRPP

A branch-and-price algorithm is proposed to solve the WRPP. The column generation heuristic is described in Section 3.1 and different algorithms to solve the sub-problem are detailed in Sections 3.2 and 3.3. The branching strategy is given in Section 3.4.

3.1. Column generation heuristic

The implementation of the standard column generation procedure for the WRPP is described as follows: starting from an initial set P' of closed-walks, which is initialized by the insertion heuristic described in Section 3.3, the RMP is solved. Then, we search for at most K closed-walks with negative reduced cost. We limit the number of the closed-walks with reduced cost to prevent adding too many columns during the first iterations, where the dual variables are not yet stabilized.

Negative reduced cost columns are first searched by means of a shortest path based heuristic (SPBH) presented in Section 3.2. If the algorithm is unable to find a negative reduced cost closed-walk p, we use the insertion heuristic detailed in Section 3.3. If this method also fails, we solve to optimality the MIP described in Section 2.3 by means of an MIP solver. Generated feasible closed-walks, which have a positive reduced cost, are kept in a pool to be used as candidate columns to be inserted in the RMP. We use the Lagrangian relaxation of the sub-problem to stop the search if the gap is closed with the best closed-walk found so far.

The procedure comes to end when there is no more variable with negative reduced cost to add to the MP.

3.2. Shortest path based approach

We search for a shortest elementary path on $G = (V, E)$. Each required edge $[i, j]$ is weighted by $c_{ij} - \pi_{ij}$ (or $c_{ji} - \pi_{ij}$, according to the direction). Naturally the weight of a non required edge is only its distance. The shortest path based approach will minimize the function $f(i, p)$ over a set of required edges not yet serviced. This function $f(i, p)$ is the reduced cost of the partial (and open) walk p when the dual value π_0 is dropped. To extend a partial walk p ending at node k by a non-incident required edge $[i, j]$, we have:

$$f^*(j, p \cup [k, i] \cup [i, j]) = \min_{[j,i] \notin p} f^*(k, p) + c_{ki} + c_{ij} - \pi_{ij}$$

where

$$f^*(s, \emptyset) = 0 \text{ and } f^*(s, p) = \bar{c}_p + \pi_0.$$

Figure 1 illustrates such extension of a partial walk p.

It is obvious that, if the nodes k and i coincide, the partial walk p is directly extended by a required edge ($[k = i, j]$), and the intermediary non-required edge

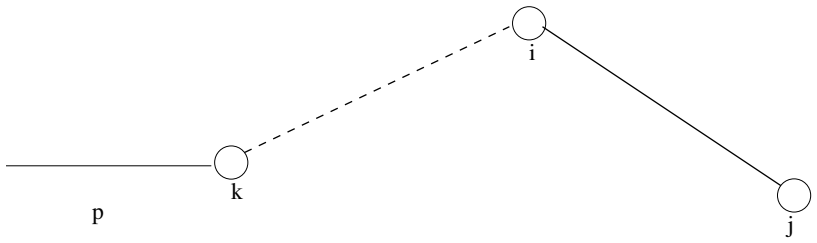


FIGURE 1. Extending a partial walk with a non-incident required edge.

cost disappears ($c_{ki} = c_{kk} = 0$). Then, the expression becomes:

$$f^*(j, p \cup [k, j]) = \min_{[j,i] \notin p} f^*(k, p) + c_{kj} - \pi_{kj}.$$

Dijkstra's label fixing algorithm (Algorithm 1) is used to find the shortest path. Each label (l_v) is initialized to infinity, except the label of the source node; at each iteration, the label with the least cost is fixed and extended to the other nodes.

An alternative to this method is to weight the arcs by only dual values and to search for the longest elementary path.

Algorithm 1. Shortest path based heuristic

$p = \{s\}$; $S_T = V$
for all $v \in S_T$ **do**
  $l_v = \infty$
end for
$l_s = 0$
while $S_T \neq \emptyset$ **do**
  Find $u = \text{arc} \min_{v \in S_t} l_v$
  $S_T = S_T - \{u\}$
  for all $v \in S_T$ **do**
    if $f(v, p \cup \{v\}) \leq l_v$ **then**
      $p = p \cup \{v\}$
      $l_v = f(v, p)$
    end if
  end for
end while
Find $p = \text{arc} \min_{v \in V} \{f(v, p) + c_{vs}\}$

The shortest path based method does not necessarily visit all the required edges. The following insertion algorithm is used to give a complete and feasible solution to the WRPP.

3.3. Insertion heuristic

The required edges are listed in a decreasing order of their dual values. Then every edge in the list is inserted at the best position. Since the cost of an edge depends on the direction, during the insertion two directions of the edge are compared for every possible insertion point. The best insertion position is the one that causes the least increase in the reduced cost. Let us assume that edge $[i, j]$ is inserted, in the position α, between edges $[k, l]$ and $[m, n]$. Reduced cost increase ($\Delta(\bar{c})$) is calculated as follows:

$$\Delta(\bar{c}(\alpha)) = \min(c_{li} + c_{ij} + c_{jm} - \pi_{ij},\ c_{lj} + c_{ji} + c_{im} - \pi_{ij}).$$

If there is not an edge between nodes l and i, c_{li} is the distance of the shortest path between them. After inserting all required edges, the closed-walk is locally improved by reversing or swapping edges.

The insertion algorithm (2) starts with a partial walk p which has only the source node and at each iteration, the edge with the highest dual value, which is not visited by p, is inserted into the best position (α).

Algorithm 2. Insertion heuristic

```
Sort edges in a decreasing order of dual values (L_T)
p = {s}
for ∀e ∈ L_T do
   Find min_α Δ(c̄)
   Insert e in the position α to p
end for
```

3.4. Branching strategy

Column generation is applied at every node of the tree generated by the branch-and-price algorithm. We branch when the flow over a non-required edge is not an integer, *i.e.*, $z_{ij} = \xi$ where $\xi \notin \mathbb{N}$. In that case two branches are created, with $z_{ij} \geq \lceil \xi \rceil$ and $z_{ij} \leq \lfloor \xi \rfloor$. These two constraints are easily added to the MIP of the sub-problem. On the other hand, such a constraint is almost impossible to verify with the insertion heuristic and the shortest path based approach. That is why, these two heuristic approaches are only used at the root node of the branch-and-price tree.

4. Experimental results

We tested our algorithm on the instances of Christofides *et al.* [4] and the instances generated from Hertz *et al.* [10]. The RPP instances of Hertz *et al.* [11] were transformed in WRPP instances by Benavent *et al.* in [1]. The instances of

TABLE 1. Data properties and execution mean times.

Instance set	$\|V\|$	$\|E_R\|$	$\|E_{NR}\|$	$\|E\|$	$\|CC\|$	Time (s)	Time (s) (Benavent *et al.*)
HG1	60	41	64	105	20	2.0	–
HG2	68	49	70	119	21	4.0	–
HG3	64	44	72	116	21	5.0	–
HG4	82	73	75	148	14	3.0	–
HG5	92	77	88	165	19	50.3	–
HG6	91	82	80	162	12	3.0	–
HG7	97	113	61	174	5	4.7	–
HG8	100	107	73	180	10	6.0	–
HG9	97	109	66	175	7	4.9	–
C01	11	7	6	13	5	0.2	2.2
C02	14	12	21	33	5	0.4	2.5
C03	28	26	31	57	5	0.5	3.4
C04	17	22	13	35	4	0.4	2.5
C05	20	16	19	35	6	0.3	3.1
C06	24	20	26	46	8	0.4	3.5
C07	23	24	23	47	4	0.3	2.0
C08	17	24	16	40	3	0.3	2.0
C09	14	14	12	26	4	0.3	2.7
C10	12	10	10	20	5	0.2	2.5
C11	9	7	7	14	4	0.2	1.7
C12	7	5	13	18	4	0.2	1.8
C13	7	4	6	10	4	0.1	1.4
C14	28	31	48	79	7	1.0	3.8
C15	26	19	18	37	9	0.4	3.1
C16	31	34	60	94	8	1.0	4.6
C17	19	17	27	44	6	0.4	2.1
C18	23	16	21	37	9	0.5	2.5
C19	33	29	25	54	8	1.0	5.5
C20	50	63	35	98	8	5.4	5.6
C21	49	67	43	110	7	3.4	6.5
C22	50	74	110	184	7	6.4	5.2
C23	50	78	80	158	7	4.0	6.6
C24	41	55	70	125	8	1.9	3.6

Christofides *et al.* are small-size instances, whereas modified Hertz *et al.* instances are large.

The proposed method is written in the programming language JAVA. CPLEX 11.1 is used as to solve LP and MIP. Experiments are conducted on an Intel dual core 3.0 GHz with 3 GB of RAM.

Each data set consists of 6 instances and the detail of each data set is given in Table 1. The table gives the number of nodes $|V|$, of required edges $|E_R|$, of unrequired edges $|E_{NR}|$, and of all edges $|E|$). The column $|CC|$ represents the number of connected components of the sub-graph induced by required edges. On the last two columns, we give the average resolution mean time of each instance

TABLE 2. Scatter search performance of Benavent *et al.*

Type of instance set	Best mean gap	Worst mean gap	Mean execution time (s)
C	0.27%	0.43%	0.68
HG	0.69%	1.11%	2.09

family obtained by our branch-and-price algorithm and by the branch-and-cut method of Benavent *et al.* [2][5].

Almost all of the instances except three are solved to optimality by the proposed branch-and-price algorithm. These three instances are in HG5, HG9, and C21. The sub-problem for these instances cannot be solved in a reasonable amount of time. Therefore even a Lagrangian lower bound cannot be computed to give a gap and they are excluded from the calculation of the average solution times of HG5, HG9, and C21.

It is interesting to see that the data set with the greatest execution time does not correspond to the largest instances. Actually, five out of six instances in each set need an execution time of less than 5 s. One instance in each set takes more than 200 s. It should be noted that the execution time of the method varies almost 50 times on two different instances that have exactly the same number of nodes and same connections. The only difference between two instances of the same family is the cost of the edges.

One of the best methods found in the literature on WRPPs is the scatter search of Benavent *et al.* [1]. The best and worst mean gaps between their solution and the optimal solution, as well as the mean execution time following the parameters are given in Table 2 on the instances of Christofides *et al.*[4] (C) and the instances generated from Hertz *et al.* [10] (HG).

Our exact method obtains the optimal solution, in less than 5 s on the instances up to 180 edges except one instance which takes a little more than 200 s. All the instances are solved on the root node of the branch-and-price tree.

Another interesting point is that, the branch-and-price algorithm is much less sensitive to the number of connected components ($|CC|$) than the number of required edges.

5. CONCLUSIONS

This paper proposes a branch-and-price algorithm for solving the windy rural postman problem. We find the optimal solution for all the instances tested except three. Another advantage is that the column generation procedure is very fast, for the instances up to 100 nodes and 180 edges of which more than 100 are required.

An interesting future work would be to understand why the proposed method cannot give any result for three instances by observing in detail the features of these instances. Discovering the structure of the *difficult* instances can help us to

[5]On a Pentium III, clocked at 1.0 GHz.

identify and to modify other instances which could be transformed into *easy* ones. This can led us to solve other and larger WRPP instances, which we cannot solve in a reasonable amount of time, until now.

Another idea is to find valid inequalities for the sub-problem and accelerate the solution of the mixed integer program. The resulting branch-and-cut-and-price algorithm would be more efficient.

Acknowledgements. This research was partly supported by Région Midi-Pyrénées and the authors gratefully acknowledge the support of this institution. Thanks are also due to the Editor and to the referee for his valuable comments.

References

[1] E. Benavent, A. Corberán, E. Piñana, I. Plana and J.M. Sanchis, New heuristic algorithms for the windy rural postman problem. *Comput. Oper. Res.* **32** (2005) 3111–3128.
[2] E. Benavent, A. Carotta, A. Corberán, J.M. Sanchis and D. Vigo, Lower bounds and heuristics for the windy rural postman problem. *Eur. J. Oper. Res.* **176** (2007) 855–869.
[3] N. Christofides, V. Campos, A. Corberán and E. Mota, *An algorithm for the rural postman problem.* Technical Report 5, Imperial College Report ICOR815 (1981).
[4] N. Christofides, V. Campos, A. Corberán and E. Mota, An algorithm for the rural postman problem on a directed graph. *Math. Program. Stud.* **26** (1986) 155–166.
[5] A. Corberán and J.M. Sanchis, A polyhedral approach to the rural postman problem. *Eur. J. Oper. Res.* **79** (1994) 95–114.
[6] A. Corberán, A. Letchford and J.M. Sanchis, A cutting plane algorithm for the general routing problem. *Math. Program.* **90** (2001) 291–316.
[7] G.B. Dantzig and P. Wolfe, Decomposition principle for linear programs. *Oper. Res.* **8** (1960) 101–111.
[8] G. Ghiani and G. Laporte, A branch and cut algorithm for the undirected rural postman problem. *Math. Program.* **87** (2000) 467–481.
[9] M. Grötschel and Z. Win, A cutting plane algorithm for the windy postman problem. *Math. Program.* **55** (1992) 339–358.
[10] A. Hertz, G. Laporte and P. Nanchen, Improvement procedures for the undirected rural postman problem. *Informs J. Comput.* **11** (1999) 53–62.
[11] A. Hertz, G. Laport and M. Mittaz, A tabu search heuristic for the capacitated arc routing problem. *Oper. Res.* **48** (2000) 129–135.
[12] J.K. Lenstra and A.H.G. Rinnooy Kan, On general routing problems. *Networks* **6** (1976) 273–280.
[13] A.N. Letchford, *Polyhedral results for some constrained arc-routing problems.* Ph.D. thesis, Lancaster University (1996).
[14] E. Minieka, The chinese postman problem for mixed networks. *Manage. Sci.* **25** (1979) 643–648.
[15] C.S. Orloff, A fundamental problem in vehicle routing. *Networks* **4** (1974) 35–64.
[16] J.M. Sanchis, *El poliedro del problema del cartero rural.* Ph.D. thesis, University of Valencia (1990) (in Spanish).
[17] Z. Win, On the windy postman problem on eulerian graphs. *Math. Program.* **44** (1989) 97–112.

J Math Model Algor (2008) 7:177–195
DOI 10.1007/s10852-008-9080-2

Multi-objective Meta-heuristics for the Traveling Salesman Problem with Profits

Nicolas Jozefowiez · Fred Glover · Manuel Laguna

Received: 1 March 2007 / Accepted: 21 December 2007 /
Published online: 23 February 2008

Abstract We introduce and test a new approach for the bi-objective routing problem known as the traveling salesman problem with profits. This problem deals with the optimization of two conflicting objectives: the minimization of the tour length and the maximization of the collected profits. This problem has been studied in the form of a single objective problem, where either the two objectives have been combined or one of the objectives has been treated as a constraint. The purpose of our study is to find solutions to this problem using the notion of Pareto optimality, i.e. by searching for efficient solutions and constructing an efficient frontier. We have developed an ejection chain local search and combined it with a multi-objective evolutionary algorithm which is used to generate diversified starting solutions in the objective space. We apply our hybrid meta-heuristic to synthetic data sets and demonstrate its effectiveness by comparing our results with a procedure that employs one of the best single-objective approaches.

Keywords Routing problem · Multi-objective optimization · Ejection chain · Evolutionary algorithm · Hybrid method

Mathematics Subject Classifications (2000) 90B06 · 90C27 · 90C29 · 68T20

N. Jozefowiez ( )
INSA, LAAS-CNRS, Université de Toulouse, 7 Avenue du Colonel Roche,
F-31077 Toulouse cedex 4, France
e-mail: nicolas.jozefowiez@laas.fr

F. Glover · M. Laguna
Leeds School of Business, University of Colorado at Boulder, CO, USA

F. Glover
e-mail: fred.glover@colorado.edu

M. Laguna
e-mail: manuel.laguna@colorado.edu

1 Introduction

We investigate the solution of the traveling salesman problem with profits (TSPP) [7] by means of an ejection chain procedure combined with a multi-objective evolutionary algorithm. The TSPP is a generalization of the traveling salesman problem (TSP) where a profit is realized when a customer is visited. The problem can be described as follows. Let $G = (V, E)$ be a graph where $V = \{v_1, \ldots, v_n\}$ is a set of n nodes and E is a set of edges. We associate a profit p_i with each node $v_i \in V$ (with $p_1 = 0$) and a distance c_{ij} with each edge $(v_i, v_j) \in E$. The TSPP consists in determining a tour on a subset of V that includes v_1. Performance is measured both in terms of the collected profit and the tour length, creating two conflicting objectives: (1) to minimize the length of the tour; (2) to maximize the total profit. From the perspective of single-objective optimization, three associated problems have been addressed in the literature:

1. Both objectives are combined in the objective function and the goal is to find a solution that minimizes the tour length minus the collected profit. Dell'Amico et al. [5] refer to this version of the TSPP as the profitable tour problem (PTP).
2. A maximum allowed tour length $c_{\max}$ is imposed as a bound and the goal is to find a tour that maximizes the total collected profit subject to satisfy this bound. This problem is called the orienteering problem (OP). The OP has also been referred to as the selective traveling salesman problem (STSP) [18] and the maximum collection problem [15].
3. A minimum allowed profit $p_{\min}$ is imposed as a bound and the goal is to find a minimal length tour whose total collected profit is not smaller than this bound. This problem is called the prize-collection traveling salesman problem (PCTSP) [2] or the quota traveling salesman problem (QTSP) [1].

More information about these problems (formulations, methods, and applications) can be found in the survey by Feillet et al. [7]. It is important to note that, while these variants of the TSPP are multi-objective in nature, they have never been studied from a multi-objective point of view [3]. An attempt to address the TSPP in its explicitly multi-objective form was made by Keller [16] and Keller and Goodchild [17], who referred to the problem as the multi-objective vending problem, but their approaches consisted of sequentially solving single-objective versions of the problem and they did not try to generate a set of non-dominated solutions.

In its general form, a multi-objective problem can be stated as follows:

$$(MOP) = \begin{cases} \min \; F(x) = (f_1(x), f_2(x), \ldots, f_n(x)) \\ s.t.\; x \in D \end{cases} \tag{1}$$

with $n \geq 2$ being the number of objective functions; $x = (x_1, x_2, \ldots, x_r)$, the decision variable vector; D, the feasible solution space; and $F(x)$, the objective vector. The set $O = F(D)$ corresponds to the feasible solutions in the objective space, and

$y = (y_1, y_2, \ldots, y_n)$, where $y_i = f_i(x)$, is a solution. A MOP solution is the set of the non-dominated solutions called the Pareto set (PS). Dominance is defined as follows:

Definition 1.1 A solution $y = (y_1, y_2, \ldots, y_n)$ dominates (denoted $\prec$) a solution $z = (z_1, z_2, \ldots, z_n)$ if and only if $\forall\, i \in \{1 \ldots n\}$, $y_i \leq z_i$ and $\exists\, i \in \{1 \ldots n\}$, such that $y_i < z_i$.

Definition 1.2 A solution y found by an algorithm A is said to be potentially Pareto optimal (PPS), relative to A, if A does not find a solution z, such that z dominates y.

Evolutionary algorithms and local search methods have been proposed to approximate PS [6]. Such heuristics must be designed with two goals in mind: (2) the algorithm must converge toward the PS, and (2) the solutions identified as efficient must provide a good coverage of the frontier.

Our goal is to apply multi-objective optimization techniques to the TSPP. The main contribution of this paper is the development of a hybrid meta-heuristic (HM) that finds high-quality approximations of the efficient frontier for this class of bi-objective problems. This approach provides a means for addressing all the single objective problems mentioned above and avoids a priori parameterization of the TSPP. The first step of the design of the HM was the definition of a neighborhood search process. To do that, two difficulties in solving the TSPP must be overcome: (1) to solve many traveling salesman problems on different sets of nodes, (2) to select different subsets of nodes to be visited. This is solved by using two sets of neighborhood moves in an ejection chain (EC) process. To provide starting solutions for the EC process, a multi-objective evolutionary algorithm (MOEA) has been developed. We have chosen a MOEA because it is one of the most studied methods for multi-objective problems and it remains, apart from the specific operators, an easy-to-implement solution. However, other methods or techniques could be chosen instead of the MOEA. The coupling of the MOEA and the EC process provides a means for both exploring and approaching the optimal Pareto set, as the EC process is an efficient way to bring solutions toward the optimal Pareto set, while the MOEA is able to provide solutions in the complete objective space thanks to its population, and, therefore, to diversify the search.

The EC process is presented in Section 2. The MOEA and the HM are described in Section 3. Experimental testing is reported in Section 4 and conclusions are summarized in Section 5.

2 Ejection Chain Process

The EC process uses two sets of moves. The first set originates from a reference structure initially proposed by Glover for the TSP [11]. These moves allow the search to be efficient in the solution of the TSP aspect of the problem. The reference structure is presented in Section 2.1 and the set of moves in Section 2.2. The second set of moves is used to modify the set of visited nodes as it is the only way to modify the profit generated by the solution. This set is described in Section 2.3. A goal programming strategy is used to guide the search in the bi-objective space. The

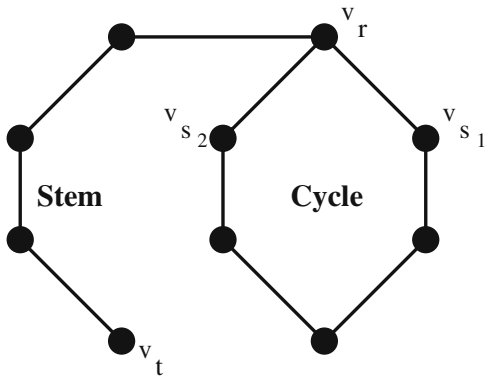


Fig. 1 The stem-and-cycle reference structure

goals are dynamically computed according to the starting solutions as explained in Section 2.4. The improvement procedure for the TSPP, called IP-TSPP, is described in Section 2.5.

2.1 The Stem-and-cycle Reference Structure

Our improvement procedure uses the stem-and-cycle reference structure proposed by Glover [11] for the TSP, which is a spanning sub-graph of G consisting of a path called a stem $ST = \{v_t, \ldots, v_r\}$ connected to a cycle $CY = \{v_r, v_{s_1}, \ldots, v_{s_2}, v_r\}$. An example of such a structure is shown in Fig. 1. Node v_r is common to both the stem and the cycle and is referred to as the *root*. The two nodes (v_{s_1} and v_{s_2}) of the cycle adjacent to v_r are known as *subroots*. Node v_t is the *tip* of the stem.

The structure obtained through the application of an ejection move does not usually represent a feasible tour (unless $v_t = v_r$, i.e. if a unique node is at the same time the root and the tip); thus, a trial move is required to generate a feasible TSP solution. Trial solutions are obtained by inserting an edge (v_t, v_s), where v_s is one of the subroots, and removing the edge (v_r, v_s).

2.2 First Set of Moves

The first set is composed of moves rearranging the sequence in which the nodes are visited. Two different ejection chain moves are possible depending on the positions of the nodes in the *stem-and-cycle* structure. The first move is called a *cycle-ejection* move. An edge of the cycle (v_p, v_q) is removed, v_p is linked to v_t and v_q becomes the new tip. The second move is called a *stem-ejection* move. The move inserts an edge between v_t and v_p where v_p is a node of the stem. Then, the edge (v_q, v_p) is deleted where v_q is the node before v_p on the subpath $(v_t, \ldots, v_p)$. Node v_q becomes the new tip.

It should be noted that a standard TSP tour can be seen as a stem-and-cycle structure where the stem is empty and the cycle is the tour. The root can be any node in the tour.

2.3 Second Set of Moves

The second set is composed of three moves which correspond to the basic operations that can be done in order to modify the set of visited nodes and the profit. These moves work as follows.

1. The first move removes a node visited in the solution. If the node is the tip, the next node in the stem becomes the new tip. Otherwise, the nodes before and after the removed node are connected together to repair the solution. In this move, v_1 and v_r cannot be removed.
2. The second move adds a node that is not visited in the solution. The node is inserted so that the increase in length is minimal. In this move, it is possible to add a node as a new tip.
3. The third move exchanges a node currently in the tour with a node outside the tour. If the removed node is the tip, then the added node becomes the tip. In this move, v_1 or v_r cannot be exchanged.

2.4 Construction of the Goal Point

A goal point is used to guide the search in the multi-objective space. The choice of the goal point is important as it should attract the search toward the section of the Pareto frontier that dominates the solution that is the starting point of the local search.

To construct the goal point, we start by calculating the upper bound on the total profit associated with the m most profitable nodes that are not currently part of the solution. In our implementation, m is fixed to $\lfloor \frac{n}{10} \rfloor$ (where n is the number of nodes). This is done to initially favor moves that result in solutions that do not differ much (in terms of the subset of visited nodes) from the starting solutions. The tabu restrictions (discussed below) diversify the subset of visited nodes that are associated with the solutions in the late stages of the search.

The difficulty with creating a goal in terms of tour length is that the impact of removing m nodes from the starting solution is not really known, because the tour will change. Therefore, the goal for the length objective is computed as follows. Let g_p be the goal value for the profit, s_l the length of the starting solution, s_p the profit of the starting solution, ub_p the maximum profit (i.e. the profit obtained when all the nodes are visited), and ub_l the length upper bound (i.e. the worst length in the archive A containing all the non-dominated solutions found so far). Then, the goal value for the length g_l is given by $g_l = s_l - \text{ub}_l \times \frac{g_p - s_p}{\text{ub}_p}$. In this way, the distance between the starting solution length and g_l is the same as the one between the starting solution profit and g_p if the values are normalized.

2.5 Improvement Procedure for the TSPP (IP-TSPP)

The improvement procedure that we have developed is similar to the PSEC LS proposed by Rego [19] for the TSP. A high level description of the improvement procedure is shown in Algorithm 1. IP-TSPP takes as an input a solution S for the TSPP. First, it computes the goal point according to S. At each iteration, itselects a new node to become the root node of the stem-and-cyle procedure. This is necessary as the explored neighborhood highly depends on the definition of the root node. The process is iterated until all the nodes have been used as the root without finding a

solution closer to the goal point. When a better solution is found, the set of possible root nodes is reset so that every node can be chosen again as a root. A solution s_1 is said to be better than another solution s_2 if the Euclidean distance between $F(s_1)$ and g is smaller than the Euclidean distance between $F(s_2)$ and g. The search also updates an archive A which contains the non-dominated solutions found during the search. This archive is the final result of IP-TSPP. It allows to find solutions obtained thanks to an oscillation around the local optimum found by the procedure.

Algorithm 1 IP-TSPP(Solution S)

Compute the goal point g according to S.
$S^* \leftarrow S$
$C \leftarrow V$ (C is the set of candidates to become the root node)
$A \leftarrow \emptyset$ (A is the archive of the non-dominated solutions found)
while $C \neq \emptyset$ **do**
  Select v_r randomly from C and set v_r as the root node
  $C \leftarrow C \setminus \{v_r\}$
  $S' \leftarrow$ core_step(S^*, g, v_r, A)
  Apply a trial move on S' (i.e. build a feasible tour as explained in Section 2.1)
  if S' is better (i.e, closer to g) than S^* **then**
    $S^* \leftarrow S'$
    $C \leftarrow V$
  end if
end while
Return A

The core step of IP-TSPP is detailed in Algorithm 2. First, the stem-and-cycle structure is completely defined by making the root node also the tip node. The starting solution is saved as the best solution found so far and as the current solution.

Algorithm 2 core_step (Solution S, Goal g, Root v_r, Archive A)

$v_t \leftarrow v_r$ (Initially, the root is also the tip)
$k \leftarrow 1$
$S^* \leftarrow S$
$S_c \leftarrow S$
repeat
  Find the best feasible move on S_c, i.e., the moves allowing the smallest Euclidean distance between the resulting solution and g
  Perform the move on S_c
  Update the tabu short term memory structure
  Try to include S_c in A
  if S_c is better than S^* **then**
    $S^* \leftarrow S_c$
  end if
  $k \leftarrow k + 1$
until $k > k_{\max}$
Return S^*

Then, the process is run for $k_{\max}$ iterations or until no move is possible. $k_{\max}$ is fixed to n. The move to perform is chosen first by considering the first set of moves and then the second set of moves. The chosen move is the one which provides the closest point in the objective space to the goal point when it is combined with a trial move. In case of a tie, the move from the first set is preferred, otherwise the choice is made randomly. During the move selection phase, the move is not performed as it is not necessary to do so to know the values of the objectives. The selected move operates on the current solution Sc and the tabu memory structure is updated. A short term memory is used to record the attribute of the (θ) most recent moves. For the first set of moves, we record the edge that was removed or inserted in order to prevent the reversal of such a move for (θ) iterations. If the selected move belongs to the second set, we record the node that was removed (added) in order to prevent the node from being added (removed) in the next (θ) iterations. The tabu status are reset between each core step as it has shown to provide better results. The current solution is also tried for inclusion in the archive A. S_c is included in the archive if no solution from the archive dominates it. All solutions from A dominated by S_c are removed. An iteration ends with the updating of the best solution found if the current solution is closer to the goal point.

3 The Multi-objective Evolutionary Algorithm and the Hybrid Meta-heuristic

To generate starting solutions for IP-TSPP, we use a multi-objective evolutionary algorithm (MOEA). The multi-objective evolutionary algorithm we propose for the TSPP is a steady-state variant of NSGA II [4]. The procedure operates on a finite population of solutions (or individuals) that we initialize as described in Section 3.1. Then, the population evolves from one generation to the next. In our MOEA, an iteration consists of the following steps (where the numbers in parentheses indicate the section number where the step is described in greater detail). In our MOEA, an iteration runs as follows:

1. Evaluation (Section 3.2): The *fitness* (quality) of the individuals in the population is evaluated.
2. Parent selection (Section 3.2): A pair of solutions (*parents)* is selected according to their fitness.
3. Crossover (Section 3.4): The pair of parents combines to produce a new solution (*offspring*). We generate only one offspring because we are implementing a steady-state variant of NSGA II.
4. Mutation (Section 3.5): The offspring is randomly modified with a given probability.
5. Population update (Section 3.2): The offspring may replace the worst individual in the current population.

These steps are explained in details below. In Section 3.6, we explain how solutions generated by the MOEA are selected for the IP-TSPP procedure and how the output from the IP-TSPP procedure is merged with the MOEA population. This forms our hybrid meta-heuristic.

3.1 Initialization of the Population

We build the initial population of N individuals by means of heuristically solving several STSPs with the *Insert and Shake* procedure of Gendreau et al. [9]. This method combines the TSP tour extension heuristic described in Rosenkrantz et al. [20] and GENIUS, a TSP heuristic developed by Gendreau et al. [8]. *Insert and Shake* gradually extends a tour T until no other node can be added without violating the length limit $c_{\max}$. Then, GENIUS is applied in an attempt to obtain a shorter tour on the nodes in T. If GENIUS fails to produce a better tour, the procedure terminates. Otherwise, more node insertions are attempted and the process is repeated. Thus, the steps to build an initial population of solutions are:

STEP 1 The IP-TSP local search described in Section 2 is applied to find a tour, considering all nodes in V, and this solution is included in the population. Let ub_l be the length of the tour. Set $c_{\max} \leftarrow \text{ub}_l - \frac{\text{ub}_l}{N-1}$ and $i \leftarrow 1$.

STEP 2 If i is equal to $N+1$, terminate. Otherwise, generate a solution by means of *Insert and Shake* and add this solution to the population.

STEP 3 Set $c_{\max} \leftarrow \frac{c_{\max} - ub_l}{N-1}$ and $i \leftarrow i+1$. Go to STEP 2.

3.2 Population Management

At each generation, NSGA II computes two values for each solution i: a rank r_i, which measures solution quality, and a crowding distance metric d_i, which estimates search diversification. To do that, NSGA II sorts the population into different non-domination levels. In this ranking phase, the non-dominated individuals in the population obtain rank 1 and form the subset E_1. Rank k is given to the solutions only dominated by the individuals belonging to the subset $E_1 \cup E_2 \cup \cdots \cup E_{k-1}$. Then, a fitness equal to its rank (1 is the best level) is assigned to each solution.

The *crowding distance metric* gives an estimate of the density of the solutions surrounding a solution i in the population and is expressed by approximating the perimeter of the cuboid formed by the nearest neighbors of i.

Two parents are selected by means of a tournament that favors the best ranked solutions and uses the crowding distance to break ties. One offspring is generated from the selected parents using the genetic operators described below. Solutions may not appear more than once in the population, following an orientation introduced in scatter search [10, 13]. This means that a recently created offspring is added to the current population if and only if it does not already exist. A new offspring replaces the solution in the current population that has the worst rank, with the crowding distance serving as the tie-breaking mechanism. The population does not change if the newly created offspring is already in the population.

3.3 Archive and Stopping Criterion

The potentially Pareto optimal solutions are stored in an archive A. In the implementation, the archive is common with IP-TSPP. At the end of the process, this archive contains our solution to the problem. This archive is also used to implement a rule that makes the search stop if the archive does not change for M generations in a row.

3.4 The Combination Operator

The main difficulty associated with designing a method to combine solutions for the TSPP (loosely referred to as *crossover* in GA parlance) is that two parent solutions may not have much in common. For instance, when considering the set of nodes visited in each parent solution, their intersection may consist of only v_1 and their cardinality may be different. The following multi-phase combination operator was designed taking these observations into account. It works by transforming the parents into parents visiting the same set of nodes such that a classic operator for the TSP can be applied. More precisely, in the first phase, the parents are modified by eliminating all nodes that are not common to both parents. Then the edge recombination crossover (ERX) [21] is applied to the modified parents. Finally, nodes that have been discarded during the first phase are tested for inclusion in the offspring. This constitutes a simple instance of the structured combination approach proposed in connection with scatter search and tabu search [12, 14].

3.4.1 Phase 1

Let s_1 and s_2 be the two parents, where s_1 (respectively s_2) visits the nodes $V_1 \subseteq V$ (respectively $V_2 \subseteq V$) following the tour σ_1 (respectively σ_2). Let $V' = V_1 \cap V_2$, identifying the set of nodes that are common to s_1 and s_2. If $|V'| = 1$, the crossover is aborted and no offspring is generated. Otherwise, we build a solution s'_1 from s_1 by keeping only the nodes in V' and building a tour σ'_1 that preserves the visiting order in σ_1. For every node pair v_i, v_j such that v_j is visited just after v_i in σ'_1, two arcs (v_i, v_j) and (v_j, v_i) are defined. Arc (v_i, v_j) receives two values: (1) the length of the path from v_i to v_j in σ_1; (2) the sum of the profits of the nodes on the path between v_i and v_j in σ_1, excluding v_i. (v_j, v_i) receives similar values by interchanging the roles of v_i and v_j. It should be noted that the arcs (v_i, v_j) and (v_j, v_i) have the same length but not the same profit as they stand for the same path but traverse in two different orders and that the profit of the starting node is not considered in the profit associated to the arc. A solution s'_2 is built in the same way from s_2. An example of this process is provided in Fig. 2, where, in the left-hand side graphs, the values next to the nodes are profit and the values next to the edges are length. In the right-hand side graphs, the values on the arcs are length/profit.

3.4.2 Phase 2

During this phase, a modified form of ERX is applied, with s'_1 and s'_2 as parents. For each node in $v_i \in V'$, a list l_i of the nodes adjacent to v_i in s'_1 and s'_2 is built. A node v_j can appear twice if the arcs (v_i, v_j) and/or (v_j, v_i) appear in both parents. (The direction of the arc is also saved.)

The offspring is created by adding one node at a time beginning with v_1. Let v_i be the last node added to the offspring. Then, v_i is removed from every list where it appears. If the list l_i associated with v_i is empty, the structured combination process stops. We point out that this operator may not add all the nodes that appear in the parent solutions. However, in practice, it has been observed that ERX builds a solution that includes all the parent nodes about 95 % of the time [21]. When ERX fails to generate a solution that includes all nodes in the parents, a node that is not

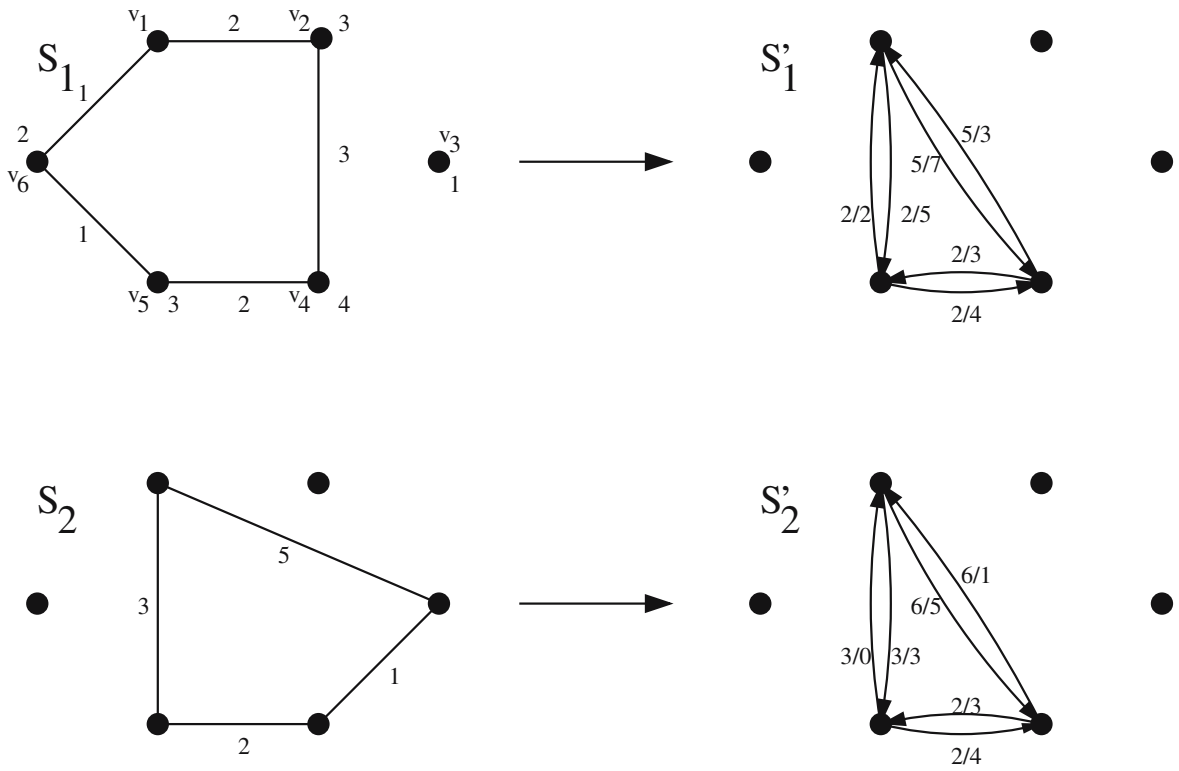


Fig. 2 An example of the phase 1 of the crossover

already in the offspring is randomly chosen and the process continues. In the context of the TSPP, we require no repairing mechanism because a solution is feasible even if all nodes are not included.

When l_i is not empty, the following rules build the set of candidates from which the next node is randomly selected:

1. Let C_1 be the subset of nodes v_j in l_i whose associated list l_j is not empty. If the l_j list associated with every node v_j in l_i is empty, then let C_1 consist of all v_j in l_i.
2. For every node v_j in C_1, let a_j be the arc between v_j and v_i, and assign node v_j a rank equal to the number of nodes v_k such that the values of a_j are dominated in the Pareto sense by a_k. Let C_2 be the subset of nodes in C_1 having rank 0.
3. Let C_3 be the subset of C_2 nodes whose associated lists tie for having the least number of elements.
4. Let C_4 be the subset of nodes v_j in C_3 such that (v_i, v_j) and/or (v_j, v_i) appears in both parents s'_1 and s'_2. If such a node v_j in C_3 does not exist, let $C_4 = C_3$.
5. Choose a node randomly from C_4.

The first step favors the adjacent nodes with available neighbors such that the tour can be extended after its application. The second step is a greedy criterion such that the current offspring at this moment is not dominated by another offspring of the same length. Experiments have shown that this choice has a positive impact. Step 3 is the standard criterion from ERX [21]. Step 4 is to encourage the choice of information present in both parents. For instance, if we have the choice to include two nodes v_j and v_k after v_i. If the arc (v_i, v_j) appears in both parents while (v_i, v_k) only appears in one parent, the former is selected.

3.4.3 Phase 3

A node v_j is added immediately after a node v_i during the second phase when either arc (v_i, v_j) or arc (v_j, v_i) appears in one of the parents s'_1 and s'_2. Let α be the added arc, s' be the modified parent where α appears and s be the original parent that was transformed into s' during phase 1. According to the construction process used in phase 1, α may stand for a path composed of several nodes in s. The nodes in such a path are added to the offspring while preserving the direction of the path between v_i and v_j in s. This is done for all the edges (v_i, v_j) in the offspring. For instance, if we consider the first parent in Fig. 2 and if the arc (v_1, v_5) has been added to the offspring from this parent, then, in phase 3, it is extended to the path $v_1 v_6 v_5$.

3.5 Mutation and Improvement Operators

After an offspring is generated, we allow it to be modified with probability P_m. There are three methods to achieve this:

Add/Remove	A node is randomly selected from V. If the node appears in the tour, it is removed and its predecessor and successor are connected to repair the tour. Otherwise, the node is added so that the length of the resulting tour is minimal.
Exchange	A node in the tour is randomly selected and replaced by another randomly selected node that is currently not in the tour.
Local search	An EC process is applied to the solution. It is the same process as IP-TSPP except that the possible moves are limited to the first set and the trial move. The goal point is defined as $(0, p)$ where p is the profit of the solution to be mutated. In that way, this operator may improve the length of the tour without modifying the set of visited nodes (i.e., with no change in the total profit).

These operators have been selected because they closely relate to the four main operations that may be used to transform a tour in the context of the TSPP.

3.6 Definition of the Hybrid Meta-heuristic

IP-TSPP is applied to every generated solution s that enters the archive A during the GA process, that is, to every solution that appears to be of good quality and promising (i.e. solutions which are potentially Pareto optimal at a given time). Then, if such a solution exists at the end of IP-TSPP, s is replaced in P by one of the solutions that dominates s found during the execution of IP-TSPP.

Then, the complete method works as follows (cf. Fig. 3): first, an initial population P is generated by a heuristic for the STSP. An archive A containing all the non-dominated solutions found in P is created. Every generated solution s will be tested for inclusion in A, i.e., if no solution from A dominates s, then s is included in A. At the same time, every solution from A dominated by s is removed from A. Then, the MOEA is run until the number of generations without a solution included in A (*nbstall*) reaches a given number of generations M. A generation works as follows: two parents are selected in the population and genetic operators are applied to obtain an offspring o. This offspring is only considered if it is not already in the population.

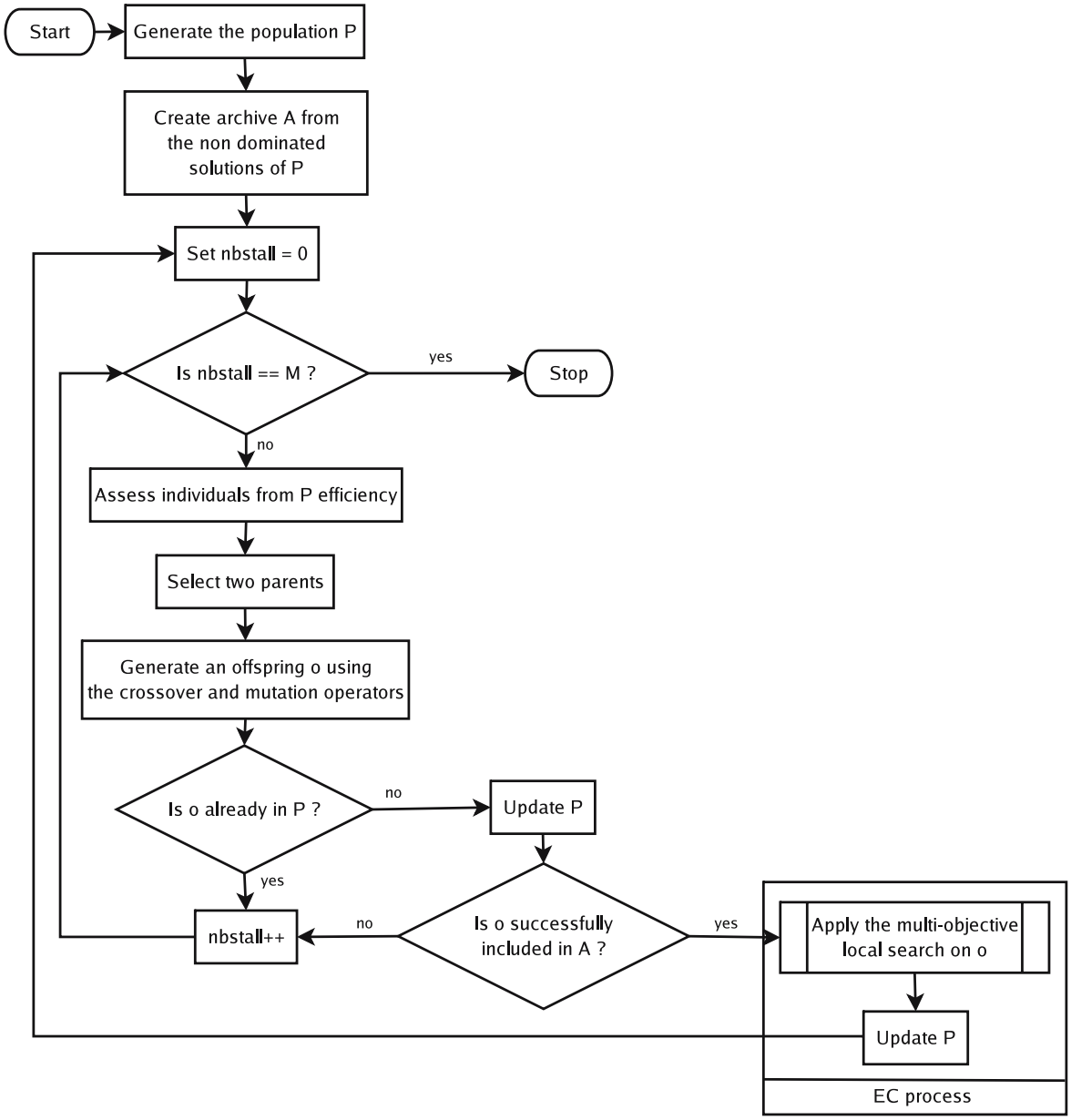


Fig. 3 Flowchart of the hybrid meta-heuristic

If o is successfully added to the archive, the EC process is executed with o as the starting solution. In any case, the worst solution in P is replaced by o or a better solution found by the EC process if it has been used.

4 Computational Results

4.1 An ϵ-constraint Method

As far as we are able to determine, there is no other method in the literature that addresses the TSPP as a bi-objective problem. However, a number of studies have investigated the single-objective variants [7], and we have undertaken to compare our procedure to the Tabu search (TS) for the STSP designed by Gendreau et al. [9]

which is reported to be one of the most efficient procedures for the STSP (see Feillet et al. [7]).

To generate solutions to the TSPP with the TS method of Gendreau et al., we use the following ϵ-constraint approach. In the bi-objective case, the ϵ-constraint method adds a new constraint to the problem: $f_i(x) \leq \epsilon$ (or $f_i(x) \geq \epsilon$), where f_i is an objective to be minimized (respectively maximized) and ϵ is a given value. The single-objective procedure is then asked to optimize the other (unconstrained) objective. Varying the ϵ parameter allows for the exploration of the bi-objective space. In the context of the STSP, the objective to be optimized is the maximization of profit and the constrained objective is the minimization of the length. The following procedure is used to choose ϵ:

STEP 1 Compute $\epsilon < 0$ such that any improvement on any tour is smaller than ϵ. This can be computed by considering the smallest possible improvement in the length. This is computed by considering the different possible moves: a node n is added or removed or exchanged with another node while being between any two nodes or swapped from any two nodes with another node which would be between any two nodes. α is computed once at this step considering all the possibilities.

STEP 2 Solve the TSP (heuristically) on V. Save the resulting solution as a potential Pareto optimal solution. Set $c_{\max} \leftarrow \text{ub}_l + \epsilon$, where ub_l is the length of the solution.

STEP 3 If $c_{\max} < 0$, stop. Solve the STSP by means of the TS. Let s be the resulting solution with a length of s_l. Save s as a potential Pareto optimal solution. Set $c_{\max} \leftarrow s_l + \epsilon$. Reiterate STEP 3.

If the TS method of Gendreau et al. were able to find the optimal solution at each iteration, then the set built during the search would be the optimal Pareto set. Furthermore, the number of executions of their TS method will not exceed the number of solutions in the optimal Pareto set in the parametric we are using.

4.2 Benchmarks and Parameters

Experiments were conducted on a series of randomly-generated instances. To generate the node set, $|V|$ different points were generated in a [0, 100] $\times$[0, 100] square with a uniform distribution. A randomly generated integer profit between 1 and 100 was assigned to each node of $V \setminus \{v_1\}$. For each value of $|V| = 75, 100, 125, 150$, five instances were generated.

The evolutionary algorithm MOEA and the hybrid meta-heuristic HM were executed ten times on each instance. The parameters used were: θ was randomly chosen in [5], $N = 256$, $M = 2{,}500$, and $P_m = 0.2$. Preliminary experimentation disclosed these parameters to be effective.

We have compared the methods according to their computational times in seconds (Time) and the number of potentially Pareto optimal solutions (NB) they were able to find. The $\mathcal{S}$ metric [22] was also used to compare the methods, based on computing the volume (area in the bi-objective case) dominated by a given Pareto-front approximation. The $\mathcal{S}$ metric requires a reference point Z_{ref} consisting in a reference value for each of both objectives. For the length value, we used the worst length found by all methods; for the profit, we took the value 0 as it is the worst

Table 1 Times (in seconds)

n	ϵ-Constraint method	HM		MOEA	
		Mean	SD	Mean	SD
75	5,114	270	33	121	40
75	5,172	388	77	120	32
75	5,610	236	19	107	21
75	4,191	305	40	147	32
75	5,603	379	22	130	25
100	20,373	1,122	132	445	128
100	20,990	1,178	156	272	67
100	17,447	1,415	127	529	185
100	15,754	1,404	350	465	80
100	17,757	1,796	745	489	112
125	39,828	4,613	542	997	237
125	36,770	2,960	434	972	182
125	39,656	4,131	761	1,206	399
125	38,313	3,496	437	876	270
125	32,800	3,687	619	1,277	464
150	55,258	7,330	1,162	2,462	635
150	59,203	8,729	1,975	2,380	494
150	53,381	5,239	670	1,998	426
150	50,395	6,637	1,467	2,999	383
150	54,780	7,224	1,409	3,220	654

possible profit. We have normalized the $\mathcal{S}$ metric so that the maximal theoretical area is 1.0. The results are respectively reported in Tables 1, 2, and 3, for the TS-based ϵ-constraint method, the MOEA alone, and the hybrid meta-heuristic (HM),

Table 2 Number of potentially Pareto optimal solutions

n	ϵ-Constraint method	HM		MOEA	
		Mean	SD	Mean	SD
75	305	645.7	17.3	516.0	39.2
75	435	723.9	42.2	569.5	50.9
75	326	660.2	11.2	539.0	28.4
75	308	669.7	12.8	516.2	33.3
75	380	733.9	10.0	607.2	45.1
100	580	1,142.9	40.6	869.7	37.9
100	444	867.4	13.8	657.0	28.3
100	445	1,271.9	29.0	886.4	121.1
100	481	1,012.2	31.3	891.2	50.9
100	562	1,093.1	31.6	854.7	61.9
125	561	1,752.9	25.1	1,382.1	147.5
125	510	1,531.8	33.8	1,068.9	85.3
125	601	2,005.8	35.5	1,318.6	172.0
125	505	1,381.4	17.8	1,129.6	102.6
125	478	1,692.0	45.1	1,366.3	99.5
150	464	1,864.6	39.0	1,490.9	167.1
150	527	2,344.8	149.4	1,767.6	261.2
150	435	1,671.0	72.4	1,341.8	100.5
150	439	2,028.1	75.6	1,326.5	89.6
150	719	2,283.6	46.8	1,829.0	178.7

Table 3 $\mathcal{S}$ metric values

n	ϵ-CM	HM			MOEA		MOEAu	
		Max.	Mean	Min.	Max.	Mean	Max.	Mean
75	0.585067	0.586240	0.585888	0.585420	0.585468	0.583486	0.585561	0.583986
75	0.593236	0.594540	0.593858	0.590694	0.590843	0.588162	0.591399	0.589119
75	0.576733	0.577810	0.577765	0.577670	0.577333	0.577040	0.577494	0.577299
75	0.596526	0.597674	0.597607	0.597486	0.596054	0.595351	0.596182	0.595912
75	0.581920	0.583361	0.583007	0.582194	0.582334	0.580350	0.582417	0.581113
100	0.606018	0.607229	0.606574	0.605043	0.605763	0.603865	0.606657	0.605292
100	0.591636	0.592637	0.592344	0.592243	0.591934	0.591287	0.592143	0.591838
100	0.611364	0.612885	0.611928	0.611251	0.610975	0.608407	0.611155	0.609311
100	0.595497	0.597554	0.597039	0.596057	0.594981	0.594168	0.596677	0.595223
100	0.594143	0.595550	0.594759	0.593889	0.593215	0.589779	0.592664	0.590929
125	0.587292	0.590175	0.589944	0.588946	0.587899	0.586793	0.588643	0.587425
125	0.595876	0.598160	0.597566	0.596947	0.595562	0.594876	0.596203	0.595445
125	0.630162	0.633343	0.632389	0.629455	0.631595	0.629748	0.631395	0.630464
125	0.600385	0.603044	0.602341	0.599631	0.601820	0.600550	0.602253	0.601144
125	0.614249	0.617887	0.616088	0.614819	0.614379	0.612510	0.614921	0.613707
150	0.587704	0.595406	0.594031	0.591555	0.592332	0.588543	0.591752	0.587566
150	0.611467	0.620409	0.619362	0.614182	0.617709	0.613773	0.616726	0.614095
150	0.614186	0.617686	0.616531	0.615642	0.614569	0.611352	0.613949	0.611781
150	0.609805	0.618804	0.618209	0.617180	0..616121	0.612857	0.616281	0.614079
150	0.629149	0.632400	0.631189	0.629091	0.631050	0.629373	0.631292	0.629490

i.e. IP-TSPP with the MOEA providing the starting solutions. We have also used the $\mathcal{C}$ metric [22] to compare the average ratio of solutions found by a given method dominated by solutions from the hybrid meta-heuristic. In Table 4, C(A, B) is the average ratio of solutions from B dominated by solutions from A.

4.3 Efficiency of the MOEA

Because the genetic algorithm literature proposes that a solution recombination method should constitute a solution procedure in itself, and not simply as an intensification or diversification process for a local search, we investigated the MOEA procedure in isolation from the local search procedure. The efficiency of the MOEA was assessed in comparison with the TS-based ϵ-constraint method. According to the $\mathcal{S}$ metric, the ϵ-constraint method was able to find better quality approximation than the MOEA for most problem instances of size smaller or equal to 125. However, the MOEA has a better average $\mathcal{S}$ value than the $\mathcal{S}$ value for the ϵ-constraint method for 4 out of 5 instances when $|V| = 150$. Furthermore, for 11 instances, i.e. more than fifty percent of the instances, there is at least one run during which the MOEA found a better approximation than the ϵ-constraint method. This seems to indicate that the MOEA is still an interesting method. Indeed, the running times of the MOEA remain particularly low compared to the TS-based ϵ-constraint method, and therefore, it can be used to obtain quickly a first approximation to the problem.

4.4 Efficiency of the Hybrid Meta-heuristic

As it can be expected, the full HM procedure which incorporates the MOEA as a process to generate starting solutions considerably increases the computational

Table 4 $\mathcal{C}$ metric values

n	HM/ϵ-CM		HM/MOEA		HM/MOEAu	
	C(HM,ϵ-CM)	C(ϵ-CM,HM)	C(HM,MOEA)	C(MOEA,HM)	C(HM,MOEAu)	C(MOEAu,HM)
75	0.26	0.03	0.42	0.04	0.40	0.04
75	0.33	0.02	0.59	0.02	0.55	0.03
75	0.32	0.01	0.38	0.02	0.28	0.02
75	0.39	0.01	0.50	0.01	0.46	0.01
75	0.40	0.06	0.56	0.04	0.49	0.04
100	0.39	0.08	0.67	0.07	0.50	0.07
100	0.34	0.01	0.41	0.01	0.30	0.01
100	0.48	0.02	0.67	0.08	0.56	0.08
100	0.58	0.07	0.67	0.02	0.57	0.07
100	0.34	0.03	0.71	0.03	0.61	0.03
125	0.63	0.01	0.71	0.01	0.63	0.01
125	0.60	0.06	0.76	0.03	0.70	0.03
125	0.61	0.05	0.72	0.01	0.64	0.02
125	0.67	0.09	0.60	0.09	0.48	0.09
125	0.48	0.03	0.73	0.06	0.67	0.06
150	0.70	0.04	0.79	0.05	0.79	0.06
150	0.70	0.02	0.80	0.03	0.78	0.02
150	0.52	0.03	0.84	0.05	0.81	0.04
150	0.78	0.01	0.83	0.02	0.78	0.07
150	0.39	0.14	0.58	0.06	0.53	0.07

time. However, as shown in Table 1, the time remains significantly smaller than the time needed by the ϵ-constraint method. From a positive point of view, IP-TSPP significantly improves the quality of the solutions. It increases the number of potentially Pareto optimal solutions found by the HM when it is compared to the MOEA. It should be noted that the average $\mathcal{S}$ value for the HM is always better than the best value for the MOEA. Moreover, the worst $\mathcal{S}$ metric value for the HM is better on 17 instances than the average $\mathcal{S}$ value for the MOEA and it is better on 10 instances than the best $\mathcal{S}$ value for the MOEA.

Another test we have done is to compare the HM with the MOEA if the latter was allowed to run for a time equal to the average of the times of the HM on each instance. The results for the $\mathcal{S}$ metric for ten runs on each instance is reported in Table 3 and the new implementation of the MOEA is denoted MOEAu. It appears that the average $\mathcal{S}$ metric value is better for the HM for all the instances and this value is also better than the one for the best run of MOEAu on 19 instances out of 20. The worst $\mathcal{S}$ metric value for the HM is better on 16 instances than the average $\mathcal{S}$ value for MOEAu and it is better on 9 instances than the best $\mathcal{S}$ value for MOEAu. Therefore, it appears that even if we let the MOEA run longer, the HM is still a better choice as it can provide better solutions on average and, in general, it seems it would be able to provide better solutions.

When compared to the ϵ-constraint method, the HM also performs well. As disclosed by Table 1, it is an order of magnitude faster than the Tabu-based ϵ-constraint approach (on average 9.2 times faster). It is also able to find many more potentially Pareto optimal solutions, which can offer more possibilities to the decision maker. Even if the number of solutions found may not be able to allow to determine which method is better, the fact that the ϵ-constraint method found fewer solutions employing significantly more computer time indicates the limitations of

Table 5 Kruskal–Wallis statistical test results

n	HM vs. ϵ-CM	HM vs. MOEAu	ϵ-CM vs. MOEAu
75	$\prec$	$\prec$	$\equiv$
75	$\prec$	$\prec$	$\succ$
75	$\prec$	$\prec$	$\prec$
75	$\prec$	$\prec$	$\succ$
75	$\prec$	$\prec$	$\equiv$
100	$\equiv$	$\prec$	$\prec$
100	$\prec$	$\prec$	$\prec$
100	$\prec$	$\prec$	$\succ$
100	$\prec$	$\prec$	$\succ$
100	$\prec$	$\prec$	$\prec$
125	$\prec$	$\prec$	$\equiv$
125	$\prec$	$\prec$	$\succ$
125	$\prec$	$\prec$	$\prec$
125	$\prec$	$\prec$	$\prec$
125	$\prec$	$\prec$	$\equiv$
150	$\prec$	$\prec$	$\equiv$
150	$\prec$	$\prec$	$\prec$
150	$\prec$	$\prec$	$\succ$
150	$\prec$	$\prec$	$\prec$
150	$\prec$	$\prec$	$\equiv$

applying an iterated single objective method to this kind of problems and shows the need to develop multi-objective methods for them. Furthermore, the large number of potentially Pareto optimal solutions found by the HM is another indicator of the interest to solve this problem as a bi-objective one. Our approach is also able to generate a better quality approximation than the ϵ-constraint method as the average $\mathcal{S}$ metric values of the HM are always better than those of the other method. Moreover, the worst $\mathcal{S}$ value of the HM is better than the $\mathcal{S}$ value for the ϵ-constraint method on 13 instances out of 20.

Additionally, the results in Table 4 show that the HM is able to dominate large parts of the approximation of the other meta-heuristics while the other methods only dominate a marginal number of solutions identified by the HM. This fact combined with the previous one that the HM found more non-dominated solutions reinforce our conclusion that the HM is the most efficient meta-heuristic proposed here.

Finally, statistical tests have been conducted to see if the results related to the $\mathcal{S}$ metric were relevant. To do that, we used the Kruskal–Wallis statistical test with a p value of 5% and compared HM with ϵ-CM and MOEAu as well as ϵ-CM with MOEAu. The results are reported in Table 5. In this table, according to the methods under comparison (A vs. B), $\prec$ means that A is significantly better than B, $\succ$ that B is significantly better than A, and $\equiv$ that there is no significant difference between both. It appears that the HM is always significantly better than MOEAu and that there is only one instance where it is not significantly better than the ϵ-CM. At the same time, no definitive conclusion can be drawn between the ϵ-CM and MOEAu, this indicates that the hybridization is able to improve the robustness of the method and it corroborates our conclusion that the HM is the most efficient meta-heuristic we have designed for the TSPP.

5 Conclusions

We have shown how our new approach to the traveling salesman problem with profits, which uses a bi-objective representation and an ejection chain process with a multi-objective evolutionary algorithm to generate starting solutions, yields an effective method for generating a high quality Pareto set. A computational comparison with an iterated ϵ-constraint implementation of one of the best meta-heuristics for the selective traveling salesman problem shows our method has advantages as the size of the problem increases.

We observe two primary opportunities to improve the performance of our method in future research: first, the possibility to upgrade the Insert and Shake procedure for generating an initial population of tours in the evolutionary approach by incorporating ejection chain strategies to improve the quality of the TSP tours produced. Second, we could employ an alternative evolutionary approach by making use of scatter search and path relinking, which have been shown in a variety of studies to perform more effectively than a genetic algorithm design.

Acknowledgements Nicolas Jozefowiez was supported by a Fulbright grant. This support is gratefully acknowledged. Thanks are also due to F. Semet for providing the code for the Tabu search for the selective traveling salesman problem. Thanks are also due to the referees for their valuable comments.

References

1. Awerbuch, B., Azar, Y., Blum, A., Vempala, S.: New approximation guarantees for minimum-weight *k*-trees and prize-collection salesmen. SIAM J. Comput. **28**, 254–262 (1998)
2. Balas, E.: The prize-collecting traveling salesman problem. Networks **19**, 621–636 (1989)
3. Boffey, B.: Multiobjective routing problems. Top **3**, 167–220 (1995)
4. Deb, K., Pratap, A., Agarwal, S., Meyarvan, T.: A fast and elitist multiobjective genetic algorithm: NSGA II. IEEE Trans. Evolution. Comput. **6**, 182–197 (2002)
5. Dell'Amico, M., Maffioli, F., Värbrand, P.: On prize-collecting tours and the asymmetric travelling salesman problem. Int. Trans. Oper. Res. **2**, 297–308 (1995)
6. Ehrgott, M., Gandibleux, X.: A survey and annotated bibliography of multi-objective combinatorial optimization. OR Spektrum **22**, 425–460 (2000)
7. Feillet, D., Dejax, P., Gendreau, M.: Traveling salesman problems with profits. Trans. Sci. **39**, 188–205 (2005)
8. Gendreau, M., Hertz, A., Laporte, G.: New insertion and postoptimization procedures for the traveling salesman problem. Oper. Res. **40**, 1086–1094 (1992)
9. Gendreau, M., Laporte, G., Semet, F.: A tabu search heuristic for the undirected selective travelling salesman problem. Eur. J. Oper. Res. **106**, 539–545 (1998)
10. Glover, F.: Heuristics for integer programming using surrogate constraints. Decis. Sci. **8**, 156–166 (1977)
11. Glover, F.: New ejection chain and alternating path methods for the traveling salesman problems. Comput. Sci. Oper. Res. 449–509 (1992)
12. Glover, F.: Tabu search for nonlinear and parametric optimization (with links to genetic algorithms). Discrete Appl. Math. **49**, 231–255 (1994)
13. Glover, F., Laguna, M.: Modern heuristic techniques for combinatorial problems. Chapt. Tabu search, pp. 71–140. Blackwell Scientific Publishing (1993)
14. Glover, F., Laguna, M.: Tabu Search. Kluwer Academic Press (1997)
15. Kataoka, S., Morito, S.: An algorithm for the single constraint maximum collection problem. J. Oper. Res. Soc. Jpn. **31**, 515–530 (1988)
16. Keller, C.P.: Multiobjective routing through space and time: the MVP and TDVRP problems. Ph.D. thesis, Department of Geography, University of Western Ontario. London, Ontario, Canada (1985)
17. Keller, C.P., Goodchild, M.: The multiobjective vending problem: a generalization of the traveling salesman problem. Environ. Plann., B. Plann. Des. **15**, 447–460 (1988)
18. Laporte, G., Martello, S.: The selective traveling salesman problem. Discrete Appl. Math. **26**, 193–207 (1990)
19. Rego, C.: Relaxed tours and path ejections for the traveling salesman problem. Eur. J. Oper. Res. **106**, 522–538 (1998)
20. Rosenkrantz, D.J., Stearns, R.E., Lewis II, P.M.: An analysis of several heuristics for the traveling salesman problem. SIAM J. Comput. **6**, 563–581 (1977)
21. Whitley, D., Starkweather, T., Fuquay, D.: Scheduling problems and traveling salesman: the genetic edge recombination operator. In: Schaffer J. (ed.) Proceedings of the Third International Conference on Genetic Algorithms, pp. 133–140 (1989)
22. Zitzler, E.: Evolutionary algorithm for multiobjective optimization: methods and applications. Ph.D. thesis, Swiss Federal Institute of Technology (ETH). Zurich, Switzerland (1999)

On the Integration of a TSP Heuristic into an EA for the Bi-objective Ring Star Problem

Arnaud Liefooghe[1], Laetitia Jourdan[1], Nicolas Jozefowiez[2,3], and El-Ghazali Talbi[1]

[1] LIFL – CNRS – INRIA Lille-Nord Europe,
Université des Sciences et Technologies de Lille,
Parc Scientifique de la Haute Borne, 40 av. Halley, 59650 Villeneuve d'Ascq, France
{Arnaud.Liefooghe,Laetitia.Jourdan,El-Ghazali.Talbi}@lifl.fr
[2] LAAS – CNRS, Université de Toulouse,
7 av. du Colonel Roche, F-31077 Toulouse, France
Nicolas.Jozefowiez@laas.fr
[3] Université de Toulouse, INSA, France

Abstract. This paper discusses a new hybrid solution method for a bi-objective routing problem, namely the bi-objective ring star problem. The bi-objective ring star problem is a generalization of the ring star problem in which the assignment cost has been dissociated from the cost of visiting a subset of nodes. Here, we investigate the possible contribution of incorporating specialized TSP heuristics into a multi-objective evolutionary algorithm. Experiments show that the use of this hybridization scheme allows a strict improvement of the generated sets of non-dominated solutions.

1 Introduction

The purpose of the Bi-objective Ring Star Problem (B-RSP) is to locate an elementary cycle, the so-called *ring*, on a subset of nodes of a graph while optimizing two conflicting costs. First is the minimization of a *ring cost*, proportional to the length of the cycle. Then, nodes that do not belong to the ring are all assigned to visited ones so that the associated cost is minimal. The resulting *assignment cost* is the second objective to be minimized. In spite of its natural bi-objective formulation, this problem is generally investigated in a single-objective way, either where both costs are combined [12] or where the assignment cost is treated as a constraint [13]. Note that both versions of the problem have also been heuristically solved in [16,18]. As pointed out in [10], a large number of routing problems are formulated as multi-objective optimization problems, and according to the same paper, the B-RSP is a generalization of a mono-objective problem. In [15], different multi-objective evolutionary algorithms have been proposed for the B-RSP. Although the approaches were already encouraging, even compared to state-of-the-art mono-objective methods, a few improvement points can be identified. First, a lack of efficiency has been detected on the ring cost throughout the output solutions. Second, the population initialization strategy used within

M.J. Blesa et al. (Eds.): HM 2008, LNCS 5296, pp. 117–130, 2008.

all the search methods was a bit rudimentary, each initial solution having approximately half of its nodes in the cycle. The challenge is then to overcome the identified problems in order to improve the efficiency of those search methods.

An interesting property of the problem under consideration is that, given a fixed set of visited nodes, the related assignment cost is always optimal. It is not the case for the ring cost, for which a classical Traveling Salesman Problem (TSP) is still to be solved among the set of nodes that belong to the ring. Then, once is decided which nodes are visited or not, an objective function is much more difficult to optimize. However, a large number of efficient heuristic methods has been proposed for the TSP. In this paper, our aim is to present a hybrid metaheuristic combining a multi-objective evolutionary algorithm and a problem-specific heuristic, initially designed for the TSP. Approaches where a TSP heuristic is successfully integrated into a multi-objective evolutionary algorithm can, for instance, be found in [8,9].

The reminder of the paper is organized as follows. In Section 2, we give the necessary background for multi-objective optimization, we introduce the B-RSP and we present a heuristic devoted to the TSP. The hybrid metaheuristic proposed to solve the B-RSP is detailed in Section 3. In Section 4, computational experiments are conducted. At last, conclusions and perspectives are drawn in the last section.

2 Background

In this section, we first discuss multi-objective optimization and define some related concepts. Then, we present the bi-objective ring star problem in details and we introduce a heuristic devoted to the traveling salesman problem.

2.1 Multi-Objective Optimization

A general *Multi-objective Optimization Problem* (MOP) can be defined by a set of $n \geq 2$ objective functions $f_1, f_2, \ldots, f_n$; a set of feasible solutions in the *decision space*, denoted by X; and a set of feasible points in the *objective space*, denoted by Z. Each function can be either minimized or maximized, but we here assume that all n objective functions are to be minimized. To each decision vector $x \in X$ is assigned exactly one objective vector $z \in Z$ on the basis of a vector function $f : X \to Z$ with $z = f(x) = (f_1(x), f_2(x), \ldots, f_n(x))$.

Definition 1. *An objective vector $z \in Z$* weakly dominates *another objective vector $z' \in Z$ if and only if $\forall i \in [1..n]$, $z_i \leq z'_i$.*

Definition 2. *An objective vector $z \in Z$* dominates *another objective vector $z' \in Z$ if and only if $\forall i \in [1..n]$, $z_i \leq z'_i$ and $\exists j \in [1..n]$ such as $z_j < z'_j$.*

Definition 3. *An objective vector $z \in Z$ is* non-dominated *if and only if there does not exist another objective vector $z' \in Z$ such that z' dominates z.*

A solution $x \in X$ is said to be *efficient* (or *non-dominated*) if $f(x)$ is non-dominated. The set of all efficient solutions is the *efficient set*, denoted by X_E. The set of all non-dominated vectors is the *non-dominated front* (or the *trade-off surface*), denoted by Z_N. A possible approach to solve a MOP consists of finding or approximating a minimal set of efficient solutions, *i.e.* one solution $x \in X_E$ for each non-dominated point $z \in Z_N$ such as $f(x) = z$ (in case multiple solutions map to the same non-dominated vector). Evolutionary algorithms are commonly used to this end as they naturally find multiple and well-spread non-dominated solutions in a single simulation run. The reader could refer to [3,4] for more details about evolutionary multi-objective optimization.

2.2 The Bi-objective Ring Star Problem

The *Bi-objective Ring Star Problem* (B-RSP) can be described as follows. Let $G = (V, E, A)$ be a complete mixed graph where $V = \{v_1, v_2, \ldots, v_n\}$ is a set of vertexes, $E = \{[v_i, v_j] | v_i, v_j \in V, i < j\}$ is a set of edges, and $A = \{(v_i, v_j) | v_i, v_j \in V\}$ is a set of arcs. Vertex v_1 is the depot. To each edge $[v_i, v_j] \in E$ we assign a non-negative *ring cost* c_{ij}, and to each arc $(v_i, v_j) \in A$ we assign a non-negative *assignment cost* d_{ij}. The B-RSP consists of locating a simple cycle through a subset of nodes $V' \subset V$ (with $v_1 \in V'$) while (i) minimizing the sum of the ring costs related to all edges that belong to the cycle, and (ii) minimizing the sum of the assignment costs of arcs directed from every non-visited node to a visited one so that the associated cost is minimum. An example of solution is given in Figure 1, where solid lines represent edges that belong to the ring and dashed lines represent arcs of the assignments.

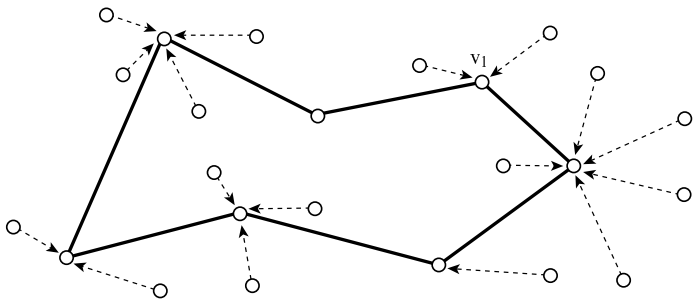


Fig. 1. An example of solution for the ring star problem

The first objective is called the *ring cost* and is defined as:

$$\sum_{[v_i, v_j] \in E} c_{ij} b_{ij} \ , \tag{1}$$

where b_{ij} is a binary variable equal to 1 if and only if the edge $[v_i, v_j]$ belongs to the cycle. The second objective, the *assignment cost*, can be computed as follows:

$$\sum_{v_i \in V \setminus V'} \min_{v_j \in V'} d_{ij} \ . \tag{2}$$

Let us remark that these two objectives are comparable only if we assume that the ring cost and the assignment cost are commensurate one to another, what is rarely the case in practice. Furthermore, the fact of privileging a cost compared to the other is closely related to the decision-maker preferences. However, the B-RSP is an NP-hard combinatorial problem since the particular case of visiting the whole set of nodes is equivalent to a traditional Traveling Salesman Problem (TSP).

2.3 GENIUS, a TSP Heuristic

A specificity of the B-RSP is that many TSP generally need to be solved. An effective TSP heuristic method is thus rather appreciated in order to improve the ring cost of a solution. There exists a large range of heuristics that are devoted to the TSP. One of them is GENIUS, proposed by Gendreau et al. [6]. Briefly, GENIUS contains a tour construction phase, called GENI, and a postoptimization phase, called US. Starting with three arbitrary nodes, GENI inserts, at each iteration, an unrouted node between two of its p closest neighbors on the partially constructed tour, where p is a user-controlled parameter. When inserting the vertex, GENI also performs a local reoptimization of the tour. Once a complete tour has been built, the US postoptimization procedure is repeatedly applied to the tour until no further improvement is possible. During this procedure, nodes are successively removed from the tour, and then reinserted, according to the same rules used in the tour construction phase. The use of GENIUS can be seen as a black-box mechanism integrated into the hybrid metaheuristic presented in the next section, and could practically be replaced by another TSP heuristic.

3 A Hybrid Metaheuristic for the Bi-objective Ring Star Problem

The main process of the Hybrid Metaheuristic (HM) proposed in the paper to solve the B-RSP consists of an elitist multi-objective Evolutionary Algorithm (EA). A first hybridization mechanism arises at the very beginning of the HM, as the initial population is built thanks to a problem-specific heuristic. This initial population is used as a starting point of the EA, so that both methods cooperate in pipeline way. Second, an additional hybridization scheme conditionally appears at every generation of the EA, where the ring cost of each population member is attempted to be improved thanks to a TSP heuristic. Finally, the EA is itself hybrid, as it is divided into two different phases. Those ones differ the one from the other at the selection and the replacement steps of the EA. During both phases, a secondary population, the so-called archive, is used to store every potentially efficient solutions found so far. The first phase is compound of an elitist selection step where parent individuals are all selected from the archive only. The replacement step is a generational one, *i.e.* the parent population is replaced by the offspring one. This phase corresponds to the *Simple*

Elitist Evolutionary Algorithm (SEEA) introduced in [15]. The main particularity of SEEA is that no fitness assignment scheme is required, the population being the only problem-independent parameter. The second phase is founded on the *Indicator-Based Evolutionary Algorithm* proposed by Zitzler and Künzli [21]. The fitness assignment scheme of IBEA is based on a pairwise comparison of population items by using a binary quality indicator I. Several indicators can be used for such a purpose [21], and we here choose to use the binary additive ϵ-indicator ($I_{\epsilon+}$) proposed in [23]. $\mathrm{I}_{\epsilon+}$ gives the minimum factor by which a non-dominated set A has to be translated in the objective space to weakly dominate a non-dominated set B. The selection scheme for reproduction is a binary tournament between randomly chosen individuals. The replacement strategy consists of deleting, one-by-one, the worst individuals, and in updating the fitness values of the remaining solutions each time there is a deletion; this is continued until the required population size is reached. The first phase of the EA will allow to find a rough approximation of the efficient set in a very short amount of time whereas the second phase will rather be devoted to improve this set in a more intensive way. The transition from Phase 1 to Phase 2 will occur as soon as the archive of non-dominated solutions does not improve enough with regards to the search scenario. The main steps of our HM are the following ones:

1. **Initialization.** Generate an initial population P of size N (see Section 3.2); generate an efficient set approximation A with the non-dominated individuals contained in P; create an empty offspring population P'.
2. **Selection.** Repeat until $|P'| = N$:
   (*Phase* 1*)* Randomly select an individual from A and add it to the offspring population P'.
   (*Phase* 2*)* Select an individual thanks to a binary tournament selection on P and add it to the offspring population P'.
3. **Recombination.** Apply a recombination operator to pairs of individuals contained in P' with a given probability p_r (see Section 3.3).
4. **Mutation.** Apply a mutation operator to individuals contained in P' with a given probability p_m (see Section 3.4).
5. **Fitness assignment.**
   (*Phase* 1*)* $\emptyset$.
   (*Phase* 2*)* Calculate fitness values of any individual x contained in $P \cup P'$; *i.e.* $F(x) \leftarrow \sum_{x' \in (P \cup P') \setminus \{x\}} -e^{-I(\{x'\},\{x\})/\kappa}$, where $\kappa > 0$ is a scaling factor.
6. **Replacement.**
   (*Phase* 1*)* $P \leftarrow P'$; $P' \leftarrow \emptyset$.
   (*Phase* 2*)* $P \leftarrow P \cup P'$; $P' \leftarrow \emptyset$. Iterate the following steps until the size of the population P does not exceed N:
   - Choose an individual $x^\star \in P$ with the smallest fitness value; *i.e.* $F(x^\star) \leq F(x)$ for all $x \in P$.
   - Remove $x^\star$ from P.
   - Update the fitness values of the individuals remaining in P; *i.e.* $F(x) \leftarrow F(x) + e^{-I(\{x^\star\},\{x\})/\kappa}$ for all $x \in P$.

7. **Elitism.** $A \leftarrow$ non-dominated individuals of $A \cup P$.
8. **Improvement.** If a given condition is satisfied, apply an improvement procedure on any individual contained in P (see Section 3.5).
9. **Termination.** If a stopping criteria is satisfied return A, else go to Step 2.

The principle of the HM is illustrated in Figure 2. According to the taxonomy of hybrid metaheuristics proposed in [20], the HM proposed in this paper can be classified on the *high-level relay hybrid* class, where self-contained heuristics are executed in sequence. The problem-specific components are explained in details below.

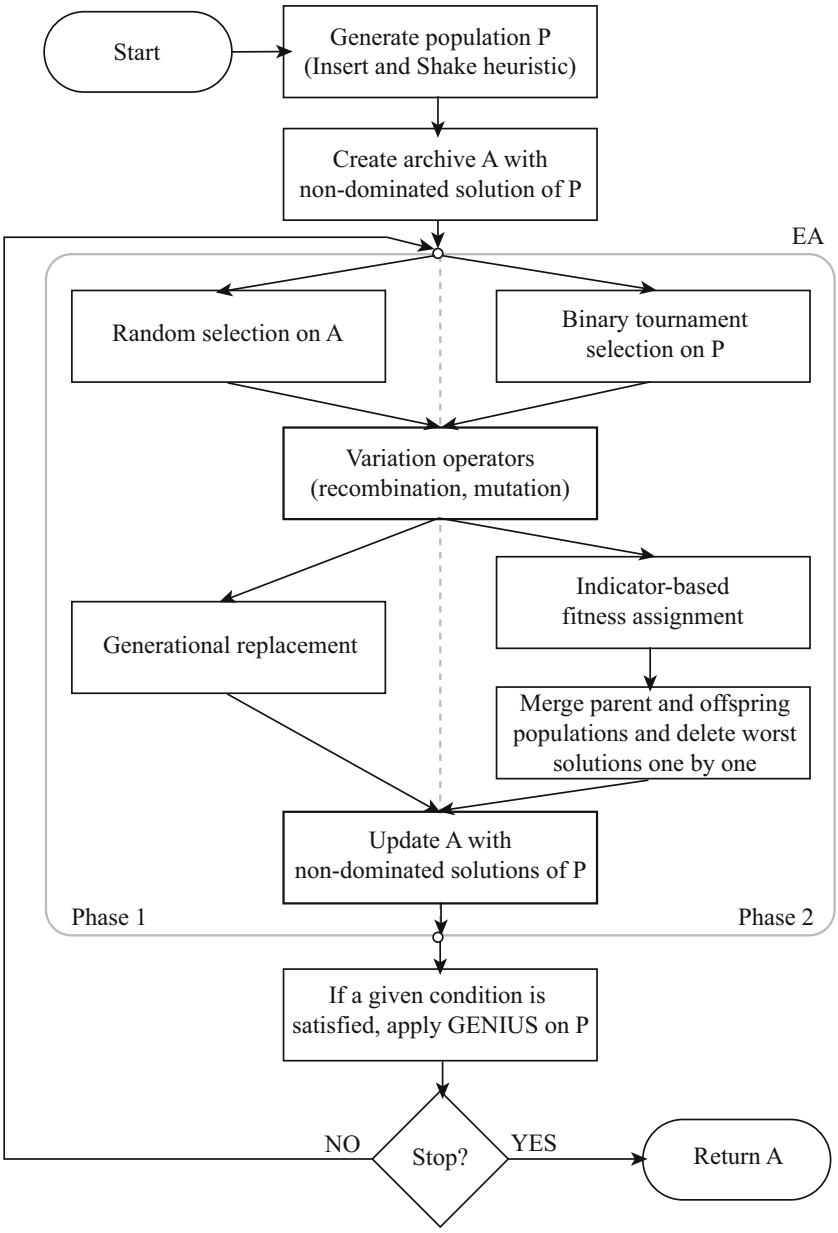


Fig. 2. Flowchart of the Hybrid Metaheuristic (HM)

Vertex	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}
Random key	0	0.7	-	0.3	-	0.8	0.2	-	0.5	-

Fig. 3. A RSP solution represented by random keys

3.1 Solution Encoding

The representation of a B-RSP solution is based on the random keys mechanism proposed by Bean [1]. A *random key* $k_i \in [0, 1[$ is assigned to every node v_i that belongs to the ring. A special value is assigned to unvisited nodes. Thus, the ring route associated to a solution corresponds to the nodes read according to their random keys in the increasing order; *i.e.* if $k_i < k_j$, then v_j comes after v_i. A possible representation for the cycle (v_1,v_7,v_4,v_9,v_2,v_6) is given in Figure 3. Nodes v_3,v_5,v_8 and v_{10} are assigned to a visited node in such a way that the associated assignment cost is minimum.

3.2 Population Initialization

An initial population of N individuals is built by means of repeatedly solving a mono-objective problem closely linked to the B-RSP. This problem, that will be denoted by Ring Cost Constrained RSP (RCC-RSP), consists of minimizing the assignment cost only, while satisfying an upper bound on the ring cost. It is obtained by removing the ring cost from the set of objective functions of the B-RSP, and by adding a new constraint stipulating that the ring cost cannot exceed a given limit $c_{\max}$. A search mechanism is iterated with distinct $c_{\max}$ values such that the set of resulting problems corresponds to different part of the objective space.

In order to approximately solve a given RCC-RSP, we use the *Insert and Shake Heuristic* (ISH), initially proposed by Gendreau et al. [7] for a single-objective routing problem called the selective TSP. This method combines a TSP tour extension heuristic described in Rosenkrantz et al. [19] and the GENIUS procedure described in Section 2.3. ISH gradually extends a tour T until no other node can be added without violating a given ring cost limit $c_{\max}$. At a given step, the non-visited node to be inserted in T is chosen so that the ratio between its current assignment cost and the increment on the global ring cost after its insertion is minimum. Then, GENIUS is applied in an attempt to obtain a better cycle on the nodes in T. If GENIUS fails, the procedure terminates. Otherwise, more node insertions are attempted and the process is repeated. The steps to build an initial population of solutions are the following ones:

1. GENIUS is applied to find a cycle containing all nodes in V, and this solution is included in the population. Let $c^\star$ be the ring cost of this solution. Set $\alpha \leftarrow \frac{c^\star}{N-1}$, $c_{\max} \leftarrow c^\star - \alpha$ and $i \leftarrow 1$.
2. If $i > N$, stop. Otherwise, generate a RCC-RSP solution by means of ISH and insert this solution into the population.
3. Set $c_{\max} \leftarrow c_{\max} - \alpha$, and $i \leftarrow i + 1$. Go to Step 2.

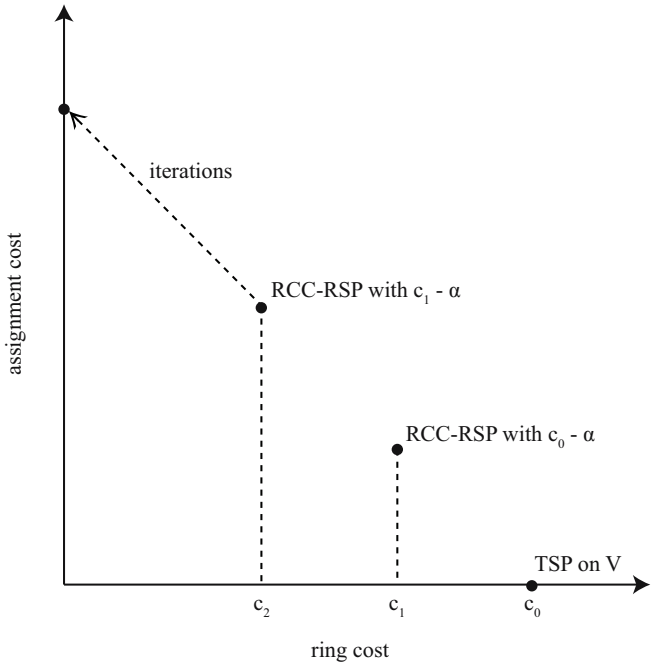


Fig. 4. Illustration of the population initialization heuristic

This initialization strategy is illustrated in Figure 4. Thanks to this heuristic, the starting set of solutions will already be both (*i*) quite efficient, and (*ii*) well-spread on the objective space.

3.3 Recombination Operator

The recombination operator is a quadratic crossover closely related to the one proposed in [18]. Two randomly selected solutions x_1 and x_2 are first divided according to a particular position. Then, the first part of x_1 is combined with the second part of x_2 to build a first offspring, and the first part of x_2 is combined with the second part of x_1 to build a second offspring. Every node retains its random key so that it enables an easy reconstruction of the new individuals. Thanks to the random keys encoding mechanism, solutions having different ring sizes can easily be recombined, even if the initial ring structures are generally broken in the offspring solutions.

3.4 Mutation Operator

The mutation operator designed for the problem under consideration consists of the following strategy. A node $v^\star \in V \setminus \{v_1\}$ is selected at random. Therefore, two cases may arise. First, if $v^\star$ belongs to the ring, it is removed and then belongs to the set of unvisited nodes. Second, if $v^\star$ does not belong to the cycle, it is added. The position to insert $v^\star$ is chosen so that the increment on the ring cost is minimum.

3.5 Improvement Procedure

The improvement procedure consists of improving the ring cost of the current population by applying GENIUS on any of its members. The main issue is now to determine when this heuristic might start in order to find the good trade-off between the efficiency and the effectiveness of GENIUS. To do so, we decide to launch it only if no more than n solutions per iteration have been included in the archive during the last M consecutive generations. Hence, we hope that the improved population will produce new non-dominated solutions and will help to build more interesting individuals in the future steps of the HM.

4 Computational Experiments

In order to assess the effectiveness of our method, we will measure its performance in comparison to the ACS method proposed in [15]. The latter is an auto-adaptive method based on a simple elitist evolutionary algorithm and a population-based local search. It has been shown to be particularly efficient to solve the RSP as a bi-objective problem. To quantity the impact of GENIUS on our HM, we also implemented a more basic version in which GENIUS is not involved, neither in the improvement step nor in the initialization step. This other hybrid metaheuristic will be denoted by HM2 in the remainder of the paper. All the algorithms have been implemented under the ParadisEO-MOEO library[1] [14] and share the same base components for a fair comparison between them. Computational runs were performed under Linux on an Intel Core 2 Duo 6600 (2×2.40 GHz) machine, with 2 GB RAM.

4.1 Performance Assessment

Experiments have been conducted on a set of benchmark test instances taken from the TSPLIB[2] [17]. These instances involve between 51 and 264 nodes. The number at the end of an instance's name represents the number of nodes for the instance under consideration. Let l_{ij} denote the distance between two nodes v_i and v_j of a TSPLIB file. Then, the ring cost c_{ij} and the assignment cost d_{ij} have both been set to l_{ij} for every pair of nodes v_i and v_j.

For each search method, a set of 20 runs per instance has been performed. In order to evaluate the quality of the non-dominated front approximations, we follow the protocol given by Knowles et al. [11]. For a given instance, we first compute a reference set $Z_N^\star$ of non-dominated points extracted from the union of all the fronts we obtained during our experiments and the best non-dominated set taken from [15][3]. Second, we define a point $z^{max} = (z_1^{max}, z_2^{max})$, where z_1^{max} (respectively z_2^{max}) denotes the upper bound of the first (respectively second) objective in the whole non-dominated front approximations. Then, to

[1] ParadisEO is available at `http://paradiseo.gforge.inria.fr`

[2] `http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/`

[3] These results are available at `http://www.lifl.fr/~liefooga/rsp/`

measure the quality of an output set A in comparison to $Z_N^\star$, we compute the difference between these two sets by using the unary hypervolume metric [22], $(1.05 \times z_1^{max}, 1.05 \times z_2^{max})$ being the reference point. The hypervolume difference indicator (I_H^-) computes the portion of the objective space that is weakly dominated by $Z_N^\star$ and not by A. The more this measure is close to 0, the better is the approximation A. Furthermore, we also consider one of the ϵ-indicators proposed in [23]. The unary additive ϵ-indicator ($\mathrm{I}_{\epsilon+}^1$) gives the minimum factor by which an approximation A has to be translated in the objective space to weakly dominate the reference set $Z_N^\star$. As a consequence, for each test instance, we obtain 20 I_H^- measures and 20 $\mathrm{I}_{\epsilon+}$ measures, corresponding to the 20 runs, per algorithm. As suggested by Knowles et al. [11], once all these values are computed, we perform a statistical analysis on pairs of optimization methods for a comparison on a specific test instance. To this end, we use the Mann-Whitney statistical test as described in [11], with a p-value lower than 5%. Hence, for a specific test instance, and according to the p-value and to the metric under consideration, this statistical test reveals if the sample of approximation sets obtained by a given search method is significantly better than the one obtained by another search method, or if there is no significant difference between both. Note that all the performance assessment procedures have been achieved using the performance assessment tool suite provided in PISA[4] [2].

4.2 Parameter Setting

For each investigated metaheuristic, the search process stops after a certain amount of run time. As shown in Table 1, this stopping criteria has been arbitrary set according to the size of the problem instance to be solved. Next, the population size N is set to 100; the recombination probability p_r is set to 0.25 and the mutation probability p_m is set to 1.0. Following [21], the scaling factor κ is set to 0.05. The improvement procedure of HM is launched only if the number of elements received by the archive is less than 1.0% of its current size for $|V|$ consecutive iterations, where $|V|$ is the number of nodes for the instance under consideration. Finally, the GENIUS parameter p is set to 7.

Table 1. Stopping criteria: running time

Instance	Running time	Instance	Running time
eil51	20”	*kroA150*	10’
st70	1’	*kroA200*	20’
kroA100	2’	*pr264*	30’
bier127	5’		

[4] The package is available at `http://www.tik.ee.ethz.ch/pisa/assessment.html`

Table 2. Comparison of HM, HM2 and ACS [15] according to the I_H^- and the $I_{\epsilon+}$ metrics by using a Mann-Whitney statistical test with a p-value of 5%. For the metric under consideration, either the results of the algorithm located at a specific row are significantly better than those of the algorithm located at a specific column ($\succ$), either they are worse ($\prec$), or there is no significant difference between both ($\equiv$).

		I_H^-			$I_{\epsilon+}$		
		HM	HM2	ACS	HM	HM2	ACS
eil51	HM	-	$\succ$	$\succ$	-	$\succ$	$\succ$
	HM2	$\prec$	-	$\equiv$	$\prec$	-	$\equiv$
	ACS	$\prec$	$\equiv$	-	$\prec$	$\equiv$	-
st70	HM	-	$\succ$	$\succ$	-	$\succ$	$\succ$
	HM2	$\prec$	-	$\equiv$	$\prec$	-	$\equiv$
	ACS	$\prec$	$\equiv$	-	$\prec$	$\equiv$	-
kroA100	HM	-	$\succ$	$\succ$	-	$\succ$	$\succ$
	HM2	$\prec$	-	$\prec$	$\prec$	-	$\prec$
	ACS	$\prec$	$\succ$	-	$\prec$	$\succ$	-
bier127	HM	-	$\succ$	$\succ$	-	$\succ$	$\succ$
	HM2	$\prec$	-	$\succ$	$\prec$	-	$\succ$
	ACS	$\prec$	$\prec$	-	$\prec$	$\prec$	-
kroA150	HM	-	$\succ$	$\succ$	-	$\succ$	$\succ$
	HM2	$\prec$	-	$\equiv$	$\prec$	-	$\prec$
	ACS	$\prec$	$\equiv$	-	$\prec$	$\succ$	-
kroA200	HM	-	$\succ$	$\succ$	-	$\succ$	$\succ$
	HM2	$\prec$	-	$\equiv$	$\prec$	-	$\prec$
	ACS	$\prec$	$\equiv$	-	$\prec$	$\succ$	-
pr264	HM	-	$\succ$	$\succ$	-	$\succ$	$\succ$
	HM2	$\prec$	-	$\equiv$	$\prec$	-	$\prec$
	ACS	$\prec$	$\equiv$	-	$\prec$	$\succ$	-

4.3 Results and Discussion

First of all, note that we initially experimented some algorithm versions where only the first or the second phase of the EA is involved, with and without GENIUS. But the resulting metaheuristics turned out to be significantly outperformed by HM and HM2. The comparison of results obtained by HM, HM2 and ACS are presented in Table 2. According to both indicators (I_H^- and $I_{\epsilon+}$), HM is statistically better than any other search methods on every instance we investigated. Besides, the difference between HM2 and ACS is often not significant according to the I_H^- metric, whereas ACS generally outperforms HM2 according to the $I_{\epsilon+}$ metric. The only instance for which HM2 performs statistically higher than ACS is the *bier127* instance, where it obtains better values for both metrics. In order to study on which part of the trade-off surface the differences between HM, HM2 and ACS appear, examples of empirical attainment functions [5] are given in Figure 5 and Figure 6 for the *bier127* instance. They represent the limit of the objective space that is attained by at least 90% of the runs for every search method. For the instance under consideration, we can see that HM seems to be more capable of finding solutions having both a large number of visited nodes

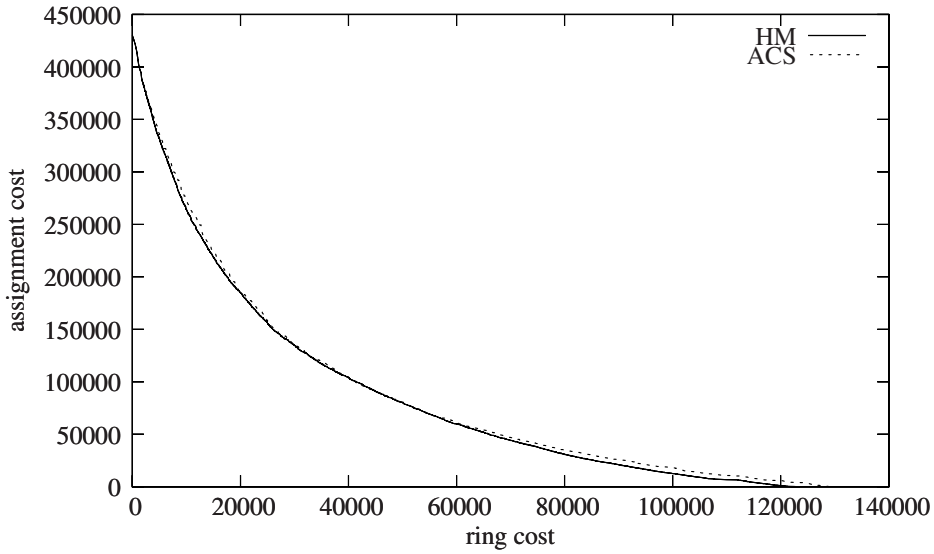


Fig. 5. 90%-attainment surface plot obtained by the approximation sets found by HM and ACS [15] for the *bier127* test instance

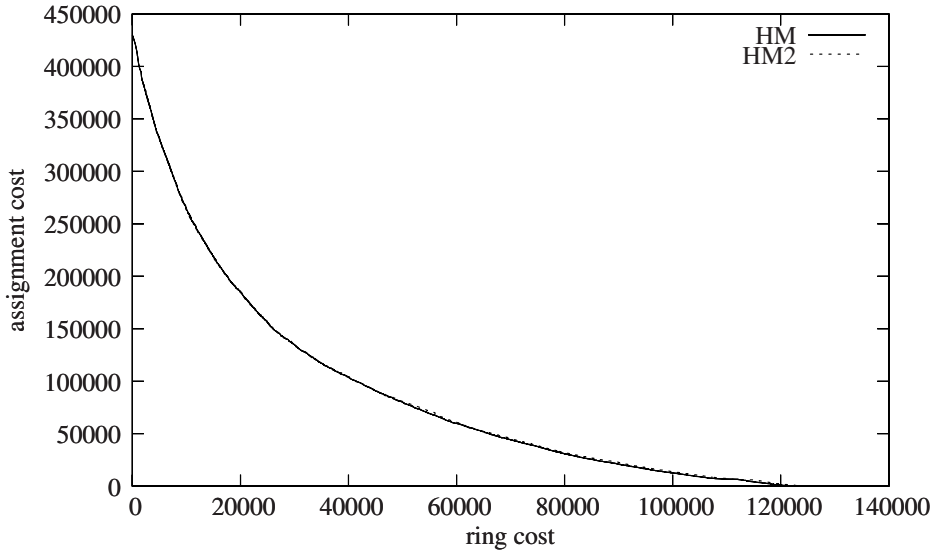


Fig. 6. 90%-attainment surface plot obtained by the approximation sets found by HM and HM2 for the *bier127* test instance

and a good ring cost. Thus, the superiority of HM relatively to HM2 reveals the benefit of integrating a TSP heuristic, here symbolized by GENIUS, into our EA for the problem to be solved. Moreover, despite its relative simplicity in comparison to ACS, the HM is quite effective to solve the B-RSP, especially to find solutions having a low ring cost. This indicates that the hybridization scheme largely improves the method and reveals that the HM introduced in this paper outperforms the metaheuristics proposed so far to solve the RSP as a bi-objective optimization problem.

5 Conclusion and Perspectives

In this paper, a new hybrid metaheuristic has been proposed to approximate the efficient set of a multi-objective routing problem called the bi-objective ring star problem. This problem is commonly investigated in a single-objective way, either where both objectives are aggregated, or where one objective is regarded as a constraint. However, within the frame of the ring star problem, many traveling salesman problems generally need to be solved. The purpose of the hybrid metaheuristic proposed here is then to integrate a heuristic algorithm for the traveling salesman problem, namely GENIUS, into a multi-objective evolutionary algorithm to solve the bi-objective ring star problem as a whole. The hybrid search method starts with a problem-specific heuristic to generate an initial set of solutions, and continues with a two-phase elitist evolutionary algorithm hybridized to the GENIUS heuristic. The latter is launched to intensify the search in an auto-adaptive manner, according to the convergence scenario of the main process. Experiments were conducted on a set of benchmark test instances, and validated the contribution of the traveling salesman problem heuristic into the hybrid method. They also reveal that the metaheuristic proposed in the paper largely outperforms our previous investigations for resolving the bi-objective ring star problem. As a next step, we will experiment other strategies to scale the GENIUS application factor in order to study the influence of this parameter on the global performance of our method. We will also try to replace GENIUS by other kinds of traveling salesman problem heuristics, or even exact methods, within our hybrid metaheuristic to assert the genericity of our method. Moreover, it could be interesting to design a more universal model of hybridization to solve multi-objective optimization problems, where both problem-specific and meta methods could be integrated.

References

1. Bean, J.: Genetic algorithms and random keys for sequencing and optimization. ORSA Journal on Computing 6(2), 154–160 (1994)
2. Bleuler, S., Laumanns, M., Thiele, L., Zitzler, E.: PISA — a platform and programming language independent interface for search algorithms. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) EMO 2003. LNCS, vol. 2632, pp. 494–508. Springer, Heidelberg (2003)
3. Coello Coello, C.A., Van Veldhuizen, D.A., Lamont, G.B.: Evolutionary Algorithms for Solving Multi-Objective Problems. Kluwer Academic Publishers, Boston (2002)
4. Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms. John Wiley & Sons, Chichester (2001)
5. Fonseca, C.M., Grunert da Fonseca, V., Paquete, L.: Exploring the performance of stochastic multiobjective optimisers with the second-order attainment function. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 250–264. Springer, Heidelberg (2005)
6. Gendreau, M., Hertz, A., Laporte, G.: New insertion and postoptimization procedures for the traveling salesman problem. Operations Research 40(6), 1086–1094 (1992)

7. Gendreau, M., Laporte, G., Semet, F.: A tabu search heuristic for the undirected selective travelling salesman problem. European Journal of Operational Research 106, 539–545 (1998)
8. Jozefowiez, N., Glover, F., Laguna, M.: Multi-objective meta-heuristics for the traveling salesman problem with profits. Journal of Mathematical Modelling and Algorithms 7(2), 177–195 (2008)
9. Jozefowiez, N., Semet, F., Talbi, E.-G.: The bi-objective covering tour problem. Computers and Operations Research 34(7), 1929–1942 (2007)
10. Jozefowiez, N., Semet, F., Talbi, E.-G.: Multi-objective vehicle routing problems. European Journal of Operational Research 189(2), 293–309 (2008)
11. Knowles, J., Thiele, L., Zitzler, E.: A tutorial on the performance assessment of stochastic multiobjective optimizers. Technical report, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Switzerland (revised version) (2006)
12. Labbé, M., Laporte, G., Rodríguez Martín, I., Salazar González, J.J.: The ring star problem: Polyhedral analysis and exact algorithm. Networks 43, 177–189 (2004)
13. Labbé, M., Laporte, G., Rodríguez Martín, I., Salazar González, J.J.: Locating median cycles in networks. European Journal of Operational Research 160(2), 457–470 (2005)
14. Liefooghe, A., Basseur, M., Jourdan, L., Talbi, E.-G.: ParadisEO-MOEO: A framework for evolutionary multi-objective optimization. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 386–400. Springer, Heidelberg (2007)
15. Liefooghe, A., Jourdan, L., Talbi, E.-G.: Metaheuristics and their hybridization to solve the bi-objective ring star problem: a comparative study. Working paper RR-6515, Institut National de Recherche en Informatique et Automatique (INRIA) (2008)
16. Moreno Pérez, J.A., Moreno-Vega, J.M., Rodríguez Martín, I.: Variable neighborhood tabu search and its application to the median cycle problem. European Journal of Operations Research 151(2), 365–378 (2003)
17. Reinelt, G.: TSPLIB – A traveling salesman problem library. ORSA Journal on Computing 3(4), 376–384 (1991)
18. Renaud, J., Boctor, F.F., Laporte, G.: Efficient heuristics for median cycle problems. Journal of the Operational Research Society 55(2), 179–186 (2004)
19. Rosenkrantz, D.J., Stearns, R.E., Lewis II., P.M.: An analysis of several heuristics for the traveling salesman problem. SIAM Journal on Computing 6(3), 563–581 (1977)
20. Talbi, E.-G.: A taxonomy of hybrid metaheuristics. Journal of Heuristics 8(2), 541–564 (2002)
21. Zitzler, E., Künzli, S.: Indicator-based selection in multiobjective search. In: Yao, X., Burke, E.K., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J.E., Tiňo, P., Kabán, A., Schwefel, H.-P. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 832–842. Springer, Heidelberg (2004)
22. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. IEEE Transactions on Evolutionary Computation 3(4), 257–271 (1999)
23. Zitzler, E., Thiele, L., Laumanns, M., Foneseca, C.M., Grunert da Fonseca, V.: Performance assessment of multiobjective optimizers: An analysis and review. IEEE Transactions on Evolutionary Computation 7(2), 117–132 (2003)