REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE SAAD DAHLAB DE BLIDA 1

Faculté des Sciences
Département de Mathématiques

THESE DE DOCTORAT

en Mathématiques
Spécialité : **Recherche Opérationnelle**

Problème de tournées de véhicules et ses variantes

Par

LAMAMRI Abdelkader

Devant le jury composé de :

Mustapha CHELLALI	Président	Prof.	Univ. Blida 1
Mohamed HACHAMA	Directeur de thèse	Prof.	Univ. Blida 1
Redouane BOUDJEMAA	Examinateur	MCA	Univ. Blida 1
Mohamed Amine BOUTICHE	Examinateur	Prof.	USTHB
Diffalah LAISSAOUI	Examinateur	MCA	Univ. Médéa
Abdelaziz RASSOUL	Examinateur	Prof.	ENSH

REMERCIEMENTS

Je tiens à exprimer toute ma reconnaissance et ma gratitude à mon Directeur de thèse, le Professeur Mohammed HACHAMA. Son expérience, sa patience, ses encouragements, sa confiance et ses orientations ont été inestimables tout au long de ce travail.

Je remercie particulièrement l'ensemble des membres du jury de thèse, les Professeurs A. RASSOUL, A. BOUTICHE, D. LAISSAOUI et R. BOUDJEMAA et, en particulier, Professeur M. CHELLALI, président du jury.

Merci également à mes collègues qui m'ont permis de travailler dans une ambiance excellente. Je salue enfin et chaleureusement la confiance de mes parents qui ont cru en moi. Ce travail leur est dédié.

ملخص

مشكلة جولات المركبات (VRP) هي مشكلة أساسية في الأمثلية التوافقية وبرمجة الأعداد الصحيحة ولها العديد من التطبيقات الهامة. عادةً ما يتم حل VRP باستخدام تقنيات التفرع والحد، التي تتطلب حل مشكلة أقصر مسار مع قيود على الموارد SPPRC أو مشكلة المسار الأقصر الأولي مع قيود الموارد (ESPPRC) وتحديد الحد الأدنى، والذي يمكن حسابه باستخدام طريقة توليد الأعمدة. يتمثل (SPPRC) أو (ESPPRC) في إيجاد مسار (أو مسار أولي) بأقل تكلفة في رسم بياني ذو قيم يخضع لقيود استهلاك الموارد. تتطلب الحلول الدقيقة المقترحة لهذه المشكلة التي تصنف على أنها صعبة وقت حساب مفرط يزيد مع عدد الموارد.

في هذه الأطروحة، نقترح حلًا تقريبيًا جديدًا لـ SPPRC وESPPRC لرسوم بيانية كيفية (دائرية أو غير دائرية). تعتمد طريقتنا على استرخاء لاجرانج لمجموعة فرعية من القيود واستخدام الهيمنة فقط على مجموعة فرعية من الموارد. يؤدي ذلك إلى تقليل مساحة البحث ويسمح للمستخدمين بحساب الحلول المستخدمة بكفاءة لتحسين إجراء طريقة توليد الأعمدة. أجرينا تجارب مكثفة لتقييم طرقنا ومقارنتها مع تقنيات أخرى. الطريقتان المقترحتان لحل SPPRC وESPPRC تحقق حل وسط جيد بين الكفاءة (وقت الحساب) وجودة حلول المشكلتين وكذلك VRP. تظهر النتائج تحسينات بالمقارنة مع تقنيات أخرى.

وبالتالي، يمكن استخدام طريقتنا في تطبيقات VRP العملية وذات حجم كبير.

ABSTRACT

Vehicle routing problem (VRP) is a fundamental combinatorial optimization and integer programming problem with several important applications. The VRP is usually solved by using branch-and-bound techniques requiring solving a shortest path problem with resource constraints (SPPRC) or an elementary shortest path problem with resource constraints (ESPPRC) and the determination of a lower bound, which can be computed using column generation. The SPPRC (or the ESPPRC) entails finding the minimum cost path (or elementary path) in a valuated graph that is subject to constraints on resource consumption. The available exact resolution methods to this NP-hard problem require an excessive computation time which increases with the number of resources.

In this thesis, we propose a new approximate resolution of the SPPRC and ESPPRC for arbitrary graphs (acyclic or cyclic). Our method is based on a Lagrangian relaxation of a subset of the constraints and using dominance only on a subset of the resources. This reduces the search space and allows users to efficiently compute solutions used to improve the column generation procedure. We conducted extensive experiments to evaluate our methods and compare them with other techniques. The two proposed methods to solve the SPPRC and ESPPRC achieve a good compromise between the efficiency (computation time) and the quality of the solutions for the two problems, in addition to the VRP. Results show improvements over state-of-the-art techniques. Our method can be used for practical large-scale VRP applications.

RESUME

Le problème de tournées de véhicules (VRP) est un problème fondamental d'optimisation combinatoire et de programmation en nombres entiers avec plusieurs applications importantes. Le VRP est généralement résolu en utilisant des techniques branch-and-bound nécessitant la résolution d'un problème du plus court chemin avec contraintes de ressources (SPPRC) ou bien le problème du plus court chemin élémentaire avec contraintes de ressources (ESPPRC) et la détermination d'une borne inférieure, qui peut être calculée en utilisant la génération de colonnes. Le SPPRC (ou le ESPPRC) consiste à trouver un chemin (ou chemin élémentaire) de coût minimum dans un graphe valué soumis à des contraintes de consommation de ressources. Les solutions exactes proposées à ce problème NP-difficile nécessitent un temps de calcul excessif qui augmente avec le nombre de ressources.

Dans cette thèse, nous proposons une nouvelle résolution approchée du SPPRC et ESPPRC pour des graphes quelconques (acycliques ou cycliques). Notre méthode est basée sur une relaxation lagrangienne d'un sous-ensemble des contraintes et l'utilisation de la dominance seulement sur un sous-ensemble des ressources. Cela réduit l'espace de recherche et permet aux utilisateurs de calculer efficacement les solutions utilisées pour améliorer la procédure de génération de colonnes. Nous avons mené des expériences approfondies pour évaluer nos méthodes et les comparer avec d'autres techniques. Les deux méthodes proposées pour la résolution du SPPRC et ESPPRC réalisent un bon compromis entre l'efficacité (temps de calcul) et la qualité des solutions des deux problèmes et aussi du VRP. Les résultats montrent des améliorations par rapport à d'autres techniques. Ainsi, notre méthode peut être utilisée pour des applications pratiques du VRP à grande échelle.

TABLE DES MATIERES

TABLE DES FIGURES

LISTE DES TABLEAUX

LISTE DES ACRONYMES

TSP : Traveling Salesman Problem (problème du voyageur de commerce).

VRP : Vehicle Routing Problem (problème de tournées de véhicules).

VRPS : Vehicle Routing Problems (problèmes de tournées de véhicules).

CVRP : Capacitated VRP (problème de tournées de véhicules avec contraintes de capacité).

VRPTW : VRP with time windows (problème de tournées de véhicules avec fenêtres de temps).

VRPPD : VRP with Pick-up and Delivery (Problème de Ramassage et de Livraison).

DVRP : Dynamic VRP (Problème de tournées de véhicules dynamique).

SPPTW : Shortest path problem with time windows (problème du plus court chemin avec fenêtres de temps).

SPPRC : Shortest path problem with resource constraints (problème de plus court chemin avec contraintes de ressources).

ESPPRC : Elementary SPPRC (SPPRC avec contrainte de chemin élémentaire).

DLC-SPPRC : Dominance on Lagrangian cost for the SPPRC.

DLC-ESPPRC : Dominance on Lagrangian cost for the ESPPRC.

INTRODUCTION GENERALE

Les problèmes d'optimisation combinatoire issus de la pratique peuvent se révéler trop complexes pour être traités de manière directe et complète par une seule technique. La difficulté principale de résolution de ces problèmes réside dans l'existence de contraintes d'intégrité où le domaine réalisable est défini par un ensemble de points discrets, difficile à caractériser explicitement. La décomposition en sous problèmes de tailles limitées et l'utilisation conjointe de plusieurs techniques de résolution offrent un recours souvent efficace. Le principe de base de la décomposition consiste à relâcher les contraintes d'intégrité sur les variables associées à un sous-ensemble de contraintes. Cela permet de diminuer la difficulté de résolution du problème mais, en contrepartie, cela détériore la valeur obtenue. En effet, la décomposition permet de fournir un minorant (resp. Majorant) sur la valeur d'un problème en minimisation (resp. Maximisation). La valeur ainsi obtenue peut-être exploitée dans une méthode de type séparation et évaluation (e.g., branch-and-bound) pour calculer une solution optimale du problème. Les formulations issues de décompositions comportent le plus souvent un nombre très important de variables, potentiellement exponentiel, ce qui les rend hors des capacités de résolution des logiciels disponibles aujourd'hui et nécessite donc des méthodes de résolution dédiées.

La génération de colonnes est une méthode de résolution adaptée aux problèmes de grande taille car elle consiste à résoudre un problème en ne considérant qu'un sous-ensemble de ses variables. De nouvelles variables sont injectées au fur et à mesure des itérations jusqu'à obtenir une base optimale. À l'instar des méthodes itératives, cette approche présente des problèmes de convergence, qui se manifestent par des changements de grande amplitude des valeurs des variables duales.

Ce travail s'inscrit dans le cadre de l'accélération de la génération de colonnes. Cette méthode est complexe et offre plusieurs possibilités d'amélioration. Dans ce travail, nous proposons quelques améliorations. Afin d'évaluer les performances des approches proposées, nous utiliserons comme problème test une variante du problème de tournées de véhicules (VRP) qui est NP-difficile [1]. Les approches les plus connues pour l'aborder se basent sur le principe de décomposition. Nous nous intéresserons à la décomposition de Dantzig et Wolfe. L'application de cette décomposition induit un problème maître de type partitionnement d'ensembles qui comporte autant de variables que de tournées possibles. Nous utilisons la génération de colonnes pour sa résolution. On considère donc un problème maître avec un ensemble restreint de tournées. Nous rajoutons ensuite de nouvelles tournées au fur et à mesure des itérations, jusqu'à obtenir une solution optimale. Une nouvelle tournée est calculée en résolvant le sous-problème, de type plus court chemin avec fenêtres de temps et contraintes de ressources. Le sous-problème fournit donc des tournées au problème maître qui les combine de manière optimale de telle sorte à ce que chaque tournée soit recouvert au moins une fois.

Le manuscrit est divisé en quatre chapitres. Le chapitre 1 présente le contexte général du VRP et en donne une définition formelle. Il décrit ensuite différentes variantes de plus en plus complexes qui sont considérées dans la littérature. Enfin, il fait un inventaire rapide des divers types de méthodes de résolution généralement utilisées pour l'aborder. Le second chapitre contient des généralités et quelques rappels pour la programmation linéaire ; abordera les techniques de décomposition, et présentera la méthode de génération de colonnes. Nous exposerons son principe, ses points forts et faibles ainsi que différentes méthodes d'amélioration qui se basent sur des principes variés, tel que l'accélération de la résolution du problème maître, du sous-problème ou bien du processus global, ou encore son hybridation avec d'autres méthodes de résolution pour tirer profit de leurs points forts. Les troisième et quatrième chapitres présentent nos contributions

qui touchent un autre aspect de l'accélération de la génération de colonnes basée sur l'amélioration de la résolution des sous-problèmes, NP-difficiles, qui correspondent à la recherche de SPPRC ou ESPPRC. Leur résolution par un algorithme de type programmation dynamique est difficile voire impossible pour des instances avec plusieurs ressources. C'est notamment le cas des problèmes de VRP où le nombre de ressources atteint fréquemment la vingtaine alors qu'en pratique, on ne peut traiter plus de cinq ressources. Ces chapitres contiennent aussi des expérimentations de notre technique sur plusieurs jeux de données. Nous terminons ce manuscrit par une conclusion et des perspectives.

CHAPITRE 1

1 PROBLEME DE TOURNEES DE VEHICULES : DEFINITION, VARIANTES ET METHODES DE RESOLUTION

1.1 Problème de base

Le problème de tournées de véhicules, ou Vehicle Routing Problem (VRP), est un problème fondamental de l'optimisation combinatoire et entière. Il s'agit de déterminer un ensemble optimal de chemins pour une flotte de véhicules devant servir un ensemble de clients (voir la figure 1.1). L'objectif est de minimiser les coûts des chemins en démarrant et terminant à un dépôt.

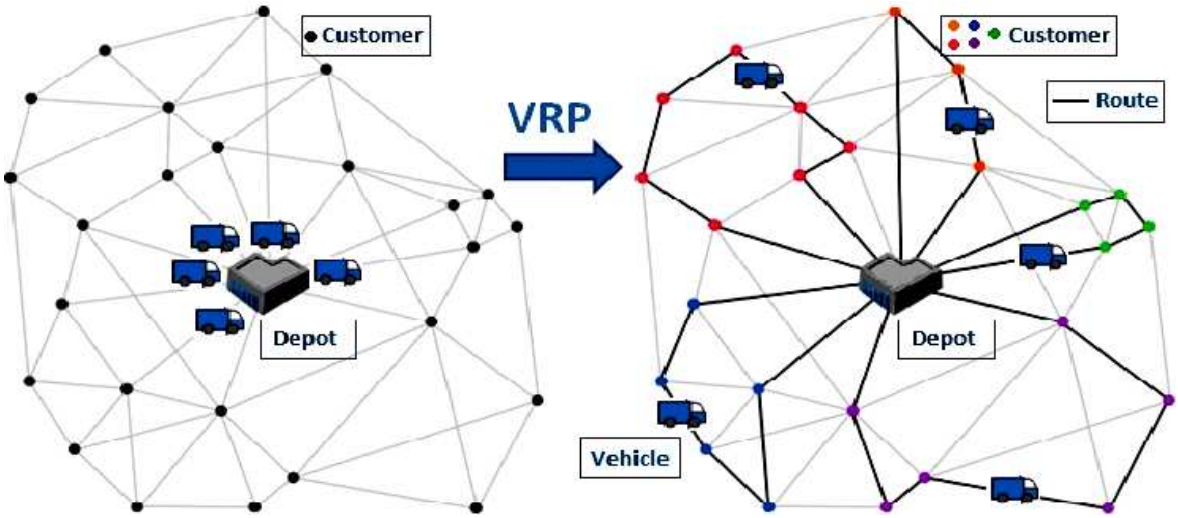

Figure 1.1: Illustration du VRP sur un réseau

La figure 1.2 présente un exemple de solution pour un problème comportant 1 dépôt (carré central) et neuf clients à visiter (nœuds C_1 à C_9). Cette solution propose 3 tournées de : (C_8, C_2, C_1), (C_4, C_6, C_9), (C_5, C_3, C_7).

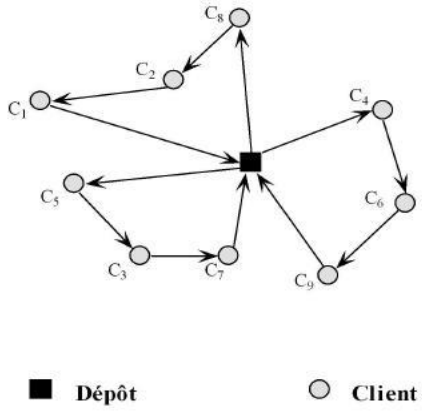

Figure 1.2: Exemple de solution du VRP

Les problèmes de tournées de véhicules sont très répandus en logistique et possèdent beaucoup d'applications importantes dans la modélisation de problèmes de production très variés, à l'instar de la planification industrielle [2], la conception des systèmes de production [3], l'automatisation des taches robotiques en industrie électronique [4], [5], l'optimisation des temps de production [6], la distribution et du transport comme le transport des biens, la livraison de marchandise ou carburant, l'alimentation des distributeurs automatiques (billets, boissons, ...), la collecte de production ou déchets, les tournées de services comme les soins, les taxis, la maintenance, ...

1.2 Variantes du problème de tournées de véhicules

La diversité des applications a amené les chercheurs à définir plusieurs versions de problèmes de tournées, adaptées à chacun des cas traités. Ces variantes sont définies en ajoutant des hypothèses, contraintes et objectifs à la version de base proposée par Dantzig et Ramser [1]. Dans ce qui suit, on présente brièvement quelques types importants. L'auteur intéressé peut consulter les références suivantes pour plus de détails [7], [8], [9], [10].

1.2.1 Problème de voyageur de commerce

Le problème du voyageur de commerce, ou Traveling Salesman Problem (TSP), proposé par Dantzig et al. en 1954 [1], [11], est un cas particulier simple du VRP. Pour un vendeur visitant un

ensemble de villes en retournant vers la même ville de départ (dépôt), le TSP cherchera une tournée (ordre de visite) de distance totale minimale. Ce problème peut être modélisé à l'aide d'un graphe dont les nœuds représentent les villes à visiter et les arcs représentent les routes liant les villes deux à deux. Les arcs sont valués avec la distance entre les villes.

1.2.2 Problème de tournées de véhicules

Le VRP est une généralisation du TSP qui considère la recherche d'itinéraires (routes) de plusieurs véhicules, à moindre coût, d'un dépôt à un ensemble de points (clients) géographiquement dispersés (villes, magasins, entrepôts, écoles, clients, machines de fabrication dans un atelier, etc.), qui minimisent le coût total (par exemple la distance totale parcourue par les véhicules). Les contraintes considérées peuvent être diverses :

- Un client ne peut être servi que par un et un seul véhicule.
- Chaque véhicule effectue une seule tournée.
- Tous les clients doivent être desservis.

1.2.3 Problème de tournées de véhicules avec contraintes de capacité

Le problème de tournées de véhicules avec contraintes de capacité ou Capacitated VRP (CVRP) [12], [8], [9] impose une contrainte de capacité (de poids, volume, ...) sur les véhicules. Il consiste à affecter chaque client à une tournée effectuée par un seul véhicule de capacité finie à ne pas dépasser par l'ensemble des clients affectés au véhicule. La demande de chaque client doit donc être inférieure à la capacité du véhicule. La contrainte peut être aussi sur d'autonomie sous forme d'une durée maximale entre le départ d'un véhicule et son retour au dépôt.

1.2.4 Problème de tournées de véhicules avec fenêtres de temps

Le problème de tournées de véhicules avec fenêtres de temps, ou VRP with Time Windows (VRPTW), affecte à chaque client une fenêtre de temps sous forme d'un intervalle durant lequel

son service est possible. Si le véhicule arriver plus tôt, il doit attendre jusqu'au début de l'intervalle du temps. S'il arrive plus tard, le service ne peut pas être accompli.

1.2.5 Problème de Ramassage et de Livraison[1]

Dans ce cas, il existe deux types de clients : les receveurs et les livreurs. Les produits livrés sont pris d'un dépôt et les produits prélevés sont retournés au dépôt. Les livraisons sont effectuées avant le premier ramassage ou bien en même temps. Aussi, un client peut être receveur et livreur en même temps. Aussi, les produits livrés peuvent être aussi pris de chez les livreurs.

1.2.6 Problème de tournées de véhicules dynamique[2]

Lorsque les informations utilisées dans le VRP ne sont pas entièrement connues dès le départ ou bien peuvent changer après démarrage de tournées initiales, le VRP est dit dynamique. C'est un cas important qui offre des flexibilités pour des applications réelles. Comme exemple, on peut citer l'apparition nouveau client ou la disparition d'un client existant.

1.3 Méthodes de résolution

Le VRP est un problème combinatoire NP-difficiles pour lequel il n'existe pas de méthode de résolution exacte pouvant obtenir une solution en un temps de calcul raisonnable lorsque le nombre de clients est grand (> 100 clients) [13]. Pour ce type de problèmes, il est nécessaire d'utiliser des méthodes approchées qui calculent une solution approximative en un temps acceptable. Le lecteur peut consulter les références suivantes pour plus de détails sur les méthodes de résolution : [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24]. La figure 1.3 présente une classification de ces méthodes.

[1] Vehicle Routing Problem with Pick-up and Delivery (VRPPD)
[2] Dynamic Vehicle Routing Problem (DVRP)

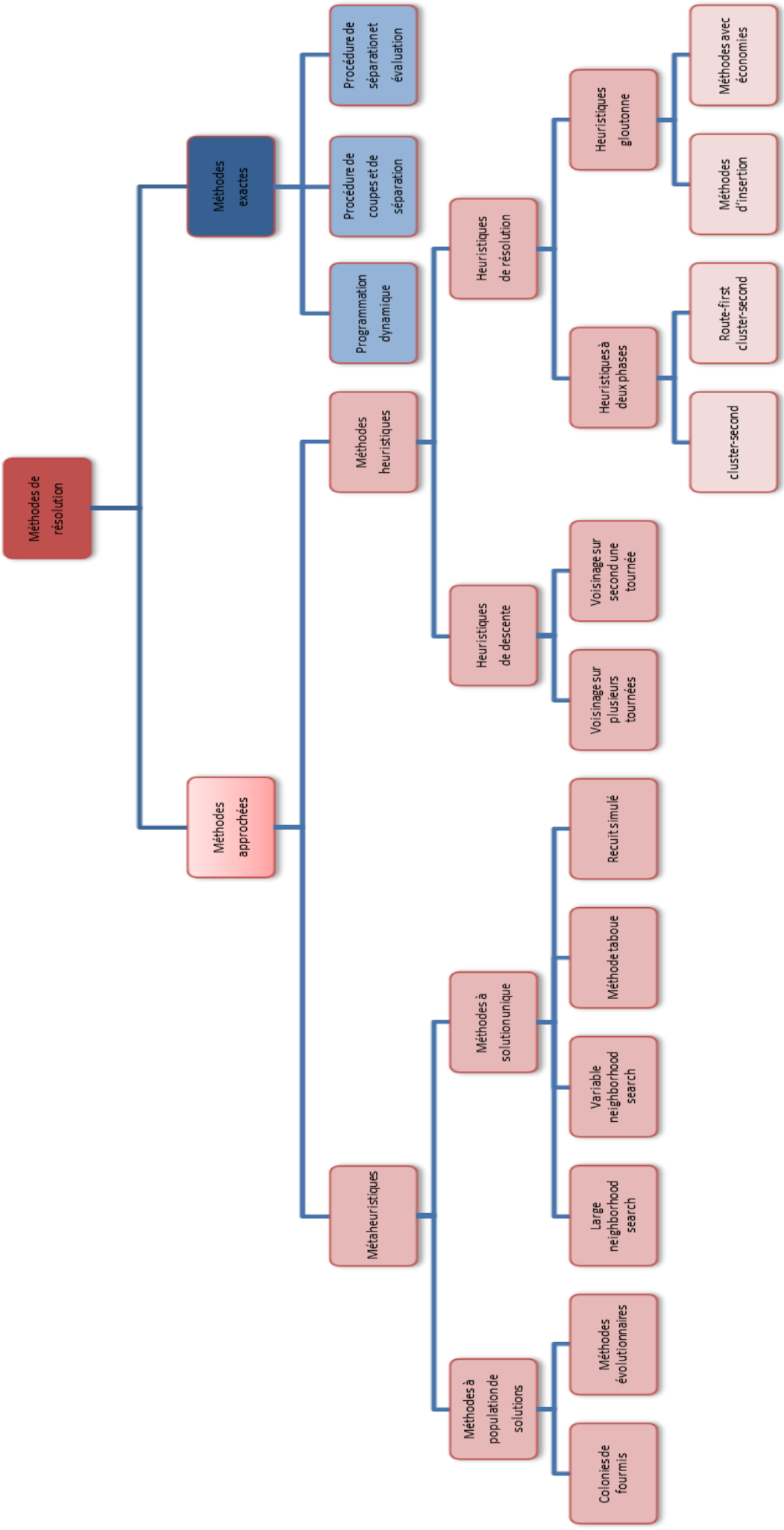

Figure 1.3: Récapitulatif des méthodes de résolution du VRP

1.3.1 Méthodes exactes

Les méthodes exactes pour le VRP peuvent être réparties en trois catégories, selon qu'elles reposent sur la séparation et évaluation (Branch and Bound), la programmation linéaire en nombres entiers (Branch and Cut), ou sur la programmation dynamique.

1.3.1.1 Branch and Bound

Cette méthode a été proposée pour la première fois par Land et Doig, en 1960 [25]. Elle construit un arbre de recherche représentant l'espace des solutions tout en supprimant des branches inutiles contenant des solutions non intéressantes ou non réalisables.

Les solutions exactes sont obtenues progressivement, arc par arc. L'exploration de l'arbre se fait avec des évaluations des branches et des comparaisons avec une borne ou une valeur seuil du critère à optimiser. L'obtention d'une borne de bonne qualité permet d'accélérer les temps de calcul car elle permet de réduire la taille de l'arbre. Ces méthodes sont limitées à des VRP de petites tailles car les branches, et donc le temps de calcul, augmentent considérablement avec la taille du problème.

1.3.1.2 Branch and Cut

Branch and Cut est une généralisation de Branch and Bound, utilisée lorsque le nombre de contraintes est élevé. Elle a été proposée en 1987 par Padberg et Rinaldi [26] pour un TSP. Elle consiste à combiner une méthode de génération de coupes avec celle de Branch-and-Bound, pour améliorer la borne inférieure du modèle linéaire en nombres entiers. À chaque nœud de l'arbre de recherche, la méthode de génération de coupes résout la relaxation, et ajoute itérativement de nouvelles inégalités valides violées par la solution non entière de la relaxation pour améliorer la solution courante.

1.3.1.3 Programmation Dynamique

La programmation dynamique est une méthode d'optimisation basée sur l'approche «diviser et régner». Elle a été introduite par Bellman dans les années 50 et appliquée pour la première fois au VRP par Eilon et al. en 1971 [26]. Elle est utilisée pour des VRP de très petites tailles allant de 10 à 25 clients [27]. Lorsqu'une solution optimale peut être représentée comme une combinaison de solutions optimales de sous-problèmes, la programmation dynamique résout chaque sous-problème une seule fois et stocke les solutions de tous les sous-problèmes rencontrés dans une matrice. Pour résoudre le problème principal, elle exploite une relation de récurrence entre les sous-problèmes.

1.4 Méthodes approchées

Les méthodes exactes sont souvent limitées à des problèmes de petites tailles, vu leur cout calculatoire élevé. Pour des problèmes réels de grandes taille, l'utilisation de méthodes approchées s'impose. Ces méthodes permettent d'obtenir des solutions approximatives, de bonne qualité, en un temps raisonnable. On peut les subdiviser en deux classes : les heuristiques et les métaheuristiques. Grosso modo, une heuristique invente une technique adaptée à un problème particulier ; tandis que les "métaheuristique" visent à résoudre des classes générales de problèmes mathématiques en combinant des procédures de recherche pour trouver rapidement une bonne approximation de la meilleure solution ; et notamment une solution globale permettant d'éviter le problème des minimas locaux.

1.4.1 Heuristiques

Le principe d'une méthode heuristique est de trouver, en un temps raisonnable une solution de bonne qualité. On peut distinguer trois grandes familles d'heuristiques : méthodes constructives, méthodes de recherches locales et méthodes en deux phases.

Les méthodes constructives construisent une seule solution progressivement, par une suite de choix partiels et définitifs et sans retours en arrière, sans amélioration. Par exemple, on peut citer la méthode des gains [28] et la méthode d'insertion [29], [20].

Les méthodes de recherches locales, appelées aussi recherches de voisinage, considèrent des améliorations itératives. Pour les problèmes de tournées, ces méthodes déplacent un client d'une première tournée vers une deuxième. Cette dernière est modifiée en transférant un client vers une troisième tournée ; et ainsi de suite jusqu' à que le transfert arrive à une dernière tournée, qui elle-même transfère un client vers la première tournée [30].

Les méthodes en deux phases sont basées sur une décomposition du problème sous-groupes et la détermination de la tournée associée à chaque sous-groupe [31]. Plusieurs applications ont été proposées pour les VRPs [32], [33].

1.4.2 Métaheuristiques

Il existe de nombreux états de l'art sur l'application des métaheuristiques aux VRP [18], [19], [21], [22], [23]. Les métaheuristiques peuvent être subdivisées en trois classes : métaheuristiques basées sur la recherche locale, métaheuristiques basées sur la population et métaheuristiques hybrides.

Les métaheuristiques basées sur la recherche locale font appel à la notion de voisinage qui offre un bon compromis entre efficacité et qualité. En effet, si un voisinage large est considéré, on a de fortes chances de pouvoir échapper aux optimums locaux au détriment du temps de calcul. Par contre, si le voisinage est très restreint, le temps de calcul est réduit mais la probabilité d'être piégé dans un minimum local est élevée. La méthode de recherche locale la plus élémentaire est la méthode de descente. On peut citer aussi le Recuit [34] et ses applications aux VRPs [27], [35], [36] ; la méthode du Tabou [37] avec ses applications aux VRPs [38], [39], [40], [41], [35] ,

la méthode de recherche à voisinages variables [42] et ses applications aux problèmes de tournées [43] , et la méthode GRASP (Greedy Randomized Adaptive Search Procedure) [44], [45], …

Contrairement aux méthodes précédentes, les métaheuristiques basées sur la population traitent une population de solutions globalement. La coopération entre les individus (solutions) est exploitée, à chaque itération, pour construire une nouvelle population à partir de la précédente, et évoluer vers un ensemble de solutions les plus adaptées. Parmi ces méthodes, on peut essentiellement distinguer les algorithmes évolutionnaires, les algorithmes de colonies de fourmis, et la recherche par dispersion.

Les algorithmes évolutionnaires sont les plus utilisés avec succès pour résoudre des problèmes d'optimisation difficiles [46], [47], [48], [49] ... Ils utilisent itérativement des processus aléatoires combinant les bonnes propriétés d'un ensemble de solutions (une population).

Les colonies de fourmis imitent le comportement réel des fourmis qui leur permet de trouver les plus courts chemins entre les sources de nourriture et leur nid [50]. Une famille d'algorithmes de colonies de fourmis a été proposée dans [51], puis dans [52]. L'idée consiste à travailler sur une population de solutions où chaque fourmi se déplace sur l'espace de recherche. Une structure de donnée commune, partagée par toutes les fourmis, contient l'information sur la phéromone accumulée dans cet espace. Les fourmis marquent les meilleures solutions, et tiennent compte des marquages précédents pour optimiser leur recherche.

La recherche par dispersion, ou « Scatter Search », a été proposée par [53]. Un ensemble de solutions dispersées de bonne qualité est généré à partir d'un ensemble de référence. Pour ce faire, de nouvelles solutions candidates sont créées par combinaisons linéaires de solutions de l'ensemble de référence ; ce qui permettra d'obtenir la diversification. Ensuite, une procédure de réparation est appliquée sur chaque solution candidate non admissible pour obtenir un ensemble de solutions admissibles. Une recherche locale est appliquée qui permet d'obtenir des solutions

dispersées. Enfin, le nouvel ensemble de solutions est sélectionné à partir des solutions dispersées générées et de l'ensemble précédent de référence. Ces étapes sont répétées jusqu'à satisfaction d'un critère d'arrêt.

1.4.3 Méthodes hybrides

Les méthodes hybrides combinant plusieurs heuristiques ont prouvé leur efficacité ; en particulier la combinaison entre la recherche locale et les méthodes basées sur la population. L'hybridation permet d'exploiter les avantages de plusieurs types de méthodes. Par exemple, une méthode évolutive permet une exploration globale de l'espace de recherche pour détecter de bonnes régions, alors qu'une recherche locale explore efficacement les régions prometteuses. L'hybridation peut aussi se faire des méthodes exactes. Par exemple, une métaheuristique peut fournir des bornes à une méthode de type Branch and Bound. Toutefois, les temps de calcul nécessaires des méthodes hybrides peuvent devenir prohibitifs à cause du nombre d'individus manipulés dans la population. Pour résoudre ce problème, on peut appliquer la parallélisation de ces algorithmes sur des machines parallèles [54], [55]. Quelques exemples de métaheuristiques hybrides peuvent être trouvés dans [56], [57], [58], [59], [60] pour le problème du voyageur de commerce, et pour le problème de tournée de véhicules [61], [62], [63].

Une description plus détaillée de cette métaheuristique est donnée dans [64], [65]. Plusieurs applications des métaheuristiques pour le VRP peuvent être trouvées dans les références suivantes : [66], [67], [68] et pour les états de l'art et livres [69], [70], [71], [72].

1.5 Conclusion

Dans ce chapitre, nous avons passé en revu quelques applications importantes des VRPs, en particulier en planification industrielle, le transport et la logistique, la collecte et la distribution, ... Cette diversité d'applications a conduit à définir plusieurs variantes du VRP dont on a proposé quelques-unes qui reposent une grande variété d'hypothèses, contraintes ou objectifs. Cela a rendu les VRPs plus difficiles à résoudre malgré le grand nombre de solutions proposées. Les VRPs deviennent particulièrement difficiles pour les problèmes de grandes tailles.

CHAPITRE 2

2 TECHNIQUES ET METHODOLOGIE DE LA PROGRAMMATION LINEAIRE ET GENERATION DE COLONNES

2.1 Généralités et rappels

La programmation linéaire représente une partie importante de la programmation mathématique. Elle consiste à modéliser des problèmes issus d'applications réelles en un langage formel et des méthodes pour les résoudre. Durant la seconde guerre mondiale, George Dantzig formula en tant que problèmes linéaires (PL), de nombreux problèmes militaires, relatifs à l'organisation et à la distribution des munitions et des vivres, ainsi qu'aux déploiements des troupes militaires. Puis, en 1947, il met au point un algorithme pour résoudre les problèmes linéaires, connus sous le nom de l'algorithme de Simplexe. Ce dernier, est considéré comme une méthode de points frontières puisqu'il projette de faire des déplacements le long des arêtes du polyèdre défini par les contraintes du (PL) jusqu'à atteindre l'optimum. La découverte et l'implémentation de l'algorithme de Simplexe, à cette époque, contribua grandement au renforcement de la valeur de la programmation mathématique. Cependant, s'il se révèle très efficace en moyenne, Klee et Minty [73] montrèrent en 1972 que cet algorithme n'est pas polynomial. Ceci a suscité un regain d'intérêt pour la recherche d'algorithmes polynomiaux de résolution des programmes linéaires.

La méthode ellipsoïde, développée en 1979 par Khachian [74], fut le premier algorithme de ce type montrant alors que les problèmes de programmation linéaires sont polynomiaux. Son idée principale consiste à utiliser une suite d'ellipsoïdes de volume décroissant mais contraint de contenir la solution optimale du problème à résoudre à chaque itération. Bien que plus rapide que l'algorithme de Simplexe sur les problèmes de Klee et Minty, la méthode des ellipsoïdes reste bien plus lente sur les problèmes réels. En 1984, un autre algorithme polynomial basé sur les principes de géométrie projective et de programmation non linéaire a été développé par Karmakar [75]. Il s'agit d'une méthode de points intérieurs dont le principe est de chercher des points, à l'intérieur du

polyèdre des contraintes, permettant de se diriger rapidement vers le sommet optimal. Depuis la découverte de cet algorithme, la recherche dans le domaine de la programmation mathématique a connu un nouvel élan et plusieurs algorithmes tels que la méthode de barrière dérivée de ce dernier virent le jour.

Actuellement, ces méthodes commencent à concurrencer l'algorithme du Simplexe sur certains problèmes de grande taille car elles ont tendance à nécessiter moins d'itérations que ce dernier bien que chaque itération soit plus longue. Néanmoins, l'algorithme du Simplexe demeure, non seulement un outil très performant pour la résolution des (PL) mais, en outre, il se distingue exclusivement par la richesse de l'interprétation géométrique et économique qu'il fournit ainsi qu'à sa capacité à fournir des solutions de base, très importantes dans les approches de décomposition ou encore dans des procédures de ré-optimisation itératives (Lebbar [76]).

2.1.1 Programme linéaire

Un programme linéaire(PL) est un problème d'optimisation consistant à maximiser ou minimiser une fonction linéaire dite fonction objectif sous contraintes linéaires exprimées sous forme d'équations ou d'inéquations. Un tel programme peut être représenté sous forme matricielle comme suit :

$$(PL) \equiv \begin{cases} \text{opt } cx \\ Ax \geq b \\ x \geq 0 \end{cases} \qquad \textbf{(2.1)}$$

avec les notations suivantes :

- n : nombre de variables,
- m : nombre de contraintes,
- $A = (a_{ij}), i = 1, \ldots, m \; ; \; j = 1, \ldots, n$: matrice réelle de format $(m \times n)$,
- $c = (c_1, \ldots, c_n)$: vecteur ligne des coûts (profits),
- $b = (b_1, \ldots, b_m)^t$: vecteur colonne des seconds membres,

- $x = (x_1, \ldots, x_n)^t$: vecteur colonne de variables,

- Sans pertes de généralité nous considérons $opt \equiv min$ pour le problème (PL).

Terminologie de la solution d'un programme linéaire

Une solution d'un programme linéaire est une affectation de valeurs aux variables du problème. Elle est dite réalisable lorsqu'elle satisfait toutes les contraintes du problème. L'ensemble de toutes les solutions réalisables s'appelle zone ou domaine de solution.

Une solution notée par x^* est optimale lorsqu'elle est réalisable et la fonction objectif atteint sa valeur optimale z_{PL}^*. Cette solution n'est pas nécessairement unique.

Problème dual

À chaque programme linéaire (PL) nommé programme linéaire primal, est associé un programme linéaire dual (DPL), défini comme suit :

$$(DPL) \equiv \begin{cases} \max ub \\ uA \leq c \\ u \geq 0 \end{cases} \qquad \textbf{(2.2)}$$

où le vecteur $u = (u_1, \ldots\ldots, u_m)$ est un vecteur ligne appelé vecteur des variables duales tel que chaque variable u_i $(i = 1, \ldots, m)$ est associée à une contrainte i du problème (PL).

Notons qu'il existe plusieurs liens entre le problème primal (PL) et son problème dual (DPL) dont nous citons quelques-uns dans les théorèmes suivants [77].

Théorème 1 Etant donnés deux programmes linéaires duaux (PL) et (DPL). Si (PL) n'admet pas de solution réalisable alors soit (DPL) n'admet pas de solution réalisable, soit (DPL) est non borné.

Théorème 2 (Théorème de dualité)

- Si x et u sont respectivement des solutions réalisables pour (PL) et (DPL), alors on a :

$$cx \geq ub.$$

- Si (PL) et (DPL) ont des solutions, alors chacun d'eux a une solution optimale et :

$$z_{PL}^{*} = z_{DPL}^{*} \tag{2.3}$$

-Si (PL) est non borné alors (DPL) n'admet pas de solution réalisable.

Le théorème de dualité est l'un des résultats fondamentaux de la programmation linéaire les plus importants, vu son intérêt théorique et pratique. Pour plus de détails sur la programmation linéaire, le lecteur est encouragé à consulter Chvàtal [77].

2.1.2 Programmation linéaire en nombres entiers

Plusieurs problèmes d'optimisation combinatoire se formulent sous forme de programmes linéaires en nombres entiers $(PLNE)$. Ils sont souvent difficiles à résoudre, du fait notamment que l'espace de recherche n'est plus convexe mais discret. La formulation générale d'un $(PLNE)$ est la suivante :

$$(PLNE) \equiv \begin{cases} \text{opt } cx \\ Ax \geq b \\ x \in \mathbb{N}^n \end{cases} \tag{2.4}$$

Sans pertes de généralité, nous considérons $opt \equiv min$ pour le problème $(PLNE)$.

Dans le cas particulier, ou les variables prennent seulement les valeurs (0 et 1), on parle de programme linéaire en variables binaires ou booléennes. Dans le cas général, fréquemment rencontré, où une partie des variables seulement sont astreintes à être entières, on parle de programme linéaire en variables mixtes.

Résolution des programmes linéaires en nombres entiers

Les méthodes classiques de programmation linéaire ne peuvent être utilisées systématiquement pour les $(PLNE)$ puisqu'elles cherchent un sommet optimal du polyèdre des contraintes $\{x|Ax \geq b, x \geq 0\}$ qui, en général, n'a pas de coordonnées entières. Différentes approches ont été étudiées pour la recherche de solutions optimales entières des $(PLNE)$. Elles font appel

essentiellement à des algorithmes de recherche arborescente par séparation et évaluation (branch and bound) [78] et à des méthodes de coupes [79].

Un moyen naturel pour résoudre un $(PLNE)$ de très petite taille consiste à calculer l'ensemble des solutions entières du problème et à en retenir la meilleure. Or, au-delà de quelques variables, cette énumération explicite devient impossible du fait de la combinatoire du problème. L'idée de la recherche arborescente par séparation et évaluation est d'effectuer une énumération implicite des solutions du $(PLNE)$.

Cette méthode consiste à séparer le problème en plusieurs sous-problèmes, puis à évaluer chaque sous-problème en calculant une borne inférieure (en minimisation) de sa valeur optimale et en la comparant à une valeur de référence (correspondant à une solution connue du problème). On itère ensuite le processus uniquement sur les sous-problèmes dont l'évaluation est supérieure à la valeur de référence (ils sont seuls susceptibles de l'améliorer). L'ensemble des solutions est ainsi représenté par une arborescence dans laquelle un grand nombre de nœuds sont éliminés.

2.1.3 Méthodes de relaxation

Lors de la résolution exacte d'un problème linéaire en nombres entiers, le calcul de bornes inférieures (dans le cas d'un problème de minimisation) qui peuvent être utilisées dans des algorithmes de séparation et évaluations par exemple, est souvent nécessaire. L'obtention d'une borne inférieure et d'une borne supérieure permet également d'avoir un encadrement de la valeur optimale du problème. Une approche classique en recherche opérationnelle pour obtenir une borne inférieure d'un problème en nombres entiers est la relaxation. Les méthodes de relaxation sont généralement utilisées afin d'estimer un minorant de la valeur optimale du problème $(PLNE)$ considère en se ramenant sur des problèmes pour lesquelles des méthodes de résolutions efficaces existent. Dans cette section nous présentons les principales relaxations existantes.

2.1.3.1 La relaxation continue (linéaire)

La relaxation continue de $(PLNE)$ que l'on note $(\overline{PLNE})$, consiste à relâcher la contrainte d'intégralité des variables :

$$(\overline{PLNE}) \equiv \begin{cases} \min cx \\ Ax \geq b \\ x \geq 0 \end{cases} \quad \textbf{(2.5)}$$

Le problème $(\overline{PLNE})$ est résolu grâce à l'utilisation de la méthode du Simplexe [80], [81]. Si les variables sont bornées, il existe une variante de la méthode du simplexe qui permet de diminuer l'occupation mémoire nécessaire pour la résolution numérique de $(\overline{PLNE})$ [82].

La qualité de cette borne inférieure n'est cependant pas toujours très bonne, et elle dépend du saut de dualité du problème, c'est-à-dire de l'écart entre la valeur de cette borne et la valeur optimale du problème.

Il existe d'autres types de relaxations plus coûteuses mais plus précises. Il s'agit des relaxations Lagrangienne, surrogate, et composite principalement.

2.1.3.2 La relaxation lagrangienne

Une méthode très employée, qui sera utilisée dans le cadre de cette thèse est la relaxations Lagrangienne. Elle consiste à dualiser les contraintes qui compliquent le problème, i.e., les intégrer à la fonction objectif du problème avec une certaine pénalité pour générer un problème plus facile à résoudre. Considérons le problème $(PLNE)$, on définit la fonction lagrangienne correspondant à cette modélisation par :

$$L_x(u) = cx + u(b - Ax) \text{ avec } u \geq 0, \quad \textbf{(2.6)}$$

et la fonction duale par :

$$\Theta(u) = \min_x L_x(u) \quad \textbf{(2.7)}$$

Le dual de la relaxation lagrangienne se formule de la manière suivante :

$$(DL) \equiv \begin{cases} \max \ \Theta(u) \\ u \geq 0 \end{cases} \quad \textbf{(2.8)}$$

La valeur du dual Lagrangien est toujours au moins aussi bonne que la valeur de la relaxation en continu du problème. Cette relaxation a été prouvée comme étant un outil efficace pour la résolution de problèmes en nombres entiers [83].

2.1.3.3 Relaxation surrogate ou agrégée

Une autre solution approchée peut être d'exterminée par la résolution du $(PLNE)$ obtenue par la relaxation surrogate de $(PLNE)$, notée $(SPLNE(\lambda))$ introduite pour la première fois par Glover [84]. La relaxation surrogate consiste à combiner l'ensemble des contraintes en une seule de la manière suivante :

$$\big(SPLNE(\lambda)\big) \equiv \begin{cases} \min cx \\ \lambda Ax \geq \lambda b \ \ ou \ \lambda \geq 0 \\ x \in \mathbb{N}^n \end{cases} \quad \textbf{(2.9)}$$

Le dual de la relaxation surrogate se formule de la manière suivante :

$$\big(DSPLNE(\lambda)\big) \equiv \begin{cases} \max(SPLNE(\lambda)) \\ \lambda \geq 0 \end{cases} \quad \textbf{(2.10)}$$

Cette relaxation a fait l'objet de nombreuses études afin de proposer des méthodes de calcul du multiplicateur et son utilisation pour la résolution du problème $(PLNE)$. Greenberg et Pierskalla [85] ont montré que la valeur du dual surrogate est toujours au moins aussi bonne que la valeur de la relaxation en continu.

2.1.3.4 La relaxation Composite

La relaxation composite pour le problème $(PLNE)$ est la combinaison de la relaxation lagrangienne et de la relaxation surrogate. Elle fut introduite par Greenberg and Pierskalla [85] et définie comme suit :

$$\big(C(u,\lambda)\big) \equiv \begin{cases} \min cx + u(b - Ax) \\ \lambda Ax \geq \lambda b \qquad ou\ \lambda \geq 0 \\ x \in \mathbb{N} \end{cases} \tag{2.11}$$

Le dual de la relaxation composite est obtenu par :

$$\big(DC(u,\lambda)\big) \equiv \begin{cases} \max(C(u,\lambda)) \\ u \geq 0\ , \lambda \geq 0 \end{cases} \tag{2.12}$$

La relaxation lagrangienne et la relaxation surrogate sont clairement des cas particuliers de la relaxation composite. La valeur du dual composite est également toujours au moins aussi bonne que la valeur de la relaxation en continu.

2.1.3.5 Choix des multiplicateurs

L'utilisation de ces différentes méthodes de relaxation nécessite le calcul des multiplicateurs. Un multiplicateur λ est dit meilleur qu'un multiplicateur λ' lorsqu'il fournit une meilleure borne inférieure de la valeur optimale. Autrement dit, le multiplicateur optimal est celui qui maximise la borne inférieure. Les multiplicateurs optimaux sont obtenus par résolution des problèmes duaux. Finalement, une comparaison des résultats obtenus entre les quatre relaxations est présentée :

$$z^*_{(\overline{PLNE})} \leq z^*_{(DL)} \leq z^*_{(DSPLNE)} \leq z^*_{(DC)} \leq z^*_{(PLNE)} \tag{2.13}$$

2.1.4 Programmes linéaires de grandes tailles

De nombreux problèmes d'optimisation combinatoire peuvent être modélisés comme des $(PLNE)$, mais leurs tailles, ainsi que leurs structures empêchent fréquemment une résolution par les méthodes usuelles de la programmation linéaire. En effet, la relaxation linéaire de ces modèles fournit souvent une borne de mauvaise qualité qui, ajoutée à la combinatoire du problème, rend la recherche arborescente par séparation et évaluation d'une solution optimale entière, très difficile voire même impossible en un temps raisonnable. Une solution à ce problème consiste à utiliser des méthodes de décomposition (décomposition de Benders [86] ou décomposition de Dantzig-Wolfe [87]) donnant lieu à de nouveaux modèles plus efficaces et fournissant de meilleures bornes.

En contrepartie, ces modèles nécessitent l'utilisation d'algorithmes complexes : génération de coupes pour la décomposition de Benders (nombre exponentiel de contraintes) et génération de colonnes pour la décomposition de Dantzig-Wolfe (nombre exponentiel de variables). Notons que le choix de la méthode de décomposition employée dépend strictement de la structure du problème linéaire, c'est-à-dire la nature de ses composantes (contraintes et variables) et les liens existants entre elles. En général, les programmes linéaires de grande taille sont caractérisés par leurs matrices de contraintes qui sont "très creuses" et les éléments nuls sont distribués de telle façon qu'ils forment de grandes sous-matrices (voir Minoux [88] pour plus de détails).

2.2 Méthodes de décomposition et génération de colonnes

Les programmes linéaires en nombres entiers sont généralement NP-difficiles. Des bornes sur la valeur optimale du problème sont souvent nécessaires pour la résolution avec des algorithmes exacts, telles que les méthodes de séparation et d'évaluation. Une borne peut être obtenue avec la relaxation continue du problème traité, mais cette borne est parfois de très mauvaise qualité. À cet effet, plusieurs méthodes de décomposition sont proposées pour réaliser un bon compromis entre la qualité de l'approximation et la difficulté de la résolution.

Nous présentons dans cette section quelques méthodes de décomposition parmi les plus étudiées dans la littérature. Le principe de ces méthodes consiste à relâcher les contraintes d'intégrité sur un sous-ensemble de contraintes. Cela permet d'une part de diminuer la difficulté de la résolution et d'autre part d'obtenir une borne sur la valeur optimale du problème. Les méthodes de décomposition présentées ici se basent sur différents principes. Nous présentons pour chacune d'entre elles, les méthodes de résolution dédiées. Nous étudions à la fin de ce chapitre les liens existants entre les différentes méthodes de décomposition exposées.

La décomposition peut engendrer des problèmes avec un nombre exponentiel de variables qui ne peuvent être résolus avec les méthodes standards comme l'algorithme du simplexe par exemple. Une méthode adaptée pour la résolution de problèmes de grandes tailles issus d'une décomposition, et très utilisée actuellement, est la génération de colonnes. Elle se base sur le principe d'une réduction de la taille du problème traité qui est augmenté au fur et à mesure, jusqu'à trouver une solution optimale. Cette méthode a prouvé son efficacité pour la résolution de divers problèmes ; cependant, elle est connue pour sa mauvaise convergence. Nous étudions aussi dans ce qui suit différentes méthodes d'amélioration de la génération de colonnes.

2.2.1 Méthodes de décomposition pour l'optimisation discrète

Les méthodes de décomposition sont généralement utilisées dans les processus de résolution (minimisation par exemple) de problèmes en nombres entiers ($PLNE$). Elles consistent à décomposer le problème en sous-problèmes plus faciles à résoudre. La décomposition entraîne la relaxation de certaines contraintes et permet donc de trouver une borne sur la valeur du ($PLNE$). Les bornes obtenues sont ensuite utilisées pour guider et accélérer la phase d'exploration par un algorithme d'énumération de type branch-and-bound [78].

Toutes les méthodes de décomposition se basent sur le même principe. Le problème traité est décomposé en un ou plusieurs sous-problèmes, généralement coordonnés par un problème maître.

Les méthodes de résolution des nouvelles formulations induites par la décomposition alternent entre une procédure de calcul d'un minorant par la résolution du problème maître, et une procédure qui améliore l'approximation courante par la résolution des sous-problèmes. Autrement dit, le principe est d'approcher l'enveloppe convexe du domaine réalisable C du $(PLNE)$. Une approximation classique est celle fournie par une relaxation continue. La borne obtenue par cette relaxation peut s'avérer très éloignée de la valeur optimale.

Nous présentons d'abord le principe général de la décomposition, puis quelques techniques de décomposition et les méthodes de résolution associées.

2.2.1.1 Principe de la décomposition

Soit le programme linéaire suivant :

$$(PLNE) \equiv \begin{cases} \quad\quad min\ c\ x \quad\quad avec \quad c \in \mathbb{R}^n \\ Ax \geq a \quad (m_1\ contraintes)\ avec\ A \in \mathbb{R}^{m_1 \times n}\ et\ a \in \mathbb{R}^{m_1} \\ Bx \geq b \quad (m_2\ contraintes)\ avec\ B \in \mathbb{R}^{m_2 \times n}\ et\ b \in \mathbb{R}^{m_2} \\ \quad\quad x \in \mathbb{N}^n \end{cases} \tag{2.14}$$

La résolution de $(PLNE)$ peut nécessiter l'utilisation d'une procédure de type branch-and-bound, qui sera d'autant plus efficace si l'on dispose de bornes sur la valeur de $(PLNE)$. La procédure la plus simple est de résoudre la relaxation continue de $(PLNE)$ que l'on note $(\overline{PLNE})$. et qui est obtenue en relâchant les contraintes d'intégrité de $(PLNE)$. Soit la relaxation continue de $(PLNE)$:

$$(\overline{PLNE}) \equiv \begin{cases} min\ c\ x \\ Ax \geq a \\ Bx \geq b \\ x \epsilon \mathbb{R}^n \end{cases} \tag{2.15}$$

En relâchant les contraintes d'intégrité, on obtient généralement un problème plus facile à résoudre, mais $z^*_{(\overline{PLNE})}$ peut-être éloignée de z^*_{PLNE}. Nous devons donc considérer des procédures

plus efficaces. Pour améliorer la borne $z^*_{(\overline{PLNE})}$, les méthodes de décomposition construisent un nouveau polyèdre d'optimisation qui intersecte $\overline{C}$ pour obtenir une meilleure approximation. Ce nouveau polyèdre possède une description de taille exponentielle, nous devons donc générer des portions de sa description de manière dynamique.

Notons $conv(X)$ l'enveloppe convexe de l'ensemble X.

Soit $C' = conv(\{x \in \mathbb{N}^n \setminus Ax \geq a\})$, $C'' = conv(\{x \in \mathbb{N}^n \setminus Bx \geq b\})$,

$$\overline{C}' = \{x \in \mathbb{R}^n \setminus Ax \geq a\}, \overline{C}'' = \{x \in \mathbb{R}^n \setminus Bx \geq b\}$$

On peut donc écrire :

$$C = C' \cap C'' \text{ et } \overline{C} = \overline{C}' \cap \overline{C}''.$$

La décomposition consiste à relâcher les contraintes d'intégrité sur un sous-ensemble de contraintes. Cela revient à résoudre $(PLNE)$ sur le domaine $D_1 = C' \cap \overline{C}''$ ou bien $D_2 = C'' \cap \overline{C}'$ (voir la figure Figure 2.1). Dans ces deux cas, la décomposition n'a un sens que si $D1 \subset \overline{C}$ (resp. $D2 \subset \overline{C}$), de manière à obtenir une borne au moins aussi bonne que $z^*_{(\overline{PLNE})}$ s'il existe un algorithme efficace d'optimisation sur le domaine C' (resp. C'').

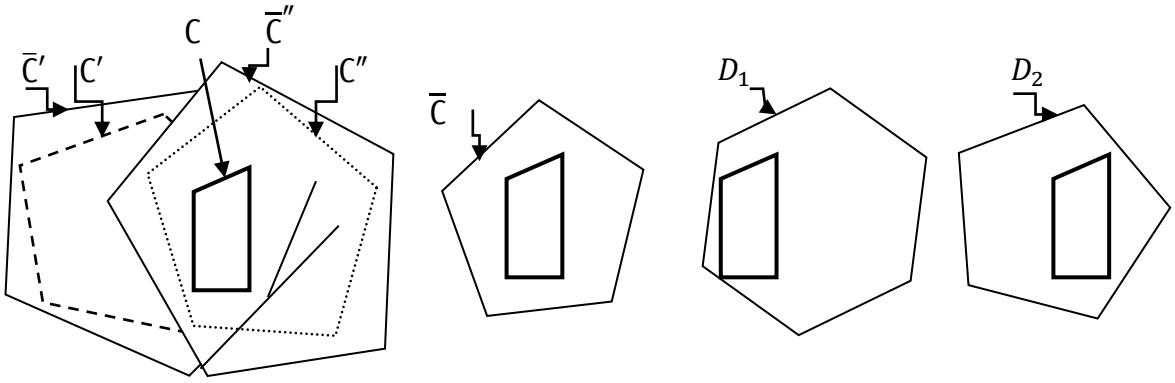

Figure 2.1: Illustration de différentes approximations du domaine C

2.2.1.2 *Décomposition lagrangienne*

La décomposition lagrangienne [89], [90] est utilisée lorsque la conjonction entre deux blocs de contraintes complique le problème. Pour le problème $(PLNE)$, la conjonction des contraintes

$Ax \geq a \; et \; Bx \geq b$ peut rendre sa résolution difficile. Cette méthode consiste alors à découpler les contraintes en introduisant des variables de copie.

Soit le problème équivalent :

$$(PLNE) \equiv \begin{cases} min\ c\ x \\ Bx \geq b \quad x \in \mathbb{N}^n \\ x = y \\ Ay \geq a \quad y \in Y \end{cases} \tag{2.16}$$

où $Y \subset \mathbb{N}^n$. En dualisant les contraintes de copie $x = y$ suivant le multiplicateur $u \in \mathbb{R}^n$, on obtient le lagrangien

$$L_{x,y}(u) = min\,\{c\,x - u(x-y) \mid Ay \geq a, y \in Y, Bx \geq b, x \in \mathbb{N}^n\} \tag{2.17}$$

qui peut être formulé comme suit : $L_{x,y}(u) = L_x(u) + L_y(u)$ où

$L_x(u) = min\,\{(c-u)x \mid Bx \geq b, x \in \mathbb{N}^n\}$ et $L_y(u) = min\,\{uy \mid Ay \geq a, y \in Y\}$.

Le dual de la décomposition lagrangienne est donné par :

$$(DDL) \equiv \begin{cases} z_{DDL} = max\ L_{x,y}(u) \\ u \in \mathbb{R}^n \end{cases} \tag{2.18}$$

Remarque 1

(a) La décomposition lagrangienne est une relaxation lagrangienne sur un problème transformé qui décompose le problème initial en une série de sous-problèmes.

(b) La valeur du problème (DDL) est égale à la valeur optimale du problème

$$min\,\{cx \mid x \in Z_x \cap Z_y\}$$

où $Z_x = conv\{x \in \mathbb{N}^n | Bx \geq b\}$ et $Z_y = conv\{y \in Y | Ay \geq a\}$.

La figure Figure 2.2 représente ces deux domaines.

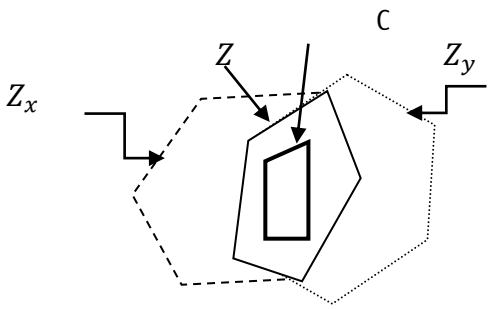

Figure 2.2: Illustration de l'approximation du domaine C avec la décomposition lagrangienne

Cette méthode a été appliquée pour la résolution du problème de localisation avec capacités dans [90]. Les bornes obtenues sont de bonne qualité, mais la résolution est coûteuse en temps. Nous exploitons dans ce qui suit la génération de colonnes pour résoudre le dual de la décomposition lagrangienne. Les lagrangiens $L_x(u)$ et $L_y(u)$ peuvent être réécrits comme suit:

$$L_x(u) = \{max\ w_1 | w_1 \leq (c - u)x, x \in Z_x\}$$

$$= \{max\ w_1 | w_1 \leq (c - u)x_k, k \in \mathcal{K}\}$$

et $$L_y(u) = \{max\ w_2 | w_2 \leq uy, y \in Z_y\}$$

$$= \{max\ w_2 | w_2 \leq uy_l, l \in L\}$$

où $\mathcal{K}$ (resp.L) désigne l'ensemble des indices des points extrêmes du polyèdre Z_x (resp.Z_y).

Supposons que les ensembles Z_x et Z_y sont convexes et bornés, le problème (DDL) peut être alors linéarisé comme suit :

$$(DDL) \equiv \begin{cases} max\ \ w_1 + w_2 \\ w_1 \leq (c-u)x\,, \quad x \in Z_x \\ w_2 \leq uy\,, \quad y \in Z_y \\ w_1\,, w_2 \in \mathbb{R} \\ u \in \mathbb{R}^n \end{cases} \equiv \begin{cases} max\ w_1 + w_2 \\ w_1 \leq (c-u)x_k\,, \ k \in K \\ w_2 \leq uy_l\,, \ \ l \in L \\ w_1\,, w_2 \in \mathbb{R} \\ u \in \mathbb{R}^n \end{cases} \quad \textbf{(2.19)}$$

Dans la décomposition lagrangienne, les multiplicateurs lagrangiens sont de taille n, alors qu'ils sont seulement de taille m_1 dans la relaxation lagrangienne. Nous présentons dans la section suivante une variante de la décomposition lagrangienne, mettant en œuvre des multiplicateurs de taille inférieure à n.

2.2.1.3 Substitution lagrangienne

La substitution lagrangienne [91] est une méthode de décomposition qui constitue un intermédiaire entre la décomposition lagrangienne et la relaxation lagrangienne. La substitution lagrangienne permet de manipuler des multiplicateurs de Lagrange dont la taille est comprise entre m_1 et n. Son principe consiste à définir des copies explicites et implicites de variables. Leur

relaxation met en jeu deux vecteurs de multiplicateurs dont la somme des dimensions est comprise entre m_1 et n. On envisage une bipartition du vecteur $x = (x', x'')$ telle que $x' \in \mathbb{R}^p$ avec $p < n - m_1$.

On considère le vecteur $y = (y', y'')$ composé d'une copie explicite y' de x' et une copie implicite y'' de x'', on obtient le problème suivant équivalent au problème initial, dans lequel $A = (A', A'')$:

$$(PLNE) \equiv \begin{cases} min\ c\ x \\ Ay \geq a\ , y \in Y \\ x' = y' \\ A''y'' \geq A''x'' \\ Bx \geq b\ , x \in \mathbb{N}^n \\ \mathbb{N}^n \subseteq Y \end{cases} \quad \textbf{(2.20)}$$

En relâchant les contraintes $y' = x'$ (avec un multiplicateur $u' \in \mathbb{R}^p$) et $A''y'' \geq A''x''$ (avec un multiplicateur $u'' \in \mathbb{R}_+^{m_1}$), on obtient le lagrangien

$$L_{x,y}(u', u'') = L_x(u', u'') + L_y(u', u'')$$

Avec
$$L_x(u', u'') \equiv \begin{cases} min\ c\ x - u'x' + u''A''x'' \\ Bx \geq b \\ x \in \mathbb{N}^n \end{cases}$$

$$L_y(u', u'') \equiv \begin{cases} min\ u'y' - u''A''y'' \\ Ay \geq a \\ y \in Y \end{cases}$$

Le dual de la substitution lagrangienne est donné par :

$$(DSL) \equiv \begin{cases} z_{DSL} = \max\limits_{u', u''} L_{x,y}(u', u'') \\ u' \in \mathbb{R}^p, u'' \in \mathbb{R}_+^{m_1} \end{cases}$$

La linéarisation du problème (DSL) est donnée comme suit :

$$(DSL) \equiv \begin{cases} max\ \ w_1 + w_2 \\ w_1 \leq min\ c\ x - u'x' + u''A''x''\ , \quad x \in Z_x \\ w_2 \leq u'y' - u''A''y''\ , \quad y \in Z_y \\ w_1\ , w_2 \in \mathbb{R}\ , (u', u'') \in \mathbb{R}^q \times \mathbb{R}_+^{m_1} \end{cases}$$

où $Z_x = \{x \in \mathbb{N}^n | Bx \geq b\}$ et $Z_y = \{y \in Y\ | Ay \geq a\}$.

Cette méthode a été appliquée pour la résolution du problème de localisation avec capacités dans [92]. Comparée à la décomposition lagrangienne, elle manipule un nombre réduit de variables duales, et permet d'obtenir en pratique les mêmes bornes en un temps réduit.

2.2.1.4 Décomposition de Dantzig et Wolfe

Cette méthode de décomposition a été proposée en 1960 dans [87] et a connu une large utilisation pour la décomposition de plusieurs problèmes complexes issus de la pratique. Elle est devenue un outil standard en programmation linéaire [93], [94]. Nous présentons dans cette section son principe, son application à un cas particulier et quelques méthodes de résolution.

Soit $X = \{x \in \mathbb{N}^n : Bx \geq b\} = \{x_i , i \in I\}$ un ensemble fini et I l'ensemble des indices de ses éléments.

D'après le théorème de Minkowski, tout point x de X peut s'écrire comme suit :

$$x = \sum_{i\in I} \lambda_i x_i \text{ avec } \sum_{i\in I} \lambda_i = 1 \text{ , et } \forall i \in I : \lambda_i \in \{0,1\}$$

Le problème $(PLNE)$ peut être reformulé comme suit :

$$(PLNE_{DW}) \equiv \begin{cases} min \sum_{i\in I} (c_i x_i)\lambda_i \\ \sum_{i\in I} (A^i x_i)\lambda_i \geq a \\ \sum_{i\in I} \lambda_i = 1 \\ \lambda_i \in \{0,1\} , \forall i \in I \end{cases} \quad \textbf{(2.21)}$$

Notons que ce problème est aussi difficile à résoudre que le problème $(PLNE)$, mais ses relaxations continues respectives sont différentes. La relaxation continue de $(PLNE)$ implique la relaxation des contraintes d'intégrité sur la globalité du problème, alors que celle de $(PLNE_{DW})$ implique la relaxation des contraintes d'intégrité sur les contraintes $Ax \geq a$ uniquement (alors qu'elles sont maintenues sur les contraintes $Bx \geq b$). La relaxation de $(PLNE_{DW})$ donne donc une meilleure borne que celle de $(PLNE)$ car l'ensemble des solutions admissibles est plus grand.

La décomposition de Dantzig et Wolfe exploite cette propriété. Elle repose sur la convexification de l'ensemble X et la relaxation des contraintes d'intégrité sur les variables du problème $(PLNE_{DW})$.

Posons $\widehat{c_i} = c_i x_i$, $\widehat{A^i} = A^i x^i$, il en découle la formulation suivante :

$$(PLNE_{DW}) \equiv \begin{cases} min \sum_{i \in I} \widehat{c_i} \lambda_i \\ \sum_{i \in I} \widehat{A^i} \lambda_i \geq a \\ \sum_{i \in I} \lambda_i = 1 \\ \lambda_i \in \{0,1\} , \forall i \in I \end{cases} \quad \textbf{(2.22)}$$

Ce nouveau modèle est équivalent à $(PLNE)$ et possède $m_1 + 1$ contraintes comparativement à $m_1 + m_2$ contraintes ; par contre, il contient un nombre beaucoup plus important de variables, celui-ci étant égal au nombre de points extrêmes de X.

La résolution de la formulation de Dantzig et Wolfe consiste à résoudre la relaxation continue ($\lambda_i \geq 0 , \forall i \in I$) du problème (2.22), appelée problème maître (PM). La taille de ce problème est potentiellement exponentielle (nombre de variables égal au nombre de points extrêmes d'un polyèdre), nous devons donc générer dynamiquement ces variables jusqu'à exhiber la base optimale. Nous présentons dans ce qui suit deux méthodes de résolution dont l'une est vue comme le dual de l'autre : La génération de colonnes et la méthode des plans coupants de Kelley.

Structure bloc diagonale

Dans les programmes linéaires qui modélisent des applications réelles comportant un grand nombre de variables et de contraintes, les matrices de contraintes sont généralement "très creuses" et les éléments nuls sont distribués de telle façon qu'ils forment de grandes sous matrices. Les sous-matrices non nulles sont alors regroupées de telle sorte que des sous-ensembles indépendants

de variables et de contraintes apparaissent, généralement liés par un ensemble de contraintes, alors qualifiées de *liantes* ou *couplantes*.

Supposons que la matrice B du problème $(PLNE)$ soit bloc diagonale ; cela induit que le problème $(PLNE)$ peut s'écrire de la façon suivante à des permutations des lignes et des colonnes près :

$$(PLNE) \equiv \begin{cases} min\ c\ x \\ \begin{pmatrix} A_1 & \cdots & A_K \\ B_1 & 0 & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & B_K \end{pmatrix} x \geq \begin{pmatrix} a \\ b_1 \\ \vdots \\ b_K \end{pmatrix} \\ x \in \mathbb{N}^n \end{cases}$$

Plus précisément, si chaque sous-matrice $B_k, k = 1, \ldots, K$, contient q_k lignes et n_k colonnes (bien sûr, $\sum_{k=1}^{q} q_k = m_2\ et\ \sum_{k=1}^{q} n_k = n$), et en notant x_k (resp. b_k, c_k, A_k) le sous-vecteur des variables x (resp. le vecteur du membre droit, le vecteur de la fonction objectif, les colonnes de la matrice de contraintes), le $(PLNE)$ s'écrit encore :

$$(PLNE) \equiv \begin{cases} min \sum_{k=1}^{K} c_k\ x_k \\ \sum_{k=1}^{K} A_k\ x_k \geq a \quad (m_1\ contraintes) \\ B_k x_k \geq b_k \quad \forall\ k = 1, \ldots., K \\ x_k \in \mathbb{N}^{n_k} \end{cases}$$

Pour écrire le problème global en fonction des points extrémaux des sous-systèmes $X^k = \{y \in \mathbb{N}^{n_k} | B_k y \geq b_k\}$, on introduit encore quelques notations ; on définit pour chaque sous-système $k \in \{1, \ldots, K\}$:

I_k : l'ensemble des indices de ses points extrémaux.

n_k : son nombre de variables ($n_k = |I_k|$).

$x_k^i , i \in I_k$: son point extrémal d'indice i.

$\lambda_k^i , i \in I_k$: la variable d'indice i (coefficient de la combinaison convexe des points extrémaux de X^k associé au point extrême x_k^i).

$\hat{c}_k^i = \sum_{i \in I_k} c_k^i x_k^i$: coût associé à la variable λ_k^i.

$\hat{A}_k^i = \sum_{i \in I_k} A_k^i x_k^i$: colonne associée à la variable λ_k^i.

Le nouveau problème maître (PM) construit par analogie avec la formulation de Dantzig et Wolfe vue précédemment est alors :

$$(PM) \equiv \begin{cases} \min \sum_{k=1}^{K} \sum_{i \in I_k} \hat{c}_k^i \lambda_k^i \\ \sum_{k=1}^{K} \sum_{i \in I_k} \hat{A}_k^i \lambda_k^i \geq a \\ \sum_{i \in I_k} \lambda_k^i = 1 \ , \qquad \forall k = 1, \dots, K \\ \lambda_k^i \in \{0,1\}, \forall i \in I_k, \forall k = 1, \dots, K \end{cases}$$

Le modèle (PM) possède $m_1 + K$ contraintes, comparativement à $m_1 + m_2$ contraintes pour le problème initial. Son nombre de variables, qui correspond au nombre de points extrémaux des sous-systèmes, est bien entendu toujours d'ordre exponentiel. Dans le cadre d'un schéma de génération de colonnes, on alimentera le problème maître (PM) à l'aide de K sous-problèmes, chacun fournissant des points extrémaux du polyèdre X^k associé.

2.2.1.5 Décomposition de Benders

La méthode de décomposition de Benders est assez caractéristique de cet état de fait. Proposée au début des années 1960, cette méthode permet de décomposer un programme mathématique mixte, i.e. présentant des variables entières et continues, afin de traiter de façon séparée la partie entière (constituant le programme maître) de la partie continue (constituant le programme satellite). L'idée de la méthode de Benders est d'obtenir la solution optimale du problème mixte de départ en résolvant alternativement et itérativement les programmes maître et satellite, le maître proposant des valeurs pour les variables entières et le satellite définissant, soit les valeurs des variables continues si c'est possible, soit de nouvelles contraintes restreignant les valeurs des variables entières.

Soit le programme linéaire suivant à n variables et m contraintes :

$$(PB) \equiv \begin{cases} min\ c_1x + c_2y \\ A_1x + A_2y \geq b\ (m\ contraintes) \\ y \in Y \subset \mathbb{N}^{n_2} \\ x \geq 0 \end{cases}$$

ou $c_1 \in \mathbb{R}^{n_1}, c_2 \in \mathbb{R}^{n_2}, A_1 \in \mathbb{R}^{m \times n_1}, A_2 \in \mathbb{R}^{m \times n_2}, b \in \mathbb{R}^m, n = n_1 + n_2\ , n\ , n_1, n_2, m \in \mathbb{N}$.

En fixant le vecteur y dans (PB), on obtient le programme linéaire $(SPB\ (y))$,

$$(SPB(y)) \equiv \begin{cases} min\ c_1x \\ A_1x \geq b - A_2y\ (m\ contraintes) \\ x \geq 0 \end{cases}$$

Soit $(DSPB\ (y))$ son dual :

$$(DSPB(y)) \equiv \begin{cases} max\ u(b - A_2y) \\ uA_1 \leq c_1 \\ u \geq 0 \end{cases} \qquad \text{ou} \quad u \in \mathbb{R}_+^m.$$

Le problème d'optimisation initial admet plusieurs écritures équivalentes, en linéarisant la dernière formulation, on obtient le problème suivant, appelé problème maître de Benders :

$$(PMB) \equiv \begin{cases} min\, z \\ z \geq c_2y + u_i(b - A_2y)\ \ , i \in I \\ y \in Y \subset \mathbb{N}^{n_2} \\ z \in \mathbb{R} \end{cases}$$

Où I l'ensemble des indices de ses points extrêmes de $U = \{u \in \mathbb{R}^+ | uA_1 \leq c_1\}$, u_i désigne pour tout y une solution optimale de $(DSPB(y))$.

Le principe de résolution consiste alors à alimenter le problème (PMB) par des solutions u_i qui sont optimales pour le sous-problème dual $(DSPB(y))$, en fixant le vecteur y à différentes valeurs.

Cet ajout se traduit par des coupes qui rétrécissent le domaine d'optimisation du problème maître. En effet, la difficulté est d'énoncer les contraintes de type $z \geq c_2y + u_i(b - A_2y)$ (2.2) où u_i est défini implicitement comme solution optimale du sous-problème dual. Pour éviter d'avoir à expliciter toutes ces contraintes (ce qui reviendrait à parcourir tout), on part d'un problème maître de Benders non contraint (problème maître restreint de Benders, noté $(PMRB)$ dans lequel on insère des contraintes (ou coupes) au fur et à mesure du déroulement de

l'algorithme. La génération de ces coupes est guidée par la résolution optimale successive d'un problème dual ($DSPB(y)$) pour un y fixé et du problème ($PMRB$) enrichi d'une nouvelle coupe.

La décomposition de Benders a été appliquée avec succès sur plusieurs applications, on cite [95] pour la résolution du problème de programmation des mouvements des moteurs ferroviaires, [96] pour le routage aérien, [97] pour la construction de systèmes de distributions industrielles et [98] pour la résolution de problèmes de tournées de véhicules. Cependant, cette méthode n'est pas adaptée à certaines applications [99], à cet effet, plusieurs méthodes ont été proposées pour son amélioration [99].

2.2.1.6 Hybridation des méthodes de décomposition

Une classe de méthodes de décomposition appelées méthodes de décomposition intégrées ont été présentées dans [100]. Elles sont issues de l'hybridation de méthodes de décomposition interne et externe qui offrent de meilleures bornes en moins de temps que celles obtenues par chacune des méthodes utilisées indépendamment l'une de l'autre. Lorsque les méthodes de décomposition classiques construisent une approximation, soit interne, soit externe, les méthodes de décomposition intégrées offrent les deux approximations à la fois. Nous présentons dans ce qui suit deux techniques qui intègrent ce principe.

Price-and-Cut

Cette méthode consiste à intégrer la méthode des plans coupants dans la décomposition de Dantzig et Wolfe. Lorsqu'elle est intégrée dans un processus de branch-and-bound, elle est appelée le *branch-and-price-and-cut* [101]. Cette technique alterne entre la résolution du sous-problème d'évaluation qui permet de générer des colonnes améliorantes et du sous-problème de séparation qui génère des inégalités valides améliorantes. La résolution de chacun de ces problèmes produit respectivement une description interne et une description externe du domaine réalisable C.

Le principe est d'optimiser l'objectif sur l'intersection des deux descriptions générées dynamiquement. Pour cela, on utilise une formulation de Dantzig et Wolfe dont le système de contraintes contient en plus des inégalités valides de la forme $\sum_{i \in I} (Dx_i)\lambda_i \geq d$. Ces contraintes changent dynamiquement, soit en nombre de colonnes (ajout de colonnes de coût réduit positif), soit en nombre de contraintes (ajout d'inégalités valides). Lors de chaque itération du *price-and-cut*, on résout au choix le sous-problème d'évaluation ou le sous-problème de séparation. La procédure s'arrête lorsqu'aucune colonne de coût réduit positif, ni aucune inégalité valide, ne peut être générée.

Relax-and-Cut

Comme pour la méthode de Dantzig et Wolfe, la méthode des plans coupants peut être intégrée dans la relaxation lagrangienne [102]. Le principe est le même que celui ci-dessus. Lors de chaque itération de cette méthode, un sous-problème d'évaluation ou un sous-problème de séparation est résolu. La mise à jour des multiplicateurs duaux se fait non pas en résolvant un problème maître, mais en utilisant un algorithme de sous-gradient. La résolution du sous-problème d'évaluation implique la génération d'un point extrême du polyèdre défini par $conv(X)$. Cette solution permet la mise à jour de la solution duale, alors que la résolution du problème de séparation fournit une inégalité améliorante qui est rajoutée au système de contraintes de la relaxation lagrangienne du problème traité.

2.2.1.7 Relation entre la relaxation lagrangienne et la décomposition de Dantzig et Wolfe

Il est bien connu que lorsque la relaxation lagrangienne est obtenue en dualisant exactement les mêmes contraintes liantes dans la formulation de Dantzig et Wolfe, la valeur optimale du dual lagrangien et de la formulation de Dantzig et Wolfe (relâchée) sont égales [103].

Les problèmes engendrés par la décomposition, sont souvent de grande taille (nombre exponentiel de variables) pour lesquels les méthodes de résolution directe s'avèrent inefficaces. Ces problèmes sont généralement résolus avec la génération de colonnes. Nous présenterons dans cette section leur principe, les avantages, les limites et quelques techniques améliorantes de cette méthode.

2.2.2 Principe et convergence de la génération de colonnes

La génération de colonnes a été introduite indépendamment par Dantzig et Wolfe [87] d'un côté et Gilmore et Gomory [104] de l'autre. Elle consiste à résoudre alternativement un problème maître et un sous-problème, résultant de la décomposition du problème d'origine.

Le principe est de sélectionner à chaque itération une variable candidate à l'amélioration de la valeur courante du problème maître, en exploitant le cas particulier où les variables ont une structure spécifique qui permet de les définir par un sous-problème d'optimisation (plus courts chemins [105]).

À l'instar des méthodes itératives, la génération de colonnes peut souffrir d'un problème de convergence, particulièrement lorsqu'elle est utilisée pour résoudre des problèmes dégénérés où elle peut passer beaucoup de temps à générer des colonnes qui n'améliorent que faiblement la valeur de l'objectif, ce phénomène est appelé *l'amortissement de la génération de colonnes*. La version duale de la génération de colonnes, connue sous le nom de la méthode des plans coupants de *Kelley* [106], est basée sur le même principe, mais utilise des techniques de résolution différentes. Du point de vue dual, le problème de convergence se traduit par des variations d'une grande amplitude des solutions duales.

Plusieurs méthodes visant à améliorer la convergence de la génération de colonnes ont été proposées dans la littérature. Elles se basent sur des principes variés selon que l'on opère sur le plan primal ou dual, que l'accélération concerne la résolution du problème maître, du sous-

problème ou bien du processus global, ou encore de son hybridation avec d'autres méthodes de résolution. Les méthodes les plus connues et les plus utilisées sont les méthodes de stabilisation [105] qui opèrent sur le processus global en ayant comme objectif la diminution du nombre d'itérations en réduisant les oscillations des valeurs des variables duales.

2.2.2.1 Principe de la génération de colonnes

Considérons le programme linéaire à n variables et m contraintes :

$$(P) \equiv \begin{cases} min \sum_{i \in I} c_i x_i \\ \sum_{i \in I} A^i x_i \geq a \\ x_i \geq 0 \, , \forall i \in I \end{cases}$$

où I est l'ensemble d'indices avec $|I| = n$, $(c_i \, , A^i) \in (\mathbb{R} \times \mathbb{R}^m), \forall i \in I \; et \; a \in \mathbb{R}^m$.

Soient B une matrice de dimensions $(m \times m)$ associée à une base réalisable de (P), $c_{(B)}$ les coûts associés et $x_{(B)}$ une solution de base associée. La solution duale correspondante est donnée par :

$$\pi = c_{(B)} B^{-1}.$$

La résolution du problème (P) par l'algorithme primal du simplexe consiste à améliorer itérativement la solution de base $x_{(B)}$ en sélectionnant, parmi toutes les variables hors base, une variable de coût réduit améliorant, i.e. négatif pour un problème de minimisation.

Le coût réduit d'une variable x_i est donné par : $\bar{c}_i = c_i - \pi A^i$. La colonne qui rentre dans la base peut être obtenue en résolvant le problème d'évaluation :

$$\bar{c}_{i^*} = \min_{i \in I} c_i - \pi A^i \qquad \textbf{(2.23)}$$

Si $\bar{c}_{i^*} < 0$ alors la solution courante peut être améliorée en introduisant x_{i^*} dans la base. Sinon, la base courante B est optimale pour (P). Si on est amené à manipuler un nombre démesuré de variables (plusieurs millions), il est alors très difficile de les énumérer explicitement et par conséquent, la résolution de (2.23) devient ardue, voire impossible. Dans le cas où les colonnes ont une structure particulière, on peut les définir en résolvant un problème d'optimisation [100].

Par exemple, le problème de découpe [104] dans lequel les colonnes sont des vecteurs à composantes entières non nulles, la définition d'une colonne revient à résoudre un problème de sac à dos à variables entières. Les méthodes de décomposition reposent sur le même principe : le problème d'origine est décomposé en un ou plusieurs sous-problèmes plus simples à résoudre, l'ensemble étant généralement coordonné par un programme linéaire appelé le problème maître.

L'algorithme de génération de colonnes est particulièrement adapté à la résolution des problèmes engendrés par la décomposition de Dantzig et Wolfe. Son principe est similaire à celui de l'algorithme du simplexe et consiste à résoudre le problème (P) avec un ensemble réduit de ses variables, puis à l'alimenter itérativement avec de nouvelles colonnes jusqu'à atteindre l'optimalité. Ainsi, on construit à une itération k donnée, un problème (P^k) (appelé problème maître restreint) à partir de (P) en utilisant un sous-ensemble de colonnes $I^k \subseteq I$:

$$(P^k) \equiv \begin{cases} min \sum_{i \in I^k} c_i x_i \\ \sum_{i \in I^k} A^i x_i \geq a \\ x_i \geq 0 \, , \forall i \in I^k \end{cases}$$

Soient B^k une base optimale et réalisable de (P^k), x^k et π^k les solutions primales et duales associées. On cherche donc une colonne de coût réduit négatif (minimal) qui rentre dans la base en résolvant le problème d'évaluation (2.16) appelé sous-problème. Deux cas peuvent se présenter :

(i) Soit $\bar{c}_{i^*} < 0$: la colonne i^* est rajoutée à l'ensemble I^k et le problème (P^k), augmenté d'une colonne, est réoptimisé sur l'ensemble $I^{k+1} = I^k \cup \{i^*\}$.

(ii) Ou bien $\bar{c}_{i^*} \geq 0$: aucune variable ne possède un coût réduit négatif et par conséquent la base B^k est également optimale pour (P).

La génération de colonnes pour l'optimisation discrète

La résolution des problèmes en nombres entiers nécessite souvent l'utilisation de techniques de décompositions [100]. Nous nous intéresserons dans ce qui suit à la méthode de décomposition de

Dantzig et Wolfe [87] dont l'application aboutit, d'une part à un problème maître et d'autre part à un ou plusieurs sous-problèmes [104]. Pour résoudre le programme en nombres entiers ainsi obtenu, la génération de colonnes peut être utilisée directement [107] ou intégrée à un schéma de type branch-and-bound pour donner la méthode de branch-and-price.

Reprenons le problème $(PLNE)$:

$$(PLNE) \equiv \begin{cases} min\ c\ x \\ Ax \geq a \quad (m_1\ contraintes) avec\ A \in \mathbb{R}^{m_1 \times n}\ et\ a \in \mathbb{R}^{m_1} \\ Bx \geq b \quad (m_2\ contraintes)\ avec\ B \in \mathbb{R}^{m_2 \times n}\ et\ b \in \mathbb{R}^{m_2} \\ x \in \mathbb{N}^n \end{cases}$$

et soit J l'ensemble des indices de $X = \{x \in \mathbb{N}^n : Bx \geq b\}$. En notant $x^j, j \in J$, les éléments de X, le problème $(PLNE)$ peut se réécrire comme suit :

$$(PLNE) \equiv \begin{cases} min\ c\ x^j \\ Ax^j \geq a \\ j \in J \end{cases}$$

En associant une variable binaire $\lambda_j, j \in J$, à chaque élément x^j de X, on obtient la formulation du problème maître de Dantzig et Wolfe équivalent au problème $(PLNE)$:

$$(PLNE_{DW}) \equiv \begin{cases} min \sum_{j \in J} (cx^j)\lambda_j \\ \sum_{j \in J} (Ax^j)\lambda_j \geq a \\ \sum_{j \in J} \lambda_j = 1 \\ \lambda_j \in \{0,1\}, \forall j \in J. \end{cases} \quad \textbf{(2.24)}$$

En relâchant les contraintes d'intégrité sur les variables du problème (2.24) et en substituant J par I, l'ensemble des indices des points extrêmes de $conv(X)$, on obtient la relaxation continue du problème maître de Dantzig et Wolfe :

$$(PM) \equiv \begin{cases} min \sum_{i \in I} (cx^i)\lambda_i & (2.25) \\ \sum_{i \in I} (Ax^i)\lambda_i \geq a & (2.26) \\ \sum_{i \in I} \lambda_i = 1 & (2.27) \\ \lambda_i \geq 0, \forall i \in I & (2.28) \end{cases}$$

Le nombre de points extrêmes étant potentiellement exponentiel, on définit le problème maître (PM^k) restreint au sous-ensemble des variables d'indices dans $I^k \subseteq I$ construit itérativement avec l'algorithme de génération de colonnes (voir procédure décrite ci-dessus).

D'après le paragraphe 2.2.2.1, le coût réduit d'une variable λ_i , $i \in I$, est donné par :

$$cx^i - \pi(Ax^i) - \pi_0,$$

et le problème d'évaluation appelé aussi le sous-problème est formulé comme suit :

$$(SP)\ \min_{i \in I}\{(c - \pi A)x^i - \pi_0\} \Leftrightarrow \min_{x \in Conv(X)}\{(c - \pi A)x - \pi_0\}$$

où $\pi \in \mathbb{R}_+^m$ est le vecteur des variables duales correspondant aux contraintes (2.26) (dites couplantes) et $\pi_0 \in \mathbb{R}$ la variable duale correspondant à la contrainte de convexité (2.27).

Remarque 2

(a) Le problème maître peut être initialisé avec un ensemble de colonnes générées par une heuristique ou bien en utilisant des variables artificielles à l'instar de la première phase de l'algorithme du simplexe.

(b) Il n'est pas nécessaire de résoudre les sous-problèmes jusqu'à l'optimalité, une méthode heuristique peut fournir des colonnes de coût réduit négatif en un temps raisonnable [108], la résolution exacte des sous-problèmes n'est nécessaire que pour prouver l'optimalité.

(c) Afin de réduire le nombre d'itérations de la génération de colonnes, il est préférable de générer lors de chaque itération un ensemble de colonnes de coûts réduits négatifs (cela revient à déterminer plusieurs solutions du sous-problème de coûts réduits négatifs en utilisant des heuristiques ou bien en calculant les k meilleures solutions) que de n'en générer qu'une seule.

(d) Les problèmes $(PLNE)$ et (2.17) sont équivalents mais leurs relaxations continues respectives sont différentes. La relaxation continue de (2.17) correspond à la relaxation lagrangienne des contraintes $Ax \geq a$ [94].

(e) D'après la remarque (b) et lorsque le sous-problème possède la propriété d'intégralité (problème du plus court chemin [105] par exemple), la borne obtenue est la même que celle de la relaxation continue. Afin d'obtenir une meilleure borne, il est préférable que le sous-problème ne possède pas la propriété d'intégralité.

La génération de colonnes dans l'espace dual

Considérons le dual du problème (PM) :

$$(DPM) \equiv \begin{cases} max\ \pi a + \pi_0 \\ \pi_0 \leq cx^i - \pi\left(Ax^i\right)\ \forall i \in I \\ (\pi, \pi_0) \in \mathbb{R}_+^m \times \mathbb{R} \end{cases}$$

Posons $\theta = \pi a + \pi_0$, le problème (DPM) peut être réécrit comme suit :

$$(DPM) \equiv \begin{cases} max\ \theta \\ \theta \leq cx^i + \pi\left(a - Ax^i\right)\ \forall i \in I \\ (\pi, \theta) \in \mathbb{R}_+^m \times \mathbb{R} \end{cases} \quad (2.9)$$

il en découle les formulations équivalentes :

$$(DPM) \equiv \begin{cases} max\ \theta \\ \theta \leq \Theta(\pi) \\ (\pi, \theta) \in \mathbb{R}_+^m \times \mathbb{R} \end{cases} \Leftrightarrow \max_{\pi \in \mathbb{R}_+^m} \Theta(\pi)$$

ou $\Theta(\pi) = \min_{i \in I}\{cx^i + \pi\left(a - Ax^i\right)\}$.

Le problème (DPM) contient un nombre potentiellement exponentiel de contraintes égal à la cardinalité de l'ensemble I, il est donc très difficile, voire impossible de le caractériser explicitement. La méthode des plans coupants de Kelley [106] consiste à manipuler un problème dual restreint sur un ensemble de contraintes $I^k \subset I$, lors de chaque itération k :

$$(DPM^k) \equiv \max_{\pi \in \mathbb{R}_+^m} \Theta^{\mathrm{k}}(\pi)$$

ou $\Theta^{\mathrm{k}}(\pi) = \min_{i \in I^k}\{cx^i + \pi\left(a - Ax^i\right)\}$

Du point de vue dual, la résolution du sous-problème permet de calculer la valeur de Θ au point considéré, définissant des facettes dont l'ensemble caractérise la fonction Θ. Le problème restreint

(DPM^k) est composé d'un sous-ensemble de ces facettes qui représente une enveloppe inférieure de la fonction Θ.

Convergence de la génération de colonnes

La génération de colonnes assure une très rapide amélioration de la valeur du problème maître durant les premières itérations. Cependant, elle peut passer beaucoup de temps à générer des colonnes qui n'améliorent que faiblement la valeur de l'objectif.

La figure 2.3 -(a) illustre *l'effet d'amortissement* lors des dernières itérations.

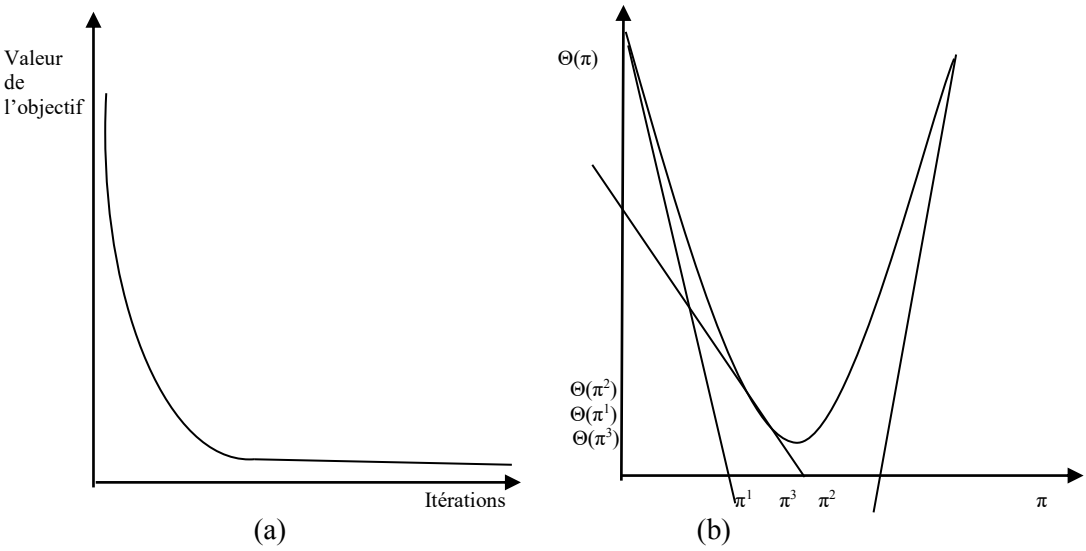

Figure 2.3: Convergence de la génération de colonnes du point de vue primal et dual

Les problèmes de convergence de la génération de colonnes proviennent essentiellement de la dégénérescence primale et de la qualité des colonnes générées qui n'appartiendront pas à la base optimale. Sous l'aspect dual, ces problèmes de convergence se traduisent par des oscillations des valeurs des variables duales où l'on observe des déplacements qui causent une détérioration de la qualité de la solution (Figure 2.3-(b)).

L'étude de la convergence de la génération de colonnes du point de vue dual a été la source du développement des méthodes de stabilisation, dont le principe de base est de générer des solutions proches de la meilleure solution courante, pour diminuer les oscillations des valeurs des variables duales. Une synthèse des différentes méthodes est présentée dans la section suivante.

2.2.2.2 Méthodes de stabilisation

Il existe dans la littérature plusieurs approches dites de stabilisation qui visent à diminuer les fluctuations des variables duales en guidant la méthode pour générer des solutions proches de la meilleure solution trouvée jusqu'à l'itération courante. Ces méthodes opèrent donc sur l'espace dual et ont pour effet de diminuer le nombre d'itérations. Le principe de base de ces méthodes est de restreindre la résolution du problème dual (2.22) au voisinage de la meilleure solution trouvée appelée *le centre de stabilité*. On cite [109] pour la Méthodes de stabilisation par pénalisation du problème dual, [110] et [111] pour les méthodes de type faisceaux, [112], [113], [114] et [115] pour les Méthodes de stabilisation par pénalisations linéaires, [116] pour la Méthode pondérée de Dantzig et Wolfe.

2.2.3 Accélération de la convergence

Nous présentons dans cette section d'autres techniques, qui visent à accélérer la convergence de la génération de colonnes, se basant sur deux principes. Le premier est de tirer profit des avantages, comme la rapidité de résolution ou la qualité des solutions calculées, qu'offrent certaines méthodes d'optimisation. Le second est d'utiliser des techniques de ré-optimisation qui sont adaptées au cas des processus itératifs et qui visent à diminuer le temps de résolution des problèmes à chaque itération.

2.2.3.1 Relaxation lagrangienne au sein d'un schéma de génération de colonnes

Cette méthode [117] consiste à modifier le schéma de la génération de colonnes en effectuant lors de chaque itération une résolution exacte et plusieurs résolutions approchées des sous-problèmes. Dans la résolution exacte, les sous-problèmes sont résolus par rapport à la solution duale obtenue en résolvant le problème maître avec l'algorithme du simplexe (boucle externe). Dans les résolutions approchées, les sous-problèmes sont résolus par rapport aux solutions duales obtenues en résolvant la relaxation lagrangienne du problème de départ (en relâchant les mêmes contraintes) avec l'algorithme de sous-gradient. Les étapes principales sont résumées ci-dessous.

Algorithme 2.1 : Relaxation lagrangienne dans un schéma de génération de colonnes

{***Début***}
Initialiser le problème maître et la précision ε.
{***début de la boucle externe***}
Résoudre le problème maître, soit z_{PM} sa valeur optimale.
Résoudre le sous-problème, soit z_{SP} sa valeur optimale.
if $z_{SP} - z_{PM} < \varepsilon$ **then** Arrêter, la solution obtenue est ε-optimale.
else {***Début de la boucle interne***}
- Générer un ensemble de solutions duales avec l'algorithme de sous-gradient appliqué au dual lagrangien du problème maître (en relâchant les contraintes liantes).
- Résoudre le sous-problème avec ces nouvelles variables duales et générer de nouvelles colonnes.

{***Fin de la boucle interne***}
Rajouter les colonnes générées dans la boucle interne au problème maître si elles n'ont pas encore été générées.
{***Fin de la boucle externe***}
{***Fin***}

Le critère d'arrêt de la boucle externe est celui de la génération de colonnes. Le critère d'arrêt de la boucle interne peut être un nombre d'itérations fixé, lié au nombre de colonnes à rajouter au problème maître lors de chaque itération. Afin de ne pas produire des colonnes déjà générées, une procédure est appliquée à la solution duale afin de générer de nouvelles solutions (pour plus de détails, voir [117]). Cette méthode permet de combiner la rapidité de l'algorithme de sous-gradient et l'exactitude de l'algorithme de Dantzig et Wolfe.

2.2.3.2 Relaxation agrégée au sein d'un schéma de génération de colonnes

Lors de chaque itération de la génération de colonnes, la résolution du sous-problème fournit un minorant sur la valeur optimale du problème maître (cas d'un problème en minimisation), cette technique [118] tente de générer de meilleurs minorants, donc de meilleures colonnes en utilisant la relaxation composite (lagrangienne/agrégée).

2.2.3.3 Ré-optimisation

La ré-optimisation consiste à étudier les possibilités de résolution efficace d'un problème (P') obtenu à partir d'un autre problème (P) en modifiant une partie de ses données. Plusieurs travaux se sont intéressés à la ré-optimisation dans le cadre de diverses méthodes de résolution itératives, nous en présentons quelques-unes ci-dessous.

Ré-optimisation dans la résolution d'une suite de problèmes de plus court chemin

Plusieurs travaux se sont intéressés à la ré-optimisation dans la résolution de plusieurs instances d'un problème de plus court chemin, lorsque certaines données changent. Dans [119] est proposé le premier algorithme de ré-optimisation pour la résolution d'une suite de problèmes de plus court chemin lorsque le coût d'un ou plusieurs arcs diminue.

Le cas où une perturbation du coût est appliquée à un seul arc est étudié dans [120] pour les graphes orientés et dans [121] pour les graphes non orientés. Le cas général où le coût d'un ou plusieurs arcs change est étudié dans [122]. Dans [123], les auteurs se sont intéressés à la ré-optimisation dans la résolution du problème du plus court chemin avec fenêtres de temps ou shortest path problem with time windows (SPPTW) pour le calcul de chemins disjoints de coûts minimaux. Cette technique peut être intégrée dans un schéma de génération de colonnes pour la résolution du problème de VRP où le problème maître est de type couverture de tâches, et le sous-problème de type SPPTW.

Ce problème est présenté plus en détails dans le chapitre 3. Le principe de cette approche est de sauvegarder les étiquettes efficaces obtenues lors du calcul d'un chemin disjoint pour construire une fonction représentant une borne supérieure sur le coût des nouvelles étiquettes efficaces lors du calcul d'un nouveau chemin disjoint.

CHAPITRE 3

3 PROBLEME DE PLUS COURT CHEMIN AVEC CONTRAINTES DE RESSOURCES

Dans la plupart des applications de tournées de véhicules résolues par génération de colonnes, le sous-problème correspond à un problème de plus court chemin avec contraintes de ressources ou shortest path problem with resource constraints (SPPRC) ou à l'une de ses variantes.

Dans ce chapitre, nous proposons un algorithme de résolution approximative du SPPRC pour des graphes arbitraires (acycliques ou cycliques). Notre approche est basée sur la relaxation lagrangienne et est appelée « Dominance on Lagrangian cost for the SPPRC » (DLC-SPPRC). Avant de décrire notre méthode, nous commençons par une classification et une formulation générique des SPPRC, et on aborde brièvement les problèmes de modélisation complexes impliquant des ressources et présente les méthodes de résolution des SPPRC les plus couramment utilisées.

3.1 Introduction

Le problème de plus courts chemins dans un graphe peut se présenter comme suit. Etant donne un graphe orienté et une fonction coût sur les arcs, le problème consiste à trouver un chemin le moins couteux d'un sommet choisi à un autre. Il se résout aisément grâce à de nombreux algorithmes polynomiaux (Bellman [124], Dijkstra [125]). Néanmoins, l'ajout de contraintes sur le chemin le rend plus difficile à résoudre. Ce chapitre présente différents algorithmes exactes ou approchés pour résoudre le SPPRC. On se place dans le cadre général de résolution d'un problème de couverture de taches par des tournées qui doivent respecter certaines contraintes dont le SPPRC est, dans une décomposition classique de type Dantzig - Wolfe, le sous-problème.

Dans de nombreuses applications, les auteurs ont itéré la résolution du SPPRC introduit par Desrochers [126] comme une généralisation multidimensionnelle du problème du plus court

chemin avec des fenêtres de temps. Les méthodes de résolution et les applications du SPPRC ont été largement discutées dans la littérature [127], [105], [126], [128]. Ces stratégies vont des méthodes exactes aux heuristiques et méta-heuristiques.

Les techniques exactes de résolution SPPRC utilisent généralement la programmation dynamique qui a une complexité pseudo-polynomiale. Desrochers et Soumis [123] ont proposé un algorithme à correction d'étiquettes qui étend l'algorithme de Ford-Bellman pour prendre en compte les contraintes de ressources. L'algorithme s'est avéré efficace pour des contraintes de ressources strictes. Feillet [129] a adapté l'algorithme de Desrochers pour résoudre exactement les problèmes de tarification ESPPRC (SPPRC avec contrainte de chemin élémentaire) dans le cadre du problème de tournées de véhicules avec fenêtres de temps (VRPTW). Depuis, cet algorithme a été l'épine dorsale d'un certain nombre d'algorithmes basés sur la génération de colonnes ou les méthodes de branchement et de prix appliqués à plusieurs problèmes importants tels que le routage des véhicules et la gestion des équipages [130], [131]. Le tableau 3-1 résume certaines publications examinées qui ont appliqué la génération de colonnes pour les VRP, classées par ordre chronologique en fonction des approches de solution utilisées pour résoudre le sous-problème SPPRC.

Tableau 3-1: Quelques travaux solvant le VRP par génération de colonnes et résolution du SPPRC.

Auteurs	Année	Classe du VRP
Desrosiers et al. [127]	1995	VRPTW
Kohl et al. [132]	1999	VRPTW
Irnich and Villeneuve [133]	2006	VRPTW
Nagih and Soumis [108]	2006	Autre
Fukasawa et al. [134]	2006	CVRP
Baldacci et al. [135]	2008	CVRP
Baldacci et al. [136]	2010	VRPTW
Baldacci et al. [137]	2011	CVRP
Dabia et al. [138]	2013	Autre

Fukasawa et al. [139]	2015	CVRP
Pecin et al. [140]	2017	CVRP
Himmich et al. [141]	2020	Autre
Sadykov et al. [142]	2021	VRPTW
Behnke et al. [130]	2021	Autre
Mathlouthi et al. [143]	2021	Autre

Le principal inconvénient des techniques de résolution exacte est un temps de calcul élevé qui augmente avec le nombre de ressources, ce qui les rend adaptées uniquement aux problèmes de petite taille. Pour accélérer les calculs et obtenir des solutions en un temps raisonnable, certains auteurs ont proposé des techniques approchées, heuristiques et méta-heuristiques [108], [10], [23], [144]. Ces algorithmes produisent des solutions quasi-optimales et sont plus adaptés aux applications réelles à plus grande échelle (par exemple, des milliers de clients provenant de, des dizaines de dépôts avec de nombreux véhicules et soumis à une variété de contraintes). Nagih et Soumis [108] ont proposé une heuristique de base pour produire rapidement des solutions réalisables. Mais, cette approche a été limitée aux graphes acycliques et sa vitesse n'a pas été prouvée. Certains auteurs ont utilisé un recuit simulé [56], des algorithmes génétiques [145], [146], un algorithme génétique de tri non-dominé [147]. Les méthodes méta-heuristiques permettent d'éviter de se faire piéger dans les optima locaux mais cela se fait au prix d'une convergence plus lente.

Dans ce chapitre, nous proposons un algorithme basé sur la relaxation lagrangienne pour résoudre approximativement le SPPRC pour des graphes arbitraires, acycliques et cycliques. Dans notre approche, appelée « Dominance on Lagrangian cost for the SPPRC » (DLC-SPPRC), une dominance est exprimée sur un sous-ensemble des ressources tandis que le reste des ressources est dualisé dans l'objectif. Des schémas de mise à jour optimisés des paramètres sont utilisés pour

assurer de meilleures performances (étapes de descente, multiplicateurs de Lagrange, sous-gradients).

Le reste du chapitre est organisé comme suit. La section 2 décrit formellement le SPPRC. La section 3 présente notre méthode de résolution approchée qui accélère les calculs tout en gardant une bonne solution approchée. Dans la section 4, nous présentons l'application à la génération de colonnes et montrons les résultats obtenus pour divers ensembles de données VRP.

3.2 Problème de plus court chemin avec contraintes de ressources

3.2.1 Notations et préliminaires

Considérons un graphe $G = (V, A)$ où $V = N \cup \{o, d\}$, est l'ensemble des nœuds $N = \{v_1, \dots, v_n\}$ qui seraient visités d'une origine $o = v_0$ à une destination $d = v_{n+1}$, et un ensemble d'arcs $A \subset V \times V$. On note $\mathcal{R}$ un ensemble de ressources de cardinalité $|\mathcal{R}|$. Un arc $(i, j) \in A$ de coût c_{ij} et consomme une quantité $t_{ij}^r \geq 0$ de chaque ressource $r \in \mathcal{R}$. On suppose que les ressources satisfont l'inégalité triangulaire. Si $(i, j) \in A$, alors le nœud i est appelé prédécesseur de j et j est appelé successeur de i. On appelle chemin une suite finie de nœuds :

$$P = (v_{k_0} = v_0, v_{k_1}, \dots, v_{k_m}), \quad \text{telque} \quad (v_{k_i}, v_{k_{i+1}}) \in A, \forall i = 0, \dots, m-1.$$

A chaque nœud v_j de P, on associe C_j^P, la somme des coûts de ses arcs composites jusqu'à v_j. On note $T_j^{r,P}$ la quantité de ressource $r \in \mathcal{R}$ utilisée pour atteindre le nœud v_j. Par souci de simplification, nous supprimons l'indice P de ces notations et écrivons C_j et T_j^r. Le chemin P est dit réalisable s'il satisfait les contraintes de ressources

$$T_{k_i}^r \in [a_{k_i}^r, b_{k_i}^r], \quad \forall r \in \mathcal{R}, \ \forall i \in \{0, .., m\}.$$

Un chemin P peut aussi être représenté par $X = (x_{ij})_{(i,j)\in A} \in \{0,1\}$, où $x_{ij} = 1$ si les nœuds v_i et v_j appartiennent à P et 0 sinon.

Le SPPRC consiste à trouver un chemin de cout minimal entre la source et la destination, satisfaisant certaines contraintes de ressource, qui seront détaillées par la suite. Si l'ensemble $\mathcal{R}$ des ressources est vide ($\mathcal{R} = 0$), on est ramené au problème usuel de plus court chemin qui est polynomial [124], [125]. En revanche, la considération d'une seule ressource rend déjà le problème d'optimisation NP-dur, et ce même lorsque les coûts et les consommations de ressource sont supposés être entiers positifs [148], [149]. Enfin, le problème consistant à décider seulement s'il existe ou non un chemin réalisable est NP-complet dès lors que l'on considère deux ressources ou plus.

3.2.2 Formulation du problème

Le SPPRC cherche un chemin réalisable de l'origine à la destination avec un coût minimal. De manière similaire à [127], on peut formuler le SPPRC comme suit :

$$minimiser \quad \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{3.1}$$

Vérifiant :

$$\sum_{(i,j)\in A} x_{ij} - \sum_{(i,j)\in A} x_{ji} = 0, \forall\, i \in V \setminus \{o,d\}, \tag{3.2}$$

$$\sum_{(o,j)\in A} x_{oj} = 1; \quad \sum_{(i,d)\in A} x_{id} = 1, \tag{3.3}$$

$$x_{ij}\left(T_i^r + t_{ij}^r - T_j^r\right) \leq 0, \quad \forall (i,j) \in A, \forall\, r \in \mathcal{R}, \tag{3.4}$$

$$a_j^r \leq T_j^r \leq b_j^r, \quad \forall\, j \in V, \forall\, r \in \mathcal{R}, \tag{3.5}$$

$$x_{ij} \in \{0,1\}, \quad \forall (i,j) \in A, \tag{3.6}$$

où $X = (x_{ij})_{i,j}$ représente une solution de chemin. Les équations (3.2)-(3.3) définissent les contraintes de flux sur le graphe. L'équation (3.4) code les exigences de compatibilité entre les variables de débit et de temps, et (3.5) est la contrainte des fenêtres de temps. Si le problème est réalisable, alors il existe une solution optimale [108].

3.2.3 Algorithmes

Pour résoudre le problème SPPRC, des algorithmes d'étiquetage ont été efficacement appliqués par de nombreux auteurs, par exemple, Desrochers [123] ; Dabia [138] et Sun et al. [[150], [151]]. Des étiquettes sont définies pour chaque nœud à coder chemins partiels, ainsi que leur coût et leur consommation de ressources. Chaque nœud reçoit des étiquettes et les étend ensuite vers chaque nœud successeur possible. L'algorithme traite les nœuds de manière itérative jusqu'à ce qu'aucune nouvelle étiquette ne soit créée.

Pour limiter la prolifération d'étiquettes et réduire le temps de calcul, des règles de dominance sont introduites. La relation de dominance est un ordre partiel appliqué pour éliminer les solutions non optimales et ne retenir que les étiquettes non dominées, dites efficientes. Plus le nombre de ressources augmente, plus le nombre d'étiquettes retenues et le temps de calcul augmentent également.

Dans la section suivante, nous proposons un algorithme pour déterminer des solutions d'approximation. La taille de l'espace de recherche est réduite en éliminant certaines étiquettes avec une procédure de dominance heuristique.

3.3 Algorithme de correction d'étiquette Lagrange SPPRC

A chaque chemin P_i arrivant au nœud v_i, on associe un vecteur étiquette (ou état)

$$L_i = \left(C_i, T_i^1; \cdots; T_i^{|\mathcal{R}|}\right). \tag{3.7}$$

Le chemin associé à une étiquette L est noté X^L. Le chemin P est étendu à chaque successeur v_j avec

$$C_j = C_i + c_{ij} \quad \text{and} \quad T_j^r = \max\{a_j^r, T_i^r + t_{ij}^r\}, \quad \forall r \in \mathcal{R}. \tag{3.8}$$

Soient P_i et P_i' deux chemins possibles arrivant à un nœud v_i. On dit que P_i' est dominé par P_i et on écrit $P_i \preccurlyeq P_i'$ si le vecteur étiquette de P_i est inférieur en composantes à celui de P_i'. Par contre, on dit que $P_i \leq P_i'$ si la première composante de P_i est plus petite que celle de P_i' ou en comparant récursivement les suivantes si les premières composantes sont égales. Au moins une des inégalités doit être stricte. Cela définit une relation d'ordre partiel.

3.3.1 Relaxation lagrangienne

Pour accélérer la résolution du problème SPPRC (3.1)-(3.6) , nous proposons de détendre un sous-ensemble de ressources $\mathcal{R}_1 \subset \mathcal{R}$ et d'appliquer la dominance sur les ressources restantes $\mathcal{R}_2 = \mathcal{R} \setminus \mathcal{R}_1$ uniquement. Cela revient à considérer les Équations (3.4)-(3.5) sur $\mathcal{R}_2$ et à remplacer l'Équation (3.1) par

$$minimier \quad \mathcal{L}(\lambda, X), \tag{3.9}$$

où la fonction lagrangienne $\mathcal{L}$ est définie par

$$\mathcal{L}(\lambda, X) = \sum_{(i,j)\in A} x_{ij} \left(c_{ij} + \sum_{r\in\mathcal{R}_1} \lambda_{i,j}^r \left(\max(a_j, T_i^r + t_{ij}^r) - b_j^r\right)\right). \tag{3.10}$$

Comme le flux est supporté par au plus un arc entrant au nœud j depuis ses prédécesseurs i, les multiplicateurs de Lagrange $\lambda_{i,j}^r = \lambda_j^r$ sont indépendants de i. Pour chaque nœud $j \in V \cup \{d\}$, le vecteur multiplicateurs $\lambda_j = (\lambda_j^r)_{r\in\mathcal{R}_1}$peut être obtenu en résolvant le problème de maximisation suivant

$$\begin{cases} \max\limits_{\lambda} & \min\limits_{X} \mathcal{L}(\lambda, X), \\ s.t. & \lambda_j^r \geq 0, \qquad \forall r \in \mathcal{R}_1 \end{cases}. \tag{3.11}$$

Pour résoudre ce sous-problème d'optimisation, on peut utiliser un algorithme de sous-gradient projeté qui s'est avéré efficace en pratique. Le schéma de mise à jour est

$$\lambda_j^{k+1} = \max\{\lambda_j^k + \tau_j^k g_j^k, 0\}, \tag{3.12}$$

où $g_j^k \in \partial\mathcal{L}(\lambda, X)$ est un vecteur de sous-gradient, $\tau_j^k > 0$ est une taille de pas, et l'opérateur max est pris pour chaque élément. Nous choisissons

$$g_j^k = \max(a_j, T_i^r + t_{ij}^r) - b_j^r, \tag{3.13}$$

$$\tau_j^k = \frac{\theta_k(Z_{\text{UB}} - \mathcal{L}(\lambda^k, X))}{||g_j^k||^2}, \tag{3.14}$$

où Z_{UB} est la borne la plus petite (supérieure) la plus connue de la solution du problème (3.1)-(3.6) et $\theta_k \in (0,2]$ est un scalaire.

Une itération pour l'estimation de λ s'est avérée suffisante (précision et rapidité) lorsqu'il est intégré à la génération de colonne.

3.3.2 Dominance

La dominance en chaque nœud j correspond à la détermination des optima de Pareto du problème multicritère appliqué à un ensemble d'étiquettes. Avec l'approche lagrangienne adoptée, le vecteur label devient

$$L_j = \left(C_j + \sum_{r_1 \in \mathcal{R}_1} \lambda_j^{r_1} (T_j^{r_1} - b_j^{r_1});\ T_j^{r_2}, r_2 \in \mathcal{R}_2; \right). \tag{3.15}$$

Lorsque la solution finale obtenue n'est pas réalisable (en raison de la relaxation), nous devons résoudre à nouveau le SPPRC relaxé en dominant sur le coût lagrangien déterminé dans la première phase et sur les ressources $\mathcal{R}_2$ et en vérifiant les fenêtres de ressources dans l'espace d'origine $\mathcal{R}$.

En supposant que tous les prédécesseurs du nœud $j \in N$ ont été considérés, la dominance au nœud j peut être interprétée comme la détermination des optima de Pareto pour le problème multicritère des fonctions $|\mathcal{R}_2| + 1$:

$$\min_{(i;(i,j)\in A)} \left(C_i + c_{ij} \ + \sum_{r_1 \in \mathcal{R}_1} \lambda_j^{r_1} (T_j^{r_1} - b_j^{r_1});\ \max\left\{a_j^{r_2}, (T_i^{r_2} + t_{ij}^{r_2})\right\}, r_2 \in \mathcal{R}_2; \right) \tag{3.16}$$
$$\text{st. } T_i^r + t_{ij}^r \le b_j^r, r \in \mathcal{R}.$$

3.3.3 Procédure de solution approximative

Dans un premier temps, nous introduisons les notations suivantes.

• E_i : Liste des étiquettes au nœud v_i.

• F_i : La liste des chemins associés aux labels de E_i.

• S_i : Ensemble des successeurs des indices du nœud v_i.

• Q : Liste des index des nœuds non traités.

• $Pareto(L, E_j)$: Ajouter une nouvelle étiquette L à un ensemble d'étiquettes non dominées E_j tout en gardant la propriété de non-dominance.

• Z_L est le coût associé à une étiquette L (première composante de l'Équation (3.15)).

Notre procédure améliorée de correction d'étiquettes est détaillée dans l'algorithme 3.1 et l'algorithme 3.2.

Algorithme 3.1 : Algorithme de correction d'étiquettes modifié

Entrée : Un graphe orienté arbitraire $G = (V, A)$, non nécessairement acyclique ; $\mathcal{R}$ est un ensemble de contraintes ; $\mathcal{R}_1 \subset \mathcal{R}$ sous-ensemble de contraintes à inclure dans la fonction objective ; Vecteur de multiplicateurs de Lagrange $\lambda_i \in R_+^{|\mathbb{R}_1|}$ où $v_i \in V - \{o\}$; $\mathcal{R}_2$ est un ensemble de ressources.
Sortie : Tous les chemins non-dominants de la source à la destination E_{n+1} et la liste associée des chemins F_{n+1}.

// Initialisation ;
$\mathcal{R}_2 = \mathcal{R} - \mathcal{R}_1$;
for $v_i \in V$ **do**
  $E_i \leftarrow \emptyset$
end
$E_0 \leftarrow \{(0,0,\cdots,0)\}$; // La taille des libellés est : $|E_o| = |\mathcal{R}_2| + 1$
$Q \leftarrow \{0\}$
while $Q \neq \emptyset$ **do**
  // Choisir un nœud à traiter
  Choisir $i \in Q$;
  for $j \in S_i$ **do**
    for $L_i \in E_i$ **do**
      if $\forall r_2 \in \mathcal{R}_2, T_{ij}^{r2} \leq b_j^{r2}$ **then**
        // Créer de nouveaux libellés L_j pour j en étendant ceux de i;

$$L_j = \left(C_j + \sum_{r_1 \in \mathcal{R}_1} \lambda_j^{r_1} \left(T_j^{r_1} - b_j^{r_1} \right) ;\ T_j^{r_2}, r_2 \in \mathcal{R}_2 \right).$$

        // Appliquer la domination ;
        $E_j \leftarrow Pareto(L_j, E_j)$
        // Mettre à jour F_j;
        Mettre à jour F_j en utilisant E_j.
      end
    end
    if L_j a changé **then**
      $Q \leftarrow Q \cup \{j\}$
    end
  end
  $Q \leftarrow Q \cup \{j\}$
end

Algorithme 3.2 : DLC-SPPRC: Dominance on Lagrangian cost for the SPPRC

Entrée : Un graphe orienté arbitraire $G = (V, A)$, non nécessairement acyclique ; $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$ est un ensemble de contraintes de ressources composé de deux types.
Sortie : $\bar{E}$, Tous les chemins réalisables non-dominés du nœud source au nœud de destination d.
// Choix des paramètres ;
Choisir $\theta \in (0,\ 2], k_{max} = 10$.
// Initialisation ;
for $v_i \in V - \{o\}$ **do**
$\lambda_i^{(0)} = (0, \ldots, 0)$ est un vecteur de taille $|\mathcal{R}_1|$.
// Valeur initiale du multiplicateur de Lagrange correspondant à la solution de (3.11)
$\bar{Z}_{max} \leftarrow -\infty$.
// Choisir une solution initiale pour le problème (3.1)-(3.6).
$X^0 = \left(x_{ij}^0\right)_{ij}$.
// Calculer la valeur de la fonction objectif (3.1) (borne supérieure).
$Z_u \leftarrow \sum_{(i,j)\in A} c_{ij} x_{ij}^0$.
$\bar{E} \leftarrow \emptyset$.
$k = 0$.
// Étape 1;

while $(k < k_{max})$ **do**
    // Appliquer la relaxation de dominance Lagrangienne ;
    $[E_{n+1}, F_{n+1}]$ = **Algorithm 3.1** $(G, \left(\lambda_i^{(k)}\right)_i, \mathcal{R}, \mathcal{R}_1, \mathcal{R}_2)$.
    $(L, X^L) \leftarrow$ est la plus petite étiquette de E_{n+1} et son chemin associé F_{n+1}.
    // Mettre à jour $\bar{Z}_{max}$
    if $(\bar{Z}_{max} < Z_L)$ **then**
        $\bar{Z}_{max} \leftarrow Z_L$
    else
        $\theta \leftarrow \theta/2$.
    if X^L est réalisable **then**
        if $Z_u > Z_L$ **then**
            $Z_u \leftarrow Z_L$
            $\bar{E} \leftarrow$ Toutes les solutions réalisables de E_{n+1}.
    // Mettre à jour les multiplicateurs de Lagrange
    for $v_i \leftarrow V - \{o\}$ **do**
        $\lambda_i^{(k+1)} \leftarrow$ Appliquer les équations (3.13), (3.14) and (3.12).
    if $\left\|\lambda_i^{(k+1)} - \lambda_i^{(k)}\right\| < \varepsilon \ \forall i$ **then**
        break
    $k \leftarrow k + 1$.
// Étape 2
if $\bar{E} = \emptyset$ **then**
    $[\bar{E},]$ = **Algorithm 3.1** $(G, \left(\lambda_i^{(k)}\right)_i, \mathcal{R}, \mathcal{R}_1, \mathcal{R})$.

3.4 Expériences

Pour résoudre le VRPTW, nous utilisons la décomposition de Dantzig-Wolfe qui définit K sous-problèmes indépendants et un problème maître global. En appliquant une génération de colonnes, nous résolvons alternativement le problème maître et les K sous-problèmes. Nous avons abordé les instances VRPTW avec une approche de génération de colonnes utilisant le SPPRC comme sous-problème.

Nous avons utilisé les ensembles de données Solomon [152] et les instances de Homberger avec 200 clients[3]. Chaque instance contient les emplacements des clients, les ressources (deux pour chaque client, c'est-à-dire $|\mathcal{R}| = 2$) et les contraintes. Ces ensembles de données sont classés en trois catégories : les r-instances (les clients sont localisés de manière aléatoire), les c-instances (les clients sont localisés en clusters) et les rc-instances (structures mixtes aléatoires et clusterisées).

De plus, chaque famille d'instances est divisée en deux types. Le premier type (r1xx, c1xx, rc1xx) a un horizon de programmation court, c'est-à-dire de petites fenêtres de temps, qui n'autorise que quelques clients par route (environ 5 à 10). Le deuxième type (r2xx, c2xx, rc2xx) a un horizon de programmation long permettant à de nombreux clients (plus de 30) d'être desservis par le même véhicule. Nous avons appliqué notre algorithme DLC-SPPRC approché et le SPPRC classique [123].

Nous avons implémenté notre algorithme en utilisant le langage de programmation Java. Pour la simulation, nous avons utilisé un CPU Intel Core i9-9900KF (8 cœurs), 3,60 GHz, RAM 32 Go, fonctionnant sous Windows 10 (64 bits). Pour le SPPRC, nous avons implémenté l'algorithme de Desrochers et Soumis [123]. Les programmes linéaires pour les problèmes maîtres restreints sont résolus avec ILOG CPLEX 20.1.

[3] https://www.sintef.no/projectweb/top/vrptw/200-customers/

Les tableaux Tableau 3-2, Tableau 3-3, Tableau 3-4 et Tableau 3-5 rapportent le nombre d'itérations (N_i), la borne inférieure (L_i^b), le temps de calcul en secondes (T_i) et le nombre de colonnes générées (C_i), où $i = 1$ pour SPPRC et $i = 2$ pour DLC-SPPRC. Un écart est calculé comme

$$Gap = 100 \times \frac{L_1^b - L_2^b}{L_1^b}.$$

La figure 3.1 montre le nombre total de colonnes générées.

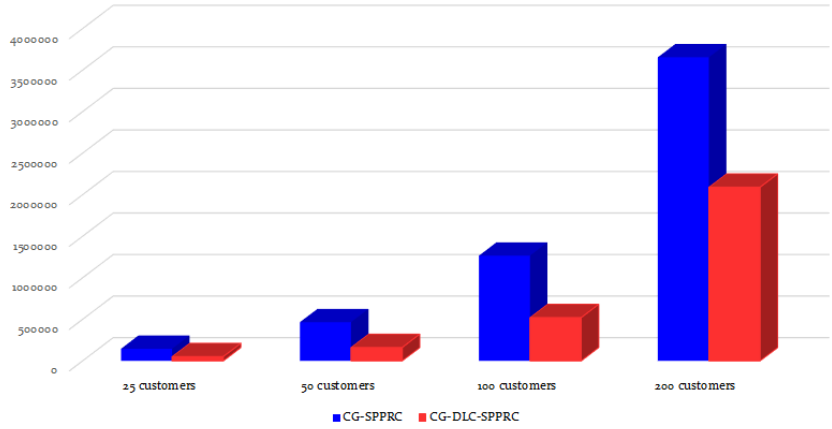

Figure 3.1: Nombre des colonnes générées pour les instances de Solomon et Homberger.

Dans la plupart des cas, nous avons obtenu les solutions optimales dans des délais raisonnables. Comme dans [129], le processus de génération de colonne est initié avec une adaptation de l'algorithme de Clarke et Wright [28]. Le sous-problème est stoppé à chaque itération lorsque 500 étiquettes ont été étendues au dépôt avec un coût négatif.

Tableau 3-2: Résolution du VRPTW (Instances de Solomon avec 25 clients) par DLC-SPPRC (3.2) et SPPRC Classique [123].

Instance	SPPRC				DLC-SPPRC				Comparison	
	L_1^b	T_1	N_1	C_1	L_2^b	T_2	N_2	C_2	Gap	T_1/T_2
c101	191.3	0.1163	33	579	191.3	0.1045	69	396	0.00	1.11
c102	189.2	0.285	40	1452	189.18	0.2817	82	976	0.01	1.01
c103	187.86	0.4005	43	1578	183.21	0.3721	66	807	2.47	1.08
c104	184.34	0.4012	45	1896	183.46	0.2618	44	787	0.48	1.53
c105	191.3	0.0651	36	718	191.3	0.0319	48	368	0.00	2.04
c106	191.3	0.0438	38	698	191.3	0.0372	73	386	0.00	1.18
c107	191.3	0.0476	27	749	191.3	0.0397	48	416	0.00	1.20
c108	187.84	0.1046	30	1090	185.81	0.0734	41	564	1.08	1.43
c109	181.93	0.1442	29	1132	185.31	0.0752	32	578	-1.86	1.92
c201	214.7	0.2174	89	2240	214.7	0.1531	237	1019	0.00	1.42
c202	214.7	0.3528	86	2637	214.7	0.2835	175	1266	0.00	1.24
c203	213.78	2.0617	76	3820	213.78	1.0133	153	1562	0.00	2.03
c204	207.17	4.3672	80	4880	207.17	2.0133	125	2083	0.00	2.17
c205	196.53	0.2943	43	3064	196.53	0.1322	88	1089	0.00	2.23
c206	194.02	0.5541	48	3472	194.02	0.2823	85	1875	0.00	1.96
c207	200.44	1.6901	67	4840	200.44	0.5865	84	2237	0.00	2.88
c208	183.86	1.0371	50	4124	183.86	0.3695	69	1751	0.00	2.81
r101	617.1	0.0065	10	108	617.1	0.0053	17	73	0.00	1.23
r102	546.33	0.0172	13	314	546.33	0.017	29	201	0.00	1.01
r103	454.07	0.0464	22	538	454.07	0.0342	35	280	0.00	1.36
r104	414.85	0.0715	25	735	414.85	0.0544	39	374	0.00	1.31
r105	530.5	0.014	19	254	530.5	0.011	25	145	0.00	1.27
r106	457.3	0.0439	22	543	457.3	0.023	27	265	0.00	1.91
r107	415.13	0.0573	25	618	415.13	0.0446	35	329	0.00	1.28
r108	389.42	0.0846	22	796	389.42	0.0623	32	422	0.00	1.36
r109	439.43	0.0247	19	341	439.43	0.0215	30	243	0.00	1.15
r110	419.07	0.0402	19	502	419.07	0.034	31	331	0.00	1.18
r111	412.82	0.044	19	554	412.82	0.036	31	302	0.00	1.22
r112	365.03	0.0994	27	818	365.03	0.07	35	464	0.00	1.42
r201	448.5	0.0706	16	877	448.5	0.0468	43	510	0.00	1.51
r202	374.09	0.1505	28	1479	374.09	0 122	53	754	0.00	1.23
r203	337.52	0.4222	37	2044	337.51	0.2901	62	897	0.00	1.46
r204	303.99	1.737	51	3900	303.99	0.615	74	1454	0.00	2.82
r205	365.48	0.2141	31	1744	365.48	0.1885	74	1043	0.00	1.14
r206	317.98	1.2275	42	3775	317.98	0.3397	59	1119	0.00	3.61
r207	309.62	3.0179	53	3413	309.61	0.7511	67	1377	0.00	4.02
r208	291.20	51.794	88	9718	291.22	3.4682	100	2952	0.00	14.9
r209	327.26	0.6002	34	2590	327.26	0.2808	53	1133	0.00	2.14
r210	340.51	0.3063	31	2033	340.51	0.182	49	875	0.00	1.68
r211	299.46	7.8134	57	5667	299.46	1.5388	71	1590	0.00	5.08
rc101	390.15	0.0187	18	284	390.15	0.0178	31	203	0.00	1.05
rc102	347.08	0.0591	20	467	347.08	0.0489	32	284	0.00	1.21
rc103	313.98	0.1227	21	751	309.91	0.08	25	330	1.30	1.53
rc104	287.54	0.255	29	888	292.46	0.1133	22	326	-1.71	2.25
rc105	408.53	0.0433	17	516	408.53	0.0299	32	202	0.00	1.45
rc106	314.27	0.0795	23	660	306.04	0.0618	33	343	2.62	1.29
rc107	281.29	0.258	31	1043	274.39	0.1383	33	460	2.45	1.87
rc108	270.57	0.4552	35	939	250.41	0.2489	40	596	7.45	1.83
rc201	315.85	0.1049	30	1325	315.85	0.0673	59	464	0.00	1.56
rc202	233.59	7.4873	57	3545	233.59	1.404	68	1043	0.00	5.33
rc203	178.26	31.379	93	8566	178.28	2.9895	77	1756	-0.01	10.5
rc204	153.91	90.171	141	12217	153.55	4.7891	107	2803	0.23	18.8
rc205	258.92	0.3965	30	1960	258.92	0.1849	50	562	0.00	2.14
rc206	188.29	4.0061	71	6013	188.25	0.7239	70	1096	0.02	5.53
rc207	169.16	11.708	79	7492	169.19	0.9203	68	1812	-0.02	12.7
rc208	138.29	145.45	163	16490	139.54	4.3513	121	3469	-0.91	33.4

Tableau 3-3: Résolution du VRPTW (Instances de Solomon avec 50 clients) par DLC-SPPRC (3.2) et SPPRC Classique [123].

Instance	SPPRC				DLC-SPPRC				Comparison	
	L_1^b	T_1	N_1	C_1	L_2^b	T_2	N_2	C_2	Gap	T_1/T_2
c101	362.40	0.4482	60	1527	362.40	0.4382	155	1052	0.00	1.02
c102	360.25	1.2188	83	2922	360.28	0.9699	136	1792	-0.01	1.26
c103	360.25	2.5992	106	3599	353.34	1.7055	182	2264	1.92	1.52
c104	352.27	6.1074	119	5660	346.95	2.8453	127	2480	1.51	2.15
c105	362.4	0.5268	81	2172	361.2	0.3185	166	1259	0.33	1.65
c106	362.4	0.3838	88	2169	362.4	0.2125	125	973	0.00	1.81
c107	362.4	0.4258	73	2435	361.2	0.3272	150	1433	0.33	1.30
c108	359.81	0.6627	62	2595	356.31	0.5390	106	1469	0.97	1.23
c109	354.32	0.9185	57	2784	340.94	0.6901	94	1528	3.78	1.33
c201	360.2	6.6625	213	13828	360.2	1.6169	489	3335	0.00	4.12
c202	360.2	8.1671	200	12216	360.2	2.5508	259	3440	0.00	3.20
c203	359.8	38.697	252	15108	359.8	14.195	333	5850	0.00	2.73
c204	347.40	114.69	243	18375	347.68	35.732	320	8201	-0.08	3.21
c205	341.76	9.5662	145	13681	341.78	3.5832	223	4713	0.00	2.67
c206	338.52	19.416	180	15798	338.38	4.8996	222	5187	0.04	3.96
c207	348.63	38.049	203	18358	348.61	7.9878	235	7176	0.01	4.76
c208	331.28	20.672	180	15147	331.28	3.7519	174	4892	0.00	5.51
r101	1043.4	0.0447	23	463	1043.4	0.0421	44	276	0.00	1.06
r102	909	0.1435	29	987	909	0.1272	54	534	0.00	1.13
r103	756.12	0.3896	43	1468	756.12	0.2930	59	790	0.00	1.33
r104	608.52	1.2382	66	2600	608.49	0.8123	87	1255	0.00	1.52
r105	890.19	0.0958	26	821	890.19	0.0834	43	395	0.00	1.15
r106	789.43	0.3278	38	1255	789.43	0.2506	59	690	0.00	1.31
r107	697.77	0.5040	46	1793	697.77	0.3672	60	855	0.00	1.37
r108	578.48	1.6057	68	2761	578.48	1.2553	106	1613	0.00	1.28
r109	727.52	0.2537	36	1403	727.52	0.1712	53	614	0.00	1.48
r110	675.46	0.4694	42	1578	675.25	0.3234	59	761	0.03	1.45
r111	658.75	0.5920	47	1927	658.75	0.4077	65	913	0.00	1.45
r112	582.72	1.3422	63	2502	582.61	0.7894	82	1170	0.02	1.70
r201	754.10	0.5013	42	2332	754.10	0.4006	91	1243	0.00	1.25
r202	637.56	1.9978	62	4231	637.56	1.1956	106	1717	0.00	1.67
r203	538.53	5.4656	78	4791	538.53	2.7795	140	2696	0.00	1.97
r204	441.01	329.02	229	23629	441.02	28.779	272	7887	0.00	11.4
r205	595.54	2.0053	66	4505	595.55	1.0479	115	1970	0.00	1.91
r206	530.53	8.3613	87	7378	530.51	3.2432	142	3182	0.00	2.58
r207	467.28	34.011	145	11323	467.29	7.7842	177	4139	0.00	4.37
r208	423.78	2234.5	302	34624	423.80	120.14	309	9984	0.00	18.6
r209	535.28	8.0839	85	6534	535.28	2.9226	139	2811	0.00	2.77
r210	532.1	5.5001	86	6296	532.10	2.3795	135	2803	0.00	2.31
r211	457.43	31.127	129	10024	457.43	6.4031	172	4082	0.00	4.86
rc101	826.61	0.1158	31	716	826.61	0.0998	47	428	0.00	1.16
rc102	706.61	0.4957	47	1160	706.56	0.3879	62	652	0.01	1.28
rc103	612.24	1.2763	58	1756	605.15	0.9114	77	1008	1.16	1.40
rc104	524.12	4.7657	88	3146	511.58	1.7867	86	1417	2.39	2.67
rc105	746.31	0.2826	38	1091	745.93	0.2183	52	625	0.05	1.29
rc106	633.23	0.5801	43	1407	628.52	0.4113	53	741	0.74	1.41
rc107	570.67	2.1353	73	2734	561.86	1.1098	87	1273	1.54	1.92
rc108	525.12	3.43	79	2630	501.70	1.9840	96	1477	4.46	1.73
rc201	530.51	1.1744	64	3001	530.51	0.5838	79	1015	0.00	2.01
rc202	416.52	25.883	132	8997	416.52	5.6157	147	2801	0.00	4.61
rc203	326.29	216.70	264	20712	326.29	15.148	184	5153	0.00	14.3
rc204	263.83	18965.	431	47037	264.30	70.370	333	12067	-0.18	270.
rc205	481.61	6.5427	96	7435	481.62	2.5118	126	2633	0.00	2.60
rc206	363.89	56.114	189	15075	363.89	7.3344	173	3830	0.00	7.65
rc207	333.07	282.08	239	23198	333.07	18.286	196	5016	0.00	15.4
rc208	265.11	1974.0	456	41459	264.91	86.384	399	10952	0.07	22.9

Tableau 3-4: Résolution du VRPTW (Instances de Solomon avec 100 clients) par DLC-SPPRC (3.2) et SPPRC Classique [123].

Instance	SPPRC				DLC-SPPRC				Comparison	
	L_1^b	T_1	N_1	C_1	L_2^b	T_2	N_2	C_2	Gap	T_1/T_2
c101	827.3	2.2196	122	3553	827.3	2.4001	383	2852	0.00	0.92
c102	827.3	9.7928	191	7359	820.3	7.3079	277	4279	0.85	1.34
c103	826.3	21.813	223	9480	817.88	17.609	344	5663	1.02	1.24
c104	821.60	55.514	327	13285	793.65	47.015	544	8789	3.40	1.18
c105	827.3	2.9878	142	4392	821.2	2.9093	308	3648	0.74	1.03
c106	827.3	4.0281	134	4928	827.4	3.5591	258	3438	-0.01	1.13
c107	827.3	3.0508	132	4626	819.6	4.128	331	4435	0.93	0.74
c108	817.35	6.5019	160	6311	801.70	5.2622	219	3318	1.92	1.24
c109	809.27	12.671	231	9430	778.53	7.4906	256	4669	3.80	1.69
c201	589.1	44.949	418	28845	589.1	11.667	528	6526	0.00	3.85
c202	589.1	179.66	552	44694	589.1	61.467	722	12701	0.00	2.92
c203	585.77	1223.0	995	75865	585.77	446.23	1488	32852	0.00	2.74
c204	582.38	4893.7	1630	136556	582.23	990.20	2056	58070	0.03	4.94
c205	582.37	151.92	472	42390	582.36	44.655	617	15218	0.00	3.40
c206	575.99	346.55	614	56564	575.85	61.825	652	16738	0.03	5.61
c207	570.52	367.88	544	52839	570.52	74.188	691	18018	0.00	4.96
c208	570.26	423.42	635	59277	570.28	65.680	614	17867	0.00	6.45
r101	1631.2	0.416	44	1458	1631.2	0.3951	92	829	0.00	1.05
r102	1466.6	1.5553	63	2627	1466.6	1.5157	126	1469	0.00	1.03
r103	1203.2	5.3805	104	4549	1203.2	3.7518	154	2123	0.00	1.43
r104	937.06	28.482	190	9368	936.74	22.017	310	4715	0.03	1.29
r105	1341.2	1.2316	64	2714	1341.2	0.8896	92	1270	0.00	1.38
r106	1212.4	6.0697	116	4719	1212.3	3.0527	143	2155	0.00	1.99
r107	1037.0	13.541	148	6645	1037.3	7.2842	192	3094	-0.03	1.86
r108	891.65	44.401	249	12269	890.07	24.632	336	5525	0.18	1.80
r109	1097.5	4.8182	101	4941	1097.3	2.2239	114	1915	0.01	2.17
r110	1021.3	11.503	135	6226	1021.1	5.6338	171	2783	0.02	2.04
r111	1006.0	12.760	153	7009	1005.7	7.4647	200	3273	0.04	1.71
r112	892.58	35.353	201	9372	887.71	18.072	283	4819	0.54	1.96
r201	1080.8	9.8496	119	9102	1080.8	6.4248	231	3947	0.00	1.53
r202	933.45	93.705	241	15273	933.46	23.759	353	6624	0.00	3.94
r203	756.74	451.44	402	28667	756.73	87.193	565	11958	0.00	5.18
r204	640.24	7638.3	638	56850	640.27	559.26	863	22660	-0.01	13.7
r205	838.77	76.568	242	19494	838.77	29.639	372	8314	0.00	2.58
r206	749.07	474.10	370	28002	749.07	84.832	538	12161	0.00	5.59
r207	668.71	3609.2	568	50978	668.71	281.51	703	19200	0.00	12.8
r208	-	-	-	-	610.28	1197.7	1001	31544	-	0.00
r209	750.46	445.08	292	28162	750.45	82.065	484	11179	0.00	5.42
r210	753.99	205.37	309	23599	753.99	63.313	494	11025	0.00	3.24
r211	650.83	1793.7	573	47048	650.85	167.98	741	17532	0.00	10.7
rc101	1567.5	1.0734	61	2250	1567.3	0.8967	92	1159	0.01	1.20
rc102	1380.2	2.9682	84	3319	1378.3	2.6881	124	1776	0.14	1.10
rc103	1170.3	12.267	155	5901	1164.4	7.3091	177	2656	0.51	1.68
rc104	1052.6	38.189	190	8215	1027.6	19.505	250	4057	2.38	1.96
rc105	1453.9	2.213	72	2938	1453.2	1.6459	105	1565	0.05	1.34
rc106	1249.0	4.2376	97	3982	1242.8	2.694	124	1847	0.49	1.57
rc107	1117.4	12.967	146	5703	1096.2	7.1789	179	2847	1.90	1.81
rc108	1035.9	32.309	198	7553	1019.8	13.186	225	3831	1.56	2.45
rc201	1107.0	13.229	149	9682	1107.0	6.554	227	3656	0.00	2.02
rc202	880.34	127.71	265	17952	880.33	29.594	339	6574	0.00	4.32
rc203	693.53	902.87	475	35439	693.1	146.63	591	13829	0.06	6.16
rc204	607.66	14066.	815	78145	606.76	587.30	854	25540	0.15	24.0
rc205	967.11	59.449	230	15595	967.10	19.762	304	5936	0.00	3.01
rc206	852.17	130.59	305	24025	852.18	30.133	348	7607	0.00	4.33
rc207	767.95	567.55	417	29098	767.98	80.877	456	10815	0.00	7.02
rc208	627.28	3223.9	638	44904	627.16	223.57	726	17074	0.02	14.4

Tableau 3-5: Résolution du VRPTW (Instances de Homberger avec 200 clients) par DLC-SPPRC (3.2) et SPPRC Classique [123].

Instance	SPPRC				DLC-SPPRC				Comparison	
	L_1^b	T_1	N_1	C_1	L_2^b	T_2	N_2	C_2	Gap	T_1/T_2
c1_2_1	2698.6	21.699	258	9936	2694.3	18.046	422	6025	0.2	1.20
c1_2_2	2681.2	268.07	636	19085	2674.9	136.10	732	10144	0.2	1.97
c1_2_3	2619.2	1091.6	855	25103	2718.2	365.00	940	14092	-3.8	2.99
c1_2_4	2504.6	2681.4	1096	33006	3018.1	297.10	1045	15950	-21.	9.03
c1_2_5	2694.9	40.977	347	13060	2693.4	30.235	490	8212	0.1	1.36
c1_2_6	2694.9	56.494	354	13018	2691.1	39.252	481	7749	0.1	1.44
c1_2_7	2694.9	57.383	404	14065	2690.6	40.905	513	9006	0.2	1.40
c1_2_8	2595.3	157.61	506	17311	2571.0	65.430	458	8383	0.9	2.41
c1_2_9	2512.5	336.61	662	21390	2468.5	134.59	667	11797	1.8	2.50
c1_2_10	2456.9	849.79	743	23092	2418.0	273.42	829	14281	1.6	3.11
c2_2_1	1915.0	586.61	1042	70794	1915.0	142.94	1122	17930	0.0	4.10
c2_2_2	1830.7	11210.	2147	133228	1830.3	3136.7	2505	41482	0.0	3.57
c2_2_3	1704.0	32211.	2903	168434	1704.6	7460.9	3318	55403	0.0	4.32
c2_2_4	1556.2	134817.	4424	294220	1555.6	27366.	5969	132146	0.0	4.93
c2_2_5	1759.7	1413.3	997	67540	1759.2	368.35	1098	22931	0.0	3.84
c2_2_6	1681.7	3003.6	1155	79219	1681.7	742.89	1250	29383	0.0	4.04
c2_2_7	1709.1	3694.6	1506	100137	1709.1	906.88	1533	34220	0.0	4.07
c2_2_8	1626.9	7029.2	1330	98417	1627.0	1292.0	1445	37099	0.0	5.44
c2_2_9	1635.2	11643.	1584	108311	1634.9	2112.0	1840	43214	0.0	5.51
c2_2_10	1572.1	13938.	1602	113454	1571.3	2543.9	1752	46140	0.1	5.48
r1_2_1	4654.9	26.963	199	10342	4651.5	16.450	363	4431	0.1	1.64
r1_2_2	3906.0	338.01	445	22615	3908.5	110.36	581	8513	-0.1	3.06
r1_2_3	3283.0	2155.4	676	31966	3528.2	216.20	859	12419	-7.5	9.97
r1_2_4	2925.6	4974.7	1081	56343	3749.5	83.842	678	11857	-28.	59.3
r1_2_5	3978.8	68.908	261	14902	3945.2	26.751	321	5028	0.8	2.58
r1_2_6	3383.4	605.13	458	23455	3442.9	110.04	539	8589	-1.8	5.50
r1_2_7	2971.8	2556.6	672	31896	3462.5	128.50	654	10654	-17.	19.9
r1_2_8	2777.4	5019.5	1357	74400	3701.8	135.88	864	15700	-34.	36.9
r1_2_9	3592.0	134.31	300	17795	3594.7	35.443	310	5635	-0.1	3.79
r1_2_10	3081.0	611.05	529	28032	3149.7	86.784	544	9846	-2.2	7.04
r2_2_1	3284.6	252.76	500	38393	3284.6	116.75	803	12874	0.0	2.17
r2_2_2	2438.0	5482.7	1069	82918	2438.0	964.81	1512	26275	0.0	5.68
r2_2_3	-	-	-	-	1851.1	10488.	3301	60935	-	-
r2_2_4	-	-	-	-	2181.1	4187.6	5641	134920	-	-
r2_2_5	2702.3	1143.0	784	65349	2702.3	314.76	1007	20114	0.0	3.63
r2_2_6	-	-	-	-	2051.4	4754.5	2050	41661	-	-
r2_2_7	-	-	-	-	1781.8	6469.1	3047	68141	-	-
r2_2_8	-	-	-	-	1524.8	25400.	6141	151513	-	-
r2_2_9	2372.8	4375.8	1069	92700	2372.9	872.38	1425	28674	0.0	5.02
r2_2_10	2057.4	51478.	1533	135588	2054.7	2828.0	1790	35968	0.1	18.2
rc1_2_1	34089.	77.122	254	14080	3335.6	28.971	251	4772	2.1	2.66
rc1_2_2	3079.3	701.38	477	22792	3064.3	170.52	610	10268	0.5	4.11
rc1_2_3	2851.5	2135.9	649	31119	3171.1	203.19	745	12443	-11.	10.5
rc1_2_4	2704.6	3337.9	1016	44426	3499.4	94.327	670	11472	-29.	35.4
rc1_2_5	3176.0	371.79	291	18201	3040.9	108.53	353	7519	4.3	3.43
rc1_2_6	3137.2	276.99	296	18063	3060.8	69.299	308	6950	2.4	4.00
rc1_2_7	3002.6	686.08	351	22094	2798.4	225.82	458	9610	6.8	3.04
rc1_2_8	2889.8	982.34	453	26034	3005.0	143.77	526	11799	-4.0	6.83
rc1_2_9	2870.4	985.56	427	25897	2906.3	161.31	474	10730	-1.3	6.11
rc1_2_10	2797.9	1378.7	516	30122	2765.2	261.12	658	14439	1.2	5.28
rc2_2_1	2418.1	1220.3	636	59964	2418.2	408.32	1094	16622	0.0	2.99
rc2_2_2	-	-	-	-	1910.3	4305.9	2020	35924	-	-
rc2_2_3	-	-	-	-	2248.7	1358.4	1925	42749	-	-
rc2_2_4	-	-	-	-	2010.2	12229.	9810	314665	-	-
rc2_2_5	-	-	-	-	2159.1	2153.8	1429	40285	-	-
rc2_2_6	-	-	-	-	2085.2	2778.7	1376	35622	-	-

rc2_2_7	-	-	-	-	1927.9	4422.9	1680	51136	-	-
rc2_2_8	-	-	-	-	1902.7	3276.8	1726	63367	-	-
rc2_2_9	-	-	-	-	1860.3	3542.1	1719	64506	-	-
rc1_2_10	-	-	-	-	1875.8	3580.7	2317	86154	-	-

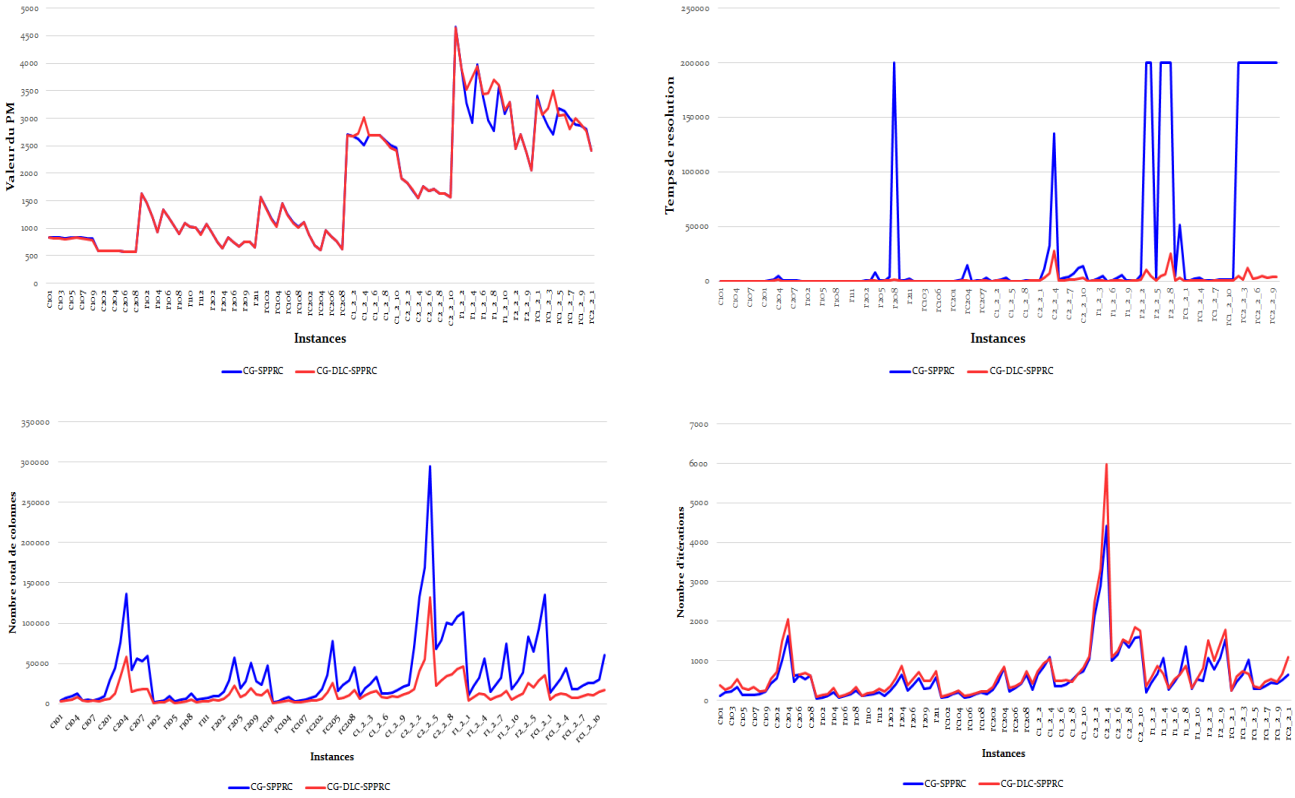

Figure 3.2: Impact de la DLC-SPPRC sur la qualité des solutions

A partir des tableaux tableau 3-2, tableau 3-3, tableau 3-4 et tableau 3-5, et la Figure 3.2, nous pouvons tirer les conclusions suivantes. Notre algorithme était plus rapide dans tous les cas (228). Dans 15 cas (7%), l'algorithme SPPRC n'a pas pu obtenir de solution dans un délai raisonnable (plus de 10 heures pour une itération). Notre algorithme était 10 fois plus rapide dans 37 cas (16% des cas) et 1,5 fois plus rapide dans 166 cas (73% des cas). En moyenne, nous avons obtenu un gain de 564% du temps de calcul. Pour la précision, dans 188 cas (82%), nous avons obtenu des précisions similaires (un écart inférieur à 1%) alors que la différence n'était supérieure à 1% que dans 25 cas (11%) et supérieure à 5% que dans 2 cas (0,88 %).

3.5 Conclusion

Dans ce chapitre, nous avons proposé un algorithme de solution approchée pour résoudre le SPPRC qui repose sur une relaxation lagrangienne. Notre méthode s'applique à la fois aux graphes acycliques et cycliques. La dominance s'applique uniquement à un sous-ensemble des ressources. Des schémas optimisés de mise à jour des paramètres sont utilisés pour assurer une convergence rapide. Lorsqu'elle est appliquée au VRP, notre approche offre un bon compromis entre précision et temps de calcul, et peut donc être appliquée à des problèmes pratiques à grande échelle. Il serait intéressant d'appliquer notre algorithme pour résoudre divers problèmes tels que branch-and-price, branch-and-cut et branch-and-price-and-cut ; et explorer des algorithmes d'optimisation plus avancés (heuristiques, méta-heuristiques, etc.) qui ont été appliqués avec succès dans d'autres domaines, tels que l'apprentissage en ligne, la planification, l'optimisation multi-objectifs, le transport, la médecine et la classification des données.

CHAPITRE 4

4 PROBLEME DE PLUS COURT CHEMIN ELEMENTAIRE AVEC CONTRAINTES DE RESSOURCES

4.1 Introduction

Le problème du plus court chemin élémentaire avec contraintes de ressources (ESPPRC) est une variante du SPPRC où le chemin de la solution est contraint d'être élémentaire [153], ce qui est requis, par exemple, dans le problème de tarification pour de nombreuses applications de tournées de véhicules [105]. Le ESPPRC a été identifié comme NP-difficile au sens fort [149] et a d'abord été étudié et résolu par Beasley et Christofides [153]. Les approches de résolution classiques sont basées sur le SPPRC non élémentaire correspondant, car il peut être résolu à l'aide d'algorithmes pseudo-polynomiaux [154], [155], [127]. Cependant, cette relaxation de sous-problème conduit parfois à des bornes inférieures faibles et éventuellement à de grands arbres branch-and-bound peu pratiques.

Comme mentionné dans le chapitre 3, l'approche standard pour résoudre exactement le SPPRC et le ESPPRC est basée sur la programmation dynamique. L'algorithme à correction d'étiquettes de Desrochers [123] étend l'algorithme de Ford-Bellman pour prendre en compte les contraintes de ressources. Le Tableau 4-1 résume certaines publications examinées qui ont appliqué la génération de colonnes pour les VRPs, classées par ordre chronologique en fonction des approches de solution utilisées pour résoudre le sous-problème ESPPRC.

Tableau 4-1: Recueil de quelques travaux basés sur la méthode sur le sous-problème ESPPRC.

Auteurs	Année	Classe du VRP
Feillet et al. [129]	2004	VRPTW
Righini and Salani [156]	2006	VRPTW
Chabrier [157]	2006	VRPTW
Feillet et al. [158]	2007	VRPTW

Tagmouti et al. [159]	2007	VRPTW
Jepsen et al. [160]	2008	VRPTW
Baldacci et al. [135]	2008	CVRP
Righini and Salani [161]	2008	VRPTW
Desaulniers et al, [162]	2008	VRPTW
Qureshi et al. [163]	2009	Autre
Baldacci et al. [137]	2011	CVRP
Bettinelli et al. [164]	2011	Autre
Liberatore et al. [165]	2011	VRPTW
Dabia et al. [138]	2013	Autre
Duque et al. [166]	2015	Autre
Lozano et al. [167]	2016	VRPTW
Pecin et al. [168]	2017	VRPTW
Pecin et al. [140]	2017	CVRP
Lera-Romero and Miranda-Bront [169]	2018	Autre
Ben Ticha et al. [131]	2019	VRPTW
Dalmeijer and Desaulniers [170]	2021	Autre
Mathlouthi et al. [143]	2021	Autre
Taş [171]	2021	Autre

Comme pour le problème SPPRC, l'inconvénient principal de l'algorithme ESPPRC est son temps de calcul élevé qui augmente avec le nombre de ressources. Pour accélérer les calculs et obtenir des solutions en un temps raisonnable, nous proposons un algorithme basé sur la relaxation lagrangienne pour résoudre approximativement le ESPPRC. Notre approche permet d'exprimer la dominance sur un sous-ensemble des ressources seulement, tandis que le reste des ressources est dualisé dans l'objectif.

Le reste du chapitre est organisé comme suit. La section 2 décrit formellement le ESPPRC. La section 3 présente notre procédure de solution approchée qui accélère les calculs. Cet algorithme est intégré à la génération de colonnes dans la section 4. Dans la section 5, nous présentons les résultats de calcul pour divers ensembles de données pour le ESPPRC et les problèmes de réalité virtuelle.

4.2 Problème de plus court chemin élémentaire avec contraintes de ressources

4.2.1 Notations et préliminaires

Nous reprenons les notations introduites au chapitre précédent. Considérons un graphe $G = (V, A)$ où $V = N \cup \{o, d\}$, est l'ensemble des nœuds $N = \{v_1, \dots, v_n\}$ qui seraient visités d'une origine $o = v_0$ à une destination $d = v_{n+1}$, et un ensemble d'arcs $A \subset V \times V$. On note $\mathcal{R}$ un ensemble de ressources de cardinalité $|\mathcal{R}|$. Un arc $(i, j) \in A$ de coût c_{ij} et consomme une quantité $t_{ij}^r \geq 0$ de chaque ressource $r \in \mathcal{R}$. On suppose que les ressources satisfont l'inégalité triangulaire. Si $(i, j) \in A$, alors le nœud i est appelé prédécesseur de j et j est appelé successeur de i. On appelle chemin une suite finie de nœuds :

$$P = (v_{k_0} = v_0, v_{k_1}, \dots, v_{k_m}), \quad \text{telque} \quad (v_{k_i}, v_{k_{i+1}}) \in A, \forall i = 0, \dots, m-1.$$

Le chemin P est dit élémentaire si chacun de ses nœuds est visité une fois.

A chaque nœud v_j de P, on associe C_j^P, la somme des coûts de ses arcs composites jusqu'à v_j. On note $T_j^{r,P}$ la quantité de ressource $r \in \mathcal{R}$ utilisée pour atteindre le nœud v_j. Par souci de simplification, nous supprimons l'indice P de ces notations et écrivons C_j et T_j^r. Le chemin P est dit réalisable s'il satisfait les contraintes de ressources

$$T_{k_i}^r \in [a_{k_i}^r, b_{k_i}^r], \quad \forall r \in \mathcal{R}, \ \forall i \in \{0, .., m\}.$$

Un chemin P (P est supposé contenir la source) peut aussi être représenté par $X = (x_{ij})_{(i,j)\in A} \in \{0,1\}$, où $x_{ij} = 1$ si les nœuds v_i et v_j appartiennent à P et 0 sinon.

4.2.2 Formulation du problème

Le ESPPRC consiste à trouver un chemin réalisable élémentaire de l'origine à la destination avec un coût minimal. De manière similaire à [127], le problème ESPPRC peut être formulé comme suit:

$$minimiser \quad \sum_{(i,j)\in A} c_{ij} x_{ij} \tag{4.1}$$

Vérifiant :

$$\sum_{(i,j)\in A} x_{ij} - \sum_{(i,j)\in A} x_{ji} = 0 \;\; ; \;\; \forall\, i \in V \setminus \{o,d\}, \tag{4.2}$$

$$\sum_{(o,j)\in A} x_{oj} = 1; \quad \sum_{(i,d)\in A} x_{id} = 1, \tag{4.3}$$

$$\sum_{(i,j)\in S\times S} x_{ij} \leq |S| - 1; \quad \forall S \subset \{v_0, \dots, v_n\}, \tag{4.4}$$

$$x_{ij}\left(T_i^r + t_{ij}^r - T_j^r\right) \leq 0, \quad \forall (i,j) \in A, \forall\, r \in \mathcal{R}, \tag{4.5}$$

$$a_j^r \leq T_j^r \leq b_j^r, \quad \forall\, j \in V, \forall\, r \in \mathcal{R}, \tag{4.6}$$

$$x_{ij} \in \{0,1\}, \quad \forall (i,j) \in A, \tag{4.7}$$

où $X = (x_{ij})_{i,j}$ représente le flux dans le graphe et définit ainsi un ensemble de solutions (chemin non dominé), les équations **(4.2)-(4.3)** définissent des contraintes de flux sur le graphe, les équations **(4.4)** sont des contraintes d'élimination de sous-tour, **(4.5)** code les exigences de compatibilité entre les variables de débit et de temps, et **(4.6)** est la contrainte de fenêtres de temps.

4.3 Algorithme de correction d'étiquette Lagrange ESPPRC

Pour résoudre le problème ESPPRC, on utilise un algorithme d'étiquetage similaire à celui du chapitre précédent, avec des règles de dominance permettant de limiter la prolifération des

étiquettes et réduire le temps de calcul, Feillet [129] a proposé la première approche exacte pour le ESPPRC.

4.3.1 Extension de chemin, étiquetage et dominance

Un chemin P arrivant au nœud v_i est étendu à chacun de ses successeurs j pour en obtenir un nouveau. Les coûts et ressources sont initialisés à 0 au sommet o et prolongés comme suit :

$$C_j = C_i + c_{ij} \quad \text{and} \quad T_j^r = \max\{a_j^r, T_i^r + t_{ij}^r\}, \quad \forall r \in \mathcal{R}. \tag{4.8}$$

Définition 1. Considérant un chemin P, un nœud est dit inaccessible par rapport à un ensemble de ressources $\bar{\mathcal{R}}$ s'il appartient à P ou lorsque son inclusion (par extension de P) viole les contraintes de ressources

$$a_j^r \leq T_j^r \leq b_j^r, \quad \forall\, j \in V, \forall\, r \in \bar{\mathcal{R}}. \tag{4.9}$$

Définition 2. A chaque chemin P_i arrivant au nœud v_i, on associe un vecteur étiquette (ou état)

$$L_i = \left(C_i, T_i^1; \cdots; T_i^{|\mathcal{R}|}, s_i, V_i^0; \cdots; V_i^{n+1}\right). \tag{4.10}$$

Où $V_i^k = 1$ si le nœud v_k est inaccessible et $V_i^k = 0$ sinon, et $s_i = \sum_{j=0}^{n+1} V_i^j$ est le nombre de nœuds inaccessibles. Le chemin associé à une étiquette L est noté X^L.

Les valeurs V_i^r stockent les nœuds précédemment visités le long du chemin P_i, permettant ainsi de détecter des cycles. Le scalaire s_i est inclus pour accélérer les calculs en filtrant certains chemins peu prometteurs.

Définition 3. Soient P_i et $P_i{}'$ deux chemins réalisables arrivant à un nœud v_i. On dit que $P_i{}'$ est dominé par P_i et on écrit $P_i \preccurlyeq P_i{}'$ si le vecteur étiquette de P_i est inférieur en composantes à celui de $P_i{}'$.

Définition 4.

Soient P_i et P_i' deux chemins arrivant à un nœud v_i. On dit que P_i est inférieur à P_i' et on écrit $P_i \leq P_i'$ si la première composante de P_i est plus petite que celle de P_i'. Si les deux premières composantes sont égales, on compare récursivement les suivantes. Au moins une des inégalités doit être stricte. Cela définit une relation d'ordre partiel.

4.3.2 Procédure de solution approximative

Considérons le problème **(4.1)-(4.7)** avec les contraintes **(4.2)-(4.7)**. Nous proposons d'intégrer un sous-ensemble prédéfini de ressources $\mathcal{R}_1 \subset \mathcal{R}$ dans l'objectif dans une relaxation lagrangienne à multiplicateurs duaux. Ensuite, nous appliquons la dominance sur les ressources restantes $\mathcal{R}_2 = \mathcal{R} \setminus \mathcal{R}_1$ uniquement, ce qui accélère le temps de calcul, mais pourrait supprimer les solutions optimales et même conduire à une solution irréalisable. Cela revient à considérer les équations **(4.5)-(4.6)** sur $\mathcal{R}_2$ et à remplacer l'Équation **(4.1)** par

$$minimiser \quad \mathcal{L}(\lambda, X), \tag{4.11}$$

Où la fonction lagrangienne $\mathcal{L}$ est définie par

$$\mathcal{L}(\lambda, X) = \sum_{(i,j)\in A} x_{ij} \left(c_{ij} + \sum_{r\in\mathcal{R}_1} \lambda_{i,j}^r \left(\max(a_j, T_i^r + t_{ij}^r) - b_j^r\right)\right). \tag{4.12}$$

Comme le flux est supporté par au plus un arc entrant au nœud j depuis ses prédécesseurs i, les multiplicateurs de Lagrange $\lambda_{i,j}^r = \lambda_j^r$ sont indépendants de i. Pour chaque nœud $j \in V \cup \{d\}$, le vecteur multiplicateurs $\lambda_j = (\lambda_j^r)_{r\in\mathcal{R}_1}$peut être obtenu en résolvant le problème de maximisation suivant

$$\begin{cases} \max\limits_{\lambda} & \min\limits_{X} \mathcal{L}(\lambda, X), \\ s.t. & \lambda_j^r \geq 0, \qquad \forall r \in \mathcal{R}_1 \end{cases}. \tag{4.13}$$

Pour résoudre ce sous-problème d'optimisation, on peut utiliser un algorithme de sous-gradient projeté qui s'est avéré efficace en pratique. Le schéma de mise à jour est

$$\lambda_j^{k+1} = \max\{\lambda_j^k + \tau_j^k g_j^k, 0\}, \tag{4.14}$$

où $g_j^k \in \partial\mathcal{L}(\lambda, X)$ est un vecteur de sous-gradient, $\tau_j^k > 0$ est une taille de pas, et l'opérateur max est pris pour chaque élément. Nous choisissons

$$g_j^k = \max(a_j, T_i^r + t_{ij}^r) - b_j^r, \tag{4.15}$$

$$\tau_j^k = \frac{\theta_k(Z_{\text{UB}} - \mathcal{L}(\lambda^k, X))}{||g_j^k||^2}, \tag{4.16}$$

Où Z_{UB} est la borne la plus petite (supérieure) la plus connue de la solution du problème (3.1)-(3.6) et $\theta_k \in (0,2]$ est un scalaire.

Précisons que nous n'avons effectué qu'une seule itération pour l'estimation de λ qui s'est avérée suffisante (précision et rapidité) lorsqu'elle a été intégrée à la génération de colonnes.

La dominance en chaque nœud j correspond à la détermination des optima de Pareto du problème multicritère appliqué à un ensemble d'étiquettes. Avec l'approche lagrangienne adoptée, le vecteur label devient

$$L_j = \left(C_j + \sum_{r_1 \in \mathcal{R}_1} \lambda_j^{r_1} (T_j^{r_1} - b_j^{r_1});\ T_j^{r_2}, r_2 \in \mathcal{R}_2;\ s_j; V_j^0; \cdots; V_j^{n+1} \right), \tag{4.17}$$

Comme expliqué au début de cette section, la solution finale obtenue peut ne pas être réalisable. Dans ce cas, on résout à nouveau le ESPPRC relaxé en dominant sur le coût lagrangien déterminé dans la première phase et sur les ressources $\mathcal{R}_2$ et en validant les fenêtres de ressources dans l'espace d'origine $\mathcal{R}$.

Maintenant, nous décrivons notre algorithme pour déterminer les solutions approchées du ESPPRC, qui améliore l'algorithme de Feillet [129] largement utilisé. En supposant que tous les

prédécesseurs du nœud $j \in N$ ont été considérés, la dominance au nœud j peut être interprétée comme la détermination des optima de Pareto pour le problème multicritère des fonctions $|V| + |\mathcal{R}_2| + 2$:

$$\min_{(i;(i,j)\in A)} \left(C_i + c_{ij} \; + \sum_{r_1 \in \mathcal{R}_1} \lambda_j^{r_1} (T_j^{r_1} - b_j^{r_1}); \; \max\left\{a_j^{r_2}, (T_i^{r_2} + t_{ij}^{r_2})\right\}, r_2 \in \mathcal{R}_2; \; s_j; V_j^0; \cdots; V_j^{n+1} \right) \quad \textbf{(4.18)}$$
$$\text{st. } T_i^r + t_{ij}^r \leq b_j^r, r \in \mathcal{R}.$$

La relation de dominance ≤ est une relation d'ordre partiel, le nombre effectif d'étiquettes à traiter augmente exponentiellement avec le nombre de ressources, ce qui rend la procédure très difficile à étendre. Pour mieux expliquer notre algorithme de correction d'étiquette modifié, nous introduisons les notations suivantes.

Notations

• E_i : Liste des étiquettes au nœud v_i.

• F_i : La liste des chemins associés aux labels de E_i.

• S_i : Ensemble des successeurs des indices du nœud v_i.

• Q : Liste des index des nœuds non traités.

• $Pareto(L, E_j)$: Ajouter une nouvelle étiquette L à un ensemble d'étiquettes non dominées E_j tout en gardant la propriété de non-dominance.

• Z_L est le coût associé à une étiquette L (première composante de l'Équation **(4.17)**).

Notre procédure améliorée de correction d'étiquettes est détaillée dans l'algorithme 4.1, l'algorithme 4.2 et l'algorithme 4.3.

Algorithme 4.1 : Algorithme de correction d'étiquettes modifié

Entrée : Un graphe orienté arbitraire $G = (V, A)$, non nécessairement acyclique ; $\mathcal{R}$ est un ensemble de contraintes ; $\mathcal{R}_1 \subset \mathcal{R}$ sous-ensemble de contraintes à inclure dans la fonction objective ; Vecteur de multiplicateurs de Lagrange $\lambda_i \in R_+^{|\mathbb{R}_1|}$ où $v_i \in V - \{o\}$; $\bar{\mathcal{R}}$ est un ensemble de ressources utilisées pour définir les nœuds inaccessibles (Définition 1), c'est-à-dire. $\left(V_i^k\right)$.
Sortie : Tous les chemins non-dominants de la source à la destination E_{n+1} et la liste associée des chemins F_{n+1}.

// Initialisation ;
$\mathcal{R}_2 = \mathcal{R} - \mathcal{R}_1$;

for $v_i \in V$ **do**
  $E_i \leftarrow \emptyset$
end

$E_0 \leftarrow \{(0,0,\cdots,0,1,1,0,\cdots,0)\}$; // La taille des libellés est : $|E_o| = |V| + |\mathcal{R}_2| + 2$
$Q \leftarrow \{0\}$

while $Q \neq \emptyset$ **do**
  // Choisir un nœud à traiter
  Choisir $i \in Q$;
  for $j \in S_i$ **do**
    for $L_i \in E_i$ **do**
      if $V_i^j = 0$ **then**
        // Créer de nouveaux libellés L_j pour j en étendant ceux de i;

$$L_j = \left(C_j + \sum_{r_1 \in \mathcal{R}_1} \lambda_j^{r_1} \left(T_j^{r_1} - b_j^{r_1}\right);\ T_j^{r_2}, r_2 \in \mathcal{R}_2;\ s_j; V_j^0; \cdots; V_j^{n+1} \right).$$

        // Appliquer la domination (Voir Algorithme 4.2);
        $E_j \leftarrow Pareto\left(L_j, E_j\right)$
        // Mettre à jour F_j;
        Mettre à jour F_j en utilisant E_j.
      end
    end
    if L_j a changé **then**
      $Q \leftarrow Q \cup \{j\}$
    end
  end
  $Q \leftarrow Q \cup \{j\}$
end

Algorithme 4.2 : $Pareto(L, E)$

Entrée : Nouvelle étiquette L, Ensemble d'étiquettes E.
Sortie : Sous-ensemble des étiquettes non-dominées.

$continu \leftarrow$ true
$add \leftarrow$ true
while ($E \neq \emptyset$ and continu) **do**
  Choose $L_0 \in E$;
  if ($L_0 \leq L$) **then**
    $continu \leftarrow$ false
    $add \leftarrow$ false
  else
    if ($L_0 > L$) **then**
      $E \leftarrow E \setminus \{L_0\}$
if (add) **then**
  $E \leftarrow E \cup \{L\}$

Algorithme 4.3 : DLC-ESPPRC : Dominance on Lagrangian cost for the ESPPRC

Entrée : Un graphe orienté arbitraire $G = (V, A)$, non nécessairement acyclique ; $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$ est un ensemble de contraintes de ressources composé de deux types.
Sortie : $\bar{E}$, Tous les chemins réalisables non-dominés du nœud source au nœud de destination d.
// Choix des paramètres ;
Choisir $\theta \in (0,\ 2], k_{max} = 10$.
// Initialisation ;
for $v_i \in V - \{o\}$ **do**
$\lambda_i^{(0)} = (0, \ldots, 0)$ est un vecteur de taille $|\mathcal{R}_1|$.
// Valeur initiale du multiplicateur de Lagrange correspondant à la solution de (4.13)
$\bar{Z}_{max} \leftarrow -\infty$.
// Choisir une solution initiale pour le problème (4.1) -(4.7).
$X^0 = \left(x_{ij}^0\right)_{ij}$.
// Calculer la valeur de la fonction objectif (4.1) (borne supérieure).
$Z_u \leftarrow \sum_{(i,j)\in A} c_{ij} x_{ij}^0$.
$\bar{E} \leftarrow \emptyset$.
$k = 0$.
// Étape 1;

while $(k < k_{max})$ **do**
    // Appliquer la relaxation de dominance Lagrangienne ;
    $[E_{n+1}, F_{n+1}]$ = **Algorithm 4.1** $(G, \left(\lambda_i^{(k)}\right)_i, \mathcal{R}, \mathcal{R}_1, \mathcal{R}_2)$.
    $(L, X^L) \leftarrow$ est la plus petite étiquette de E_{n+1} (Définition 4) et son chemin associé F_{n+1}.
    // Mettre à jour $\bar{Z}_{max}$
    if $(\bar{Z}_{max} < Z_L)$ **then**
        $\bar{Z}_{max} \leftarrow Z_L$
    else
        $\theta \leftarrow \theta/2$.
    if X^L est réalisable **then**
        if $Z_u > Z_L$ **then**
            $Z_u \leftarrow Z_L$
            $\bar{E} \leftarrow$ Toutes les solutions réalisables de E_{n+1}.
    // Mettre à jour les multiplicateurs de Lagrange
    for $v_i \leftarrow V - \{o\}$ **do**
        $\lambda_i^{(k+1)} \leftarrow$ Appliquer les équations (4.15), (4.16), (4.14).
    if $\left\|\lambda_i^{(k+1)} - \lambda_i^{(k)}\right\| < \varepsilon \ \forall i$ **then**
        break
    $k \leftarrow k + 1$.
// Étape 2
if $\bar{E} = \emptyset$ **then**
    $[\bar{E},]$ = **Algorithm 4.1** $(G, \left(\lambda_i^{(k)}\right)_i, \mathcal{R}, \mathcal{R}_1, \mathcal{R})$.

Cet algorithme peut facilement être adapté pour produire tous les chemins non dominés de la source à tous les autres nœuds du graphe $(E_j)_{j\in \mathrm{V}}$.

4.4 Intégration dans la génération de colonnes

Le problème de tournées de véhicules avec fenêtres de temps ($VRPTW$) est défini par n clients et K véhicules disponibles dans un référentiel. Un parcours est une suite de visites (effectuées par un même véhicule) respectant les plages horaires, qui débute et se termine au même dépôt. Le VRPTW peut être exprimé sur un graphe G. Le dépôt est représenté par les deux nœuds o et d. Chaque client i est associé au nœud v_i et visité par un véhicule k pour satisfaire une demande d_i, pendant une fenêtre de temp donnée $[a_i^k, b_i^k]$. De plus, nous réutilisons les notions de coût d'arc c_{ij}^k, de durée d'arc t_{ij}^k. Nous ajoutons également une nouvelle contrainte sur la capacité de chaque véhicule k, notée Q_k. Un itinéraire est dit faisable si la demande cumulée des clients visités par un véhicule k n'excède pas Q_k et lorsque les contraintes de temps sont respectées. Le VRPTW consiste à trouver des itinéraires réalisables à coût minimum où chaque client est visité par un véhicule. Cela peut être résolu en tant que K problèmes ESPPRC indépendants, un pour chaque véhicule, avec deux ressources : la durée de l'arc et les demandes des clients. Pour prendre en compte des types de véhicules hétérogènes, le ESPPRC doit être résolu séparément pour chaque type de véhicule.

Pour résoudre le VRPTW, nous utilisons la décomposition de Dantzig-Wolfe qui définit K sous-problèmes indépendants et un problème maître global. En appliquant une génération de colonnes, nous résolvons alternativement le problème maître et les K sous-problèmes.

Le problème maître est réécrit comme un partitionnement d'ensembles [155]:

$$minimiser \quad \sum_{p \in \Omega} c_p y_p \tag{4.19}$$

Vérifiant :

$$\sum_{p \in \Omega} e_{ip} y_p = 1, \quad i \in \{1, \ldots, n\}, \tag{4.20}$$

$$y_p \in \{0,1\}, \qquad \forall p \in \Omega, \tag{4.21}$$

où Ω est l'ensemble de toutes les routes possibles, C_p représente le coût d'une route $p \in \Omega$ (somme des coûts de ses arcs composites), $e_{ip} = 1$ si et seulement si le client i est visité par la route p, et y_p indique si oui ou non la route p appartient à la solution (sous-ensemble de routes). On applique une relaxation continue sur les contraintes. Comme noté dans [129], une telle relaxation n'affecte pas l'ensemble des solutions. De plus, alors que le nombre de routes éligibles est généralement exponentiel, il est possible de trouver en un temps raisonnable une solution optimale sur un petit sous-ensemble de routes (c'est-à-dire des colonnes de la matrice de contraintes). Ainsi, nous résolvons itérativement un problème maître restreint ($\overline{RMP}^{(t)}$):

$$minimiser \quad \sum_{p \in \Omega^t} c_p y_p \tag{4.22}$$

Vérifiant :

$$\sum_{p \in \Omega^t} e_{ip} y_p \geq 1, \quad i \in \{1, \ldots, n\}, \tag{4.23}$$

$$y_p \in \{0,1\}, \quad \forall p \in \Omega^t, \tag{4.24}$$

où Ω^t est un sous-ensemble de routes définies à chaque itération t par génération de colonnes.

Si $\delta^{(t)}$ désigne le vecteur de variables duales associé à la solution des ($\overline{RMP}^{(t)}$), la route de moindre coût négatif réduit $p^{(t)}$ est définie par

$$p^{(t)} = \arg \min_{p \in \Omega^t} (c_p - \sum_{i=1}^{n} \delta_i^{(t)} e_{ip}) \tag{4.25}$$

qui définit le sous-problème, qui peut être résolu comme un ESPPRC avec des coûts d'arc égaux à $c_{ij} - \delta_i^{(t)}$.

Le processus itératif de résolution du problème maître restreint (4.22)-(4.24) et du sous-problème (4.25) est arrêté lorsque toutes les voies ont un coût réduit positif dans la résolution du problème, ce qui indique que l'optimum a été atteint. Si la génération de colonnes produit une solution fractionnaire à la fin, nous obtenons alors une solution réalisable en nombre entier en résolvant

un problème de programmation en nombre entier sur toutes les colonnes incluses par le RMP final.

4.5 Expériences

Dans un premier temps, nous avons considéré le problème ESPPRC. Deuxièmement, nous avons abordé les instances VRPTW avec une approche de génération de colonne en utilisant le ESPPRC comme sous-problème. Dans les deux cas, nous avons utilisé les jeux de données Solomon [152]. Nous avons appliqué notre algorithme approché DLC-ESPPRC et la méthode bien connue de Feillet (ESPPRC) [129]. Précisons que, dans ce dernier, un graphe réduit est utilisé alors que nous avons conservé le graphe complet pour mieux évaluer les performances de notre algorithme. À des fins de comparaison, comme dans [129], nous avons considéré deux ressources et un graphe aléatoire coûts la génération.

Nous avons implémenté notre algorithme en utilisant le langage de programmation Java. Pour la simulation, nous avons utilisé un CPU Intel Core i9-9900KF (8 cœurs), 3,60 GHz, RAM 32 Go, fonctionnant sous Windows 10 (64 bits). Pour le ESPPRC, nous avons implémenté l'algorithme de Feillet et obtenu des résultats identiques (fonction objectif). Lors des simulations, le temps de calcul constaté avec notre implémentation était, en moyenne, le double de celui rapporté dans [129]. Cela peut être exprimé par le langage de programmation utilisé, les ressources de calcul, et le fait que Feillet et al. ont utilisé le graphique réduit. Nous rapportons ici les temps de calcul obtenus avec notre implémentation, à titre de comparaison avec notre algorithme introduit.

4.5.1 Comparaison entre le ESPPRC et le DLC-ESPPRC

Pour chaque problème de données, nous avons effectué plusieurs exécutions (jusqu'à 12). Les solutions sont ensuite moyennées pour chaque type de problème et les résultats sont rapportés dans les tableaux Tableau 4-2, Tableau 4-3 et

Tableau 4-4. Nous avons calculé le rapport des arcs à coût négatif ($\boldsymbol{N_a}$), la valeur de la fonction objectif ($\boldsymbol{O_i}$), le temps de calcul en secondes ($\boldsymbol{T_i}$), le nombre d'étiquettes à destination après convergence ($\boldsymbol{L_i}$), où $\boldsymbol{i = 1}$ pour ESPPRC et $\boldsymbol{i = 2}$ pour DLC-ESPPRC. On choisit une valeur initiale du paramètre $\boldsymbol{\theta = 0.25}$ pour toutes les instances, sauf C103 pour laquelle on choisit $\boldsymbol{\theta = 0.01}$. Notre algorithme est plus rapide que l'algorithme de Feillet sur les instances de Solomon modifiées dans 134 des 168 instances. En moyenne, nous avons réduit le temps d'exécution de 26% et le nombre de chemins partiels de 47% ce qui constitue une réduction effective de l'espace de solution. Par conséquent, la vitesse de résolution de divers VRP à l'aide de l'algorithme de branchement et de prix qui utilise le ESPPRC peut également être améliorée.

Tableau 4-2: Comparaison du ESPPRC et le DLC-ESPPRC des instances de Solomon modifiées avec 25 clients.

		ESPPRC			DLC-ESPPRC			Comparison	
Instance	N_a	O_1	T_1	L_1	O_2	T_2	L_2	T_1/T_2	L_1/L_2
C101	22.62	-66.4	0.0325	27	-66.4	0.0250	11	1.30	2.45
C102	24.15	-111.5	0.1735	1058	-111.5	0.1008	594	1.72	1.78
C103	24.46	-118.8	1.6653	1717	-118.8	0.6767	987	2.46	1.74
C104	22.15	-137.3	38.357	8189	-137.3	29.093	6660	1.32	1.23
C105	23.85	-86.6	0.0020	47	-86.6	0.0014	23	1.43	2.04
C106	22.92	-69.6	0.0287	28	-69.6	0.0241	13	1.19	2.15
C107	23.85	-105.6	0.0023	46	-105.6	0.001	32	2.30	1.44
C108	22.92	-83.2	0.0043	80	-83.2	0.0019	47	2.26	1.70
C109	25.23	-115	0.0116	159	-115	0.0056	104	2.07	1.53
C201	17.38	-91.9	0.0032	37	-91.9	0.0019	11	1.68	3.36
C202	14.46	-92.5	0.1335	524	-92.5	0.0343	163	3.89	3.21
C203	16.92	-133.9	30.389	17429	-133.9	15.163	10207	2.00	1.71
C204	16.46	-154.8	10232	282041	-154.8	5147.1	202306	1.99	1.39
C205	16.62	-105.2	0.0050	124	-105.2	0.0021	33	2.38	3.76
C206	14.62	-153.9	0.0134	353	-153.9	0.0037	70	3.62	5.04
C207	16.46	-154.6	0.9546	4192	-154.6	0.3846	1998	2.48	2.10
C208	16.62	-113.6	0.0290	435	-113.6	0.0096	187	3.02	2.33
R101	5.38	-9.8	0.0017	10	-9.8	0.0015	6	1.13	1.67
R102	6.46	-16.3	0.0117	25	-16.3	0.0060	21	1.95	1.19
R103	5.85	-13.7	0.0171	31	-13.7	0.0098	29	1.74	1.07
R104	6.31	-40.5	0.0113	68	-40.5	0.0043	65	2.63	1.05
R105	5.08	-16.2	0.0008	15	-16.2	0.0003	8	2.67	1.88
R106	6.92	-21.1	0.0025	32	-21.1	0.0013	28	1.92	1.14
R107	5.69	-21.9	0.0021	42	-21.9	0.002	38	1.05	1.11
R108	5.38	-26.2	0.0102	90	-26.2	0.0086	84	1.19	1.07
R109	6.15	-14.5	0.0006	16	-14.5	0.0005	12	1.20	1.33
R110	5.85	-8.1	0.0013	22	-8.1	0.0010	18	1.30	1.22
R111	6.62	-18.6	0.0020	51	-18.6	0.0018	50	1.11	1.02
R112	6.00	-26.3	0.0033	67	-26.3	0.0031	64	1.06	1.05
R201	6.31	-7.8	0.0007	14	-7.8	0.0005	9	1.40	1.56
R202	5.69	-13.6	0.0028	46	-13.6	0.0025	42	1.12	1.10
R203	6.62	-38.8	0.0169	84	-38.8	0.0150	83	1.13	1.01
R204	5.23	-30.9	0.0251	142	-30.9	0.0225	139	1.12	1.02
R205	5.08	-7.9	0.0016	31	-7.9	0.0012	26	1.33	1.19
R206	4.15	-11.5	0.0043	52	-11.5	0.0034	42	1.26	1.24
R207	5.54	-37.2	0.0536	242	-37.2	0.0453	215	1.18	1.13
R208	4.62	-43.2	0.2008	775	-43.2	0.1864	705	1.08	1.10
R209	4.92	-24.6	0.0052	55	-24.6	0.0037	49	1.41	1.12
R210	5.54	-28.4	0.0151	90	-28.4	0.0135	86	1.12	1.05
R211	5.69	-24.4	0.0188	117	-24.4	0.0169	111	1.11	1.05
RC101	17.23	1.5	0.0006	34	1.5	0.0004	18	1.50	1.89
RC102	18.92	-18.4	0.0037	129	-18.4	0.0031	90	1.19	1.43
RC103	18.46	-32.8	0.0091	249	-32.8	0.0071	176	1.28	1.41
RC104	15.85	-28	0.0244	367	-28	0.0219	309	1.11	1.19
RC105	18.77	-22.8	0.0026	141	-22.8	0.0023	122	1.13	1.16
RC106	16.62	-22.5	0.0019	81	-22.5	0.0014	47	1.36	1.72
RC107	15.85	-48.4	0.0120	290	-48.4	0.0100	241	1.20	1.20
RC108	18.31	-46.2	0.0602	862	-46.2	0.0568	826	1.06	1.04
RC201	16.92	-18.1	0.0015	25	-18.1	0.0010	14	1.50	1.79
RC202	16.46	-55.5	0.0510	476	-55.5	0.0270	319	1.89	1.49
RC203	18.46	-73.6	3.1893	1880	-73.6	2.1306	1477	1.50	1.27
RC204	19.08	-99.7	5395.7	127192	-99.7	3785.3	93215	1.43	1.36
RC205	16.00	-43.3	0.0341	206	-43.3	0.0182	112	1.87	1.84
RC206	18.62	-37.2	0.0125	156	-37.2	0.0081	86	1.54	1.81
RC207	17.08	-64.7	0.1821	1390	-64.7	0.1243	928	1.47	1.50
RC208	16.62	-94.3	102.96	25336	-94.3	75.872	21074	1.36	1.20

Tableau 4-3: Comparaison du ESPPRC et le DLC-ESPPRC des instances de Solomon modifiées avec 50 clients.

		ESPPRC			DLC-ESPPRC			Comparison	
Instance	N_a	O_1	T_1	L_1	O_2	T_2	L_2	T_1/T_2	L_1/L_2
C101	13.37	-91.3	0.0752	68	-91.3	0.0467	22	1.61	3.09
C102	13.92	-110.2	0.5424	982	-110.2	0.3055	568	1.78	1.73
C103	14.16	-142.6	436.50	30999	-142.6	284.15	20286	1.54	1.53
C104	14.00								
C105	13.25	-83.5	0.0089	50	-83.5	0.0048	11	1.85	4.55
C106	13.80	-101.9	0.008	41	-101.9	0.006	22	1.33	1.86
C107	14.71	-92.5	0.0107	75	-92.5	0.0059	38	1.81	1.97
C108	14.20	-94.4	0.029	118	-94.4	0.0139	71	2.09	1.66
C109	14.16	-121.5	0.0882	225	-121.5	0.0439	178	2.01	1.26
C201	10.24	-198.6	0.0531	251	-198.6	0.0105	34	5.06	7.38
C202	9.96	-188.6	105.54	21766	-188.6	4.843	2611	21.79	8.34
C203	10.67								
C204	10.82								
C205	9.61	-218.6	0.2029	905	-218.6	0.0282	114	7.20	7.94
C206	10.00	-282.3	1.0527	2021	-282.3	0.1106	537	9.52	3.76
C207	10.27	-291.9	18.589	8523	-291.9	4.3147	4421	4.31	1.93
C208	10.47	-305.5	6.8473	3803	-305.5	0.6319	1317	10.84	2.89
R101	5.92	-17	0.0018	13	-17	0.0013	5	1.38	2.60
R102	5.92	-40.8	0.0267	152	-40.8	0.0231	131	1.16	1.16
R103	5.96	-50.4	0.0747	152	-50.4	0.0676	129	1.11	1.18
R104	6.24	-89.1	9.0622	6446	-89.1	7.7583	5518	1.17	1.17
R105	6.51	-39.8	0.0041	17	-39.8	0.0028	8	1.46	2.13
R106	6.04	-58.7	0.0484	192	-58.7	0.036	147	1.34	1.31
R107	6.63	-75.3	0.365	738	-75.3	0.2769	642	1.32	1.15
R108	6.24	-71.4	4.6012	5158	-71.4	3.7994	4481	1.21	1.15
R109	6.20	-38.8	0.0116	47	-38.8	0.0079	22	1.47	2.14
R110	5.88	-44.4	0.0498	62	-44.4	0.0309	44	1.61	1.41
R111	6.78	-80.3	0.0889	225	-80.3	0.0663	195	1.34	1.15
R112	6.31	-61.7	0.2283	314	-61.7	0.165	283	1.38	1.11
R201	6.47	-58	0.0114	37	-58	0.0067	25	1.70	1.48
R202	5.96	-103.1	55.596	12046	-103.1	26.764	7780	2.08	1.55
R203	6.12								
R204	6.39								
R205	6.00	-82	0.1746	280	-82	0.0751	143	2.32	1.96
R206	6.39	-139	7956.5	159005	-139	26490	92806	3.00	1.71
R207	6.86								
R208	6.43								
R209	6.00	-87.8	34.361	9377	-87.8	9.8462	4639	3.49	2.02
R210	6.16								
R211	6.35								
RC101	11.22	-6.7	0.0030	60	-6.7	0.0023	44	1.30	1.36
RC102	10.47	-37	0.0252	303	-37	0.0206	252	1.22	1.20
RC103	10.94	-63.2	0.1822	748	-63.2	0.1334	637	1.37	1.17
RC104	10.90	-71.3	0.9317	2196	-71.3	0.7943	1910	1.17	1.15
RC105	11.06	-16.1	0.0229	305	-16.1	0.0139	236	1.65	1.29
RC106	11.06	-47.4	0.0154	149	-47.4	0.01	122	1.54	1.22
RC107	10.86	-35.1	0.1117	773	-35.1	0.0889	655	1.26	1.18
RC108	10.20	-59.9	0.6536	3168	-59.9	0.611	2927	1.07	1.08
RC201	11.18	-40.1	0.0151	90	-40.1	0.0091	41	1.66	2.20
RC202	10.94	-82.7	12.646	4503	-82.7	4.9088	2308	2.58	1.95
RC203	11.14								
RC204	10.75								
RC205	10.71	-71	0.2819	621	-71	0.1405	320	2.01	1.94
RC206	10.47	-74.3	0.1471	360	-74.3	0.0936	230	1.57	1.57
RC207	11.25	-126.6	464.34	38924	-126.6	170.07	26759	2.73	1.45
RC208	11.22								

Tableau 4-4: Comparaison du ESPPRC et le DLC-ESPPRC des instances de Solomon modifiées avec 100 clients.

		ESPPRC			DLC-ESPPRC			Comparison	
Instance	N_a	O_1	T_1	L_1	O_2	T_2	L_2	T_1/T_2	L_1/L_2
c101	6.99	-105.1	0.1815	96	**-103.8**	0.1768	26	1.03	3.69
c102	7.22	-129.5	10.139	371	-129.5	4.0248	213	2.52	1.74
c103	7.15	-139.4	3082.7	145182	-139.4	6162.8	121567	**0.50**	1.19
c104	6.93								
c105	6.75	-96.6	0.0741	89	-96.6	0.0439	31	1.69	2.87
c106	7.03	-101.9	0.1229	42	-101.9	0.0696	21	1.77	2.00
c107	7.02	-109.6	0.0892	70	-109.6	0.05	38	1.78	1.84
c108	6.82	-117.4	0.1891	163	-117.4	0.1168	85	1.62	1.92
c109	7.20	-134.3	0.7145	643	-134.3	0.4016	378	1.78	1.70
c201	6.17	-239.7	0.3631	306	-239.7	0.0882	43	4.12	7.12
c202	6.43								
c203	6.11								
c204	6.23								
c205	6.30	-284.3	2.9741	1276	-284.3	0.4389	288	6.78	4.43
c206	6.29	-294.8	8.9111	3511	-294.8	1.0416	712	8.56	4.93
c207	6.27	-3109	770.34	13546	-310.9	43.551	4136	17.69	3.28
c208	6.32	-323.1	55.227	9985	-323.1	8.7077	3508	6.34	2.85
r101	7.78	-55.8	0.0253	100	-55.8	0.012	20	2.11	5.00
r102	8.31	-130.3	40.031	14538	**-127.6**	69.775	10950	**0.57**	1.33
r103	7.77	-139.4	3238.7	154182	-139.4	6052.1	121567	**0.54**	1.27
r104	7.79								
r105	8.04	-101	0.0814	179	-101	0.0396	40	2.06	4.48
r106	7.69	-150.6	67.921	19398	-150.6	85.162	11022	**0.80**	1.76
r107	7.93								
r108	7.70								
r109	8.09	-117.1	0.4476	619	-117.1	0.3985	197	1.12	3.14
r110	7.71	-124.2	5.9877	3071	-124.2	3.3088	1132	1.81	2.71
r111	8.43	-147.6	81.694	20080	-147.6	78.554	11804	1.04	1.70
r112	8.25	-161	653.51	79018	-161	767.98	58556	**0.85**	1.35
r201	8.02	-168.1	1.7684	1580	-168.1	0.5749	407	3.08	3.88
r202	8.26								
r203	7.72								
r204	8.16								
r205	8.02	-298.2	19640	80905	-298.2	174.00	27431	11.29	2.95
r206	8.52								
r207	8.04								
r208	8.18								
r209	7.65								
r210	7.98								
r211	8.19								
rc101	5.07	-50	0.1491	77	-50	0.0723	22	2.06	3.50
rc102	4.92	-86.4	0.6175	412	-86.4	0.375	257	1.65	1.60
rc103	5.13	-98.4	11.704	5534	**-96.1**	10.733	2816	1.09	1.97
rc104	4.93	-107.4	1313.5	64331	-107.4	1150.0	54236	1.14	1.19
rc105	4.72	-72.4	0.2069	67	-72.4	0.1348	25	1.53	2.68
rc106	5.27	-76.5	0.1907	121	-76.5	0.1766	51	1.08	2.37
rc107	5.16	-92.4	1.7064	324	-92.4	1.6783	215	1.02	1.51
rc108	5.07	-108.8	9.0981	2018	-108.8	12.797	1530	**0.71**	1.32
rc201	4.94	-95.1	0.4211	210	-95.1	0.1667	52	2.53	4.04
rc202	4.88								
rc203	5.19								
rc204	5.21								
rc205	5.12	-201	116.86	17352	-201	36.348	7099	3.22	2.44
rc206	5.18	-214.5	42.717	13857	-214.5	16.833	7349	2.54	1.89
rc207	5.14								
rc208	5.20								

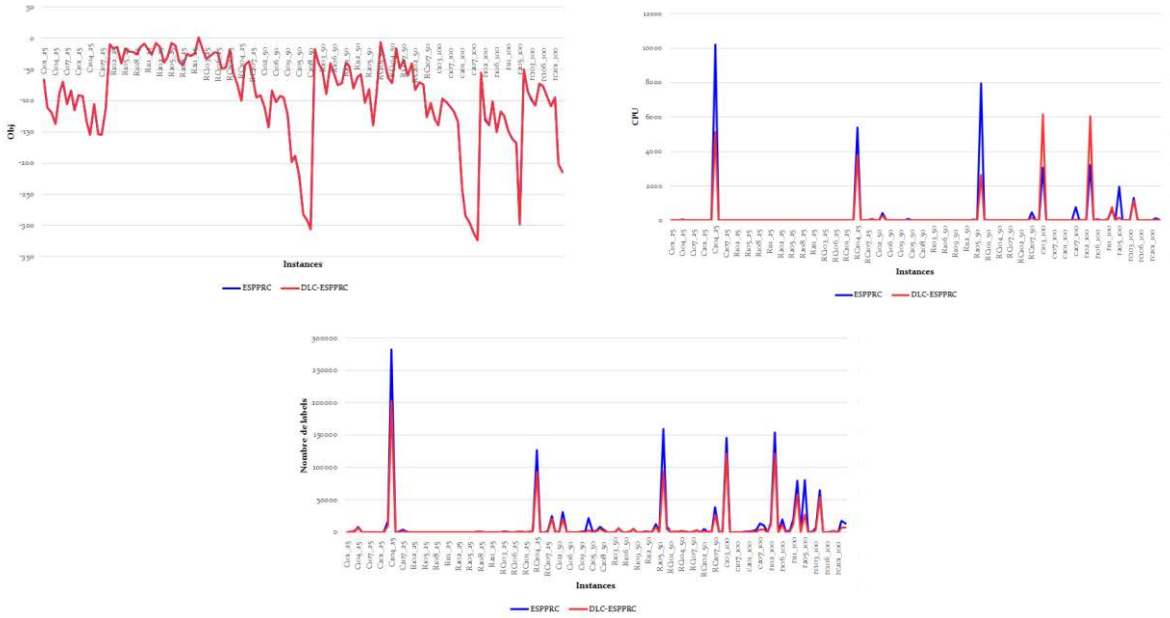

Figure 4.1: Comparaison du ESPPRC et le DLC-ESPPRC

Comme indiqué dans les tableaux Tableau 4-2, Tableau 4-3 et Tableau 4-4 , et dans la figure 4.1 notre algorithme DLC-ESPPRC approché a été plus rapide que le ESPPRC exact (2 à 3 fois) tout en obtenant une solution exacte. Néanmoins, nous avons noté 6 cas seulement (sur 133 cas) pour lesquels ESPPRC était plus rapide, et 3 autres cas avec un écart non nul mais négligé (moins de 0,02%) entre les résultats des deux algorithmes. Nous avons mis en évidence ces valeurs en caractères gras. De plus, le nombre d'étiquettes a été réduit en utilisant le DLC-ESPPRC.

4.5.2 Génération de colonnes avec relaxation lagrangienne du ESPPRC

Dans cette section, nous évaluons l'efficacité de notre algorithme intégré dans une procédure de génération de colonnes pour résoudre le VRPTW. Même ainsi, nous obtenons directement des solutions entières qui sont écrites en gras dans les tableaux de résultats. Les programmes linéaires pour les problèmes maîtres restreints sont résolus avec ILOG CPLEX 20.1.

Sur les tableaux Tableau 4-5, Tableau 4-6 et Tableau 4-7, nous avons reporté le nombre d'itérations (N_i), la borne inférieure (L_i^b) et la borne supérieure (U_i) de l'objectif, le temps de calcul en secondes (T_i), le nombre de colonnes générées (C_i), où $i = 1$ pour ESPPRC et $i = 2$ pour DLC-ESPPRC. Pour obtenir la

borne supérieure, nous avons utilisé la méthode branch-and-bound. Néanmoins, la comparaison des deux algorithmes est réalisée en utilisant leurs bornes inférieures calculées.

Dans la plupart des cas, nous avons obtenu les solutions optimales dans des délais raisonnables. Ces solutions entières obtenues sont mises en évidence en gras dans les tableaux. Comme dans [129], le processus de génération de colonne est lancé avec une adaptation de l'algorithme de Clarke et Wright [28]. Le sous-problème est stoppé à chaque itération lorsque 500 étiquettes ont été étendues au dépôt avec un coût négatif.

Tableau 4-5: Résolution du VRPTW (Instances de Solomon avec 25 clients) par DLC-ESPPRC (4.3) et ESPPRC Classique [129].

	ESPPRC					DLC-ESPPRC					Comparison
Instance	L_1^o	T_1	N_1	C_1	U_1	L_2^o	T_2	N_2	C_2	U_2	T_1/T_2
c101	**191.3**	0.1511	33	579		**191.3**	0.1444	58	407		1.05
c102	**190.3**	0.6057	42	2427		**190.3**	0.5976	40	1415		1.01
c103	**190.3**	4.4735	46	4730		**190.3**	1.6253	32	2293		2.75
c104	**186.9**	14.299	46	5603		**186.9**	11.819	33	4126		1.21
c105	**191.3**	0.0762	41	716		**191.3**	0.0663	62	478		1.15
c106	**191.3**	0.1523	28	596		**191.3**	0.1371	52	348		1.11
c107	**191.3**	0.1838	28	707		**191.3**	0.1790	54	500		1.03
c108	**191.3**	0.4445	57	2230		**191.3**	0.3077	58	840		1.44
c109	**191.3**	0.4289	52	2489		**191.3**	0.3796	64	1712		1.13
c201	**214.7**	0.2284	89	2240		**214.7**	0.2075	237	1019		1.10
c202	**214.7**	1.2143	59	5321		**214.7**	0.5300	69	2288		2.29
c203	**214.7**	21.112	56	10477		**214.7**	7.8315	53	7581		2.70
c204	**213.1**	387.31	61	14633		**213.1**	214.68	64	12641		1.80
c205	**214.7**	0.5769	68	4229		**214.7**	0.2321	125	1426		2.49
c206	**214.7**	0.8878	76	5888		**214.7**	0.5638	181	3039		1.57
c207	**214.5**	2.7248	57	8564		**214.5**	1.8335	60	8398		1.49
c208	**214.5**	1.1114	68	6608		**214.5**	0.9782	126	4583		1.14
r101	**617.1**	0.0060	10	97		**617.1**	0.0056	13	69		1.07
r102	546.3	0.1276	14	443	547.8	546.3	0.1089	18	386	547.8	1.17
r103	**454.6**	0.0664	16	588		**454.6**	0.0631	20	475		1.05
r104	**416.9**	0.3680	17	968		**416.9**	0.3566	21	993		1.03
r105	**530.5**	0.0148	14	268		**530.5**	0.0147	23	152		1.01
r106	457.3	0.0474	15	579	482.3	457.3	0.0398	19	458	469	1.19
r107	**424.3**	0.1217	19	964		**424.3**	0.1058	22	730		1.15
r108	396.8	0.4917	20	1496	**397.3**	396.8	0.4755	24	1276	**397.3**	1.03
r109	**441.3**	0.0449	26	466		**441.3**	0.0352	31	330		1.28
r110	438.4	0.0836	19	704	**444.1**	438.4	0.0623	23	486	444.7	1.34
r111	427.3	0.1876	22	810	430.1	427.3	0.1570	21	577	430.8	1.19
r112	387.1	0.4393	22	1263	394	387.1	0.3932	24	1055	406.5	1.12
r201	460.1	0.1253	22	825	467.4	460.1	0.1092	47	516	**463.3**	1.15
r202	**410.5**	0.2018	19	1391		**410.5**	0.1939	24	1502		1.04
r203	**391.4**	0.4606	17	2187		**391.4**	0.3941	19	2107		1.17
r204	350.5	6.4292	26	4579	359.1	350.5	5.3685	29	4674	359.1	1.20
r205	390.6	0.2780	35	2154	395.8	390.6	0.2457	63	1642	395.8	1.13
r206	373.6	0.5028	25	2764	**374.4**	373.6	0.4811	29	2527	374.9	1.05
r207	360.1	1.7059	22	3760	**361.6**	360.1	1.2286	23	3290	371.3	1.39
r208	**328.2**	25.401	29	5452		**328.2**	23.231	38	5438	0.00	1.09
r209	364.1	0.4151	23	2394	376.4	364.1	0.3461	39	1934	383.6	1.20
r210	404.2	0.5822	36	2714	**404.6**	404.2	0.4839	40	2182	**404.6**	1.20
r211	341.3	2.4530	26	3979	352.7	341.3	2.1140	30	3924	354.6	1.16
rc101	406.6	0.1178	18	380	475.8	406.6	0.0998	28	261	462.5	1.18
rc102	**351.8**	0.0996	24	662		**351.8**	0.0479	23	368		2.08
rc103	**332.8**	0.1460	22	858		**332.8**	0.1304	27	566		1.12
rc104	**306.6**	0.9123	25	1386		**306.6**	0.8882	28	879		1.03
rc105	**411.3**	0.0495	16	595		**411.3**	0.0487	31	425		1.02
rc106	**345.5**	0.0662	25	587		**345.5**	0.0597	30	374		1.11
rc107	**298.3**	0.1960	29	797		**298.3**	0.1800	29	672		1.09
rc108	**294.5**	0.5009	33	972		**294.5**	0.4740	22	875		1.06
rc201	**360.2**	0.0923	27	895		**360.2**	0.069	54	456		1.34
rc202	**338**	0.2939	26	1555		**338**	0.2632	33	1475		1.12
rc203	**326.9**	0.7161	25	2732		**326.9**	0.6459	28	2295		1.11
rc204	**299.7**	36.832	42	9243		**299.7**	18.884	39	6640		1.95
rc205	**338**	0.3323	25	1788		**338**	0.2578	59	1041		1.29
rc206	**324**	0.3496	38	1894		**324**	0.1880	53	951		1.86
rc207	**298.3**	0.6462	27	1639		**298.3**	0.4816	44	1474		1.34
rc208	**269.1**	392.34	44	9204		**269.1**	332.65	47	8569		1.18

Tableau 4-6: Résolution du VRPTW (Instances de Solomon avec 50 clients) par DLC-ESPPRC (4.3) et ESPPRC Classique [129].

	ESPPRC					DLC-ESPPRC					Comparison
Instance	L_1^b	T_1	N_1	C_1	U_1	L_2^b	T_2	N_2	C_2	U_2	T_1/T_2
c101	**362.4**	1.0115	75	1856		**362.4**	0.7978	109	953		1.27
c102	**361.4**	3.8341	68	4035		**361.4**	2.7340	78	3735		1.40
c103	**361.4**	40.836	64	7112		**361.4**	39.489	70	5652		1.03
c104											
c105	**362.4**	0.7816	81	2301		**362.4**	0.5786	107	1129		1.35
c106	**362.4**	0.6941	71	2022		**362.4**	0.6258	138	1211		1.11
c107	**362.4**	0.8750	70	2754		**362.4**	0.8505	130	2037		1.03
c108	**362.4**	1.9062	77	3954		**362.4**	1.4778	108	2731		1.29
c109	**362.4**	5.3634	95	6691		**362.4**	3.6194	96	4610		1.48
c201	**360.2**	6.2337	142	11729		**360.2**	3.1029	452	3301		2.01
c202	**360.2**	227.03	239	44378		**360.2**	20.298	150	17332		11.18
c203						**359.8**	45441	215	63825		0.00
c204											
c205	**359.8**	23.848	227	24835		**359.8**	4.6578	237	4962		5.12
c206	**359.8**	39.175	232	31193		**359.8**	7.0369	169	7645		5.57
c207	**359.6**	357.29	235	51066		**359.6**	72.0327	209	32006		4.96
c208	350.5	91.240	230	47308		350.5	16.726	186	16042		5.45
r101	1043.	0.2037	24	455	**1044**	1043.	0.1945	45	321	**1044**	1.05
r102	**909**	0.3125	24	1303		**909**	0.2761	30	939		1.13
r103	769.2	1.7881	31	1991	775.7	769.2	1.5898	35	1904	**772.9**	1.12
r104	619.1	18.052	44	5094	632.1	619.1	21.587	58	5198	632.6	0.84
r105	892.1	0.1609	24	875	908.1	892.1	0.1565	38	590	907.1	1.03
r106	791.4	0.6901	30	2017	797.3	791.4	0.5481	34	1392	797.3	1.26
r107	707.3	2.4484	23	4251	714.4	707.3	2.3982	37	2912	711.3	1.02
r108	594.7	33.785	43	5951	627.6	594.7	41.571	52	5714	630.6	0.81
r109	775.3	0.4373	30	1526	790.7	775.3	0.3899	41	936	802.5	1.12
r110	695.1	1.3995	32	3114	710.6	695.1	1.0832	41	1754	716.8	1.29
r111	696.3	2.4375	37	2771	713.9	696.3	1.7339	40	2419	720	1.41
r112	614.9	7.1667	41	3880	648	614.9	7.4684	46	3118	655.4	0.96
r201	**791.9**	1.5324	33	5151		**791.9**	1.2532	108	2685		1.22
r202	**698.5**	3.5430	37	5868		**698.5**	3.1496	52	4664		1.12
r203	598.6	58.661	48	11336	606.6	598.6	43.665	51	10801	606.6	1.34
r204											
r205	682.9	2.9196	45	5133	711.5	682.9	2.8777	69	4343	712.6	1.01
r206	626.3	18.991	46	9127	641.4	626.3	13.571	52	6926	644.9	1.40
r207	564.1	478.56	56	16097	596.6	564.1	463.02	61	16036	580.5	1.03
r208											
r209	599.8	19.011	46	7418	**600.6**	599.8	18.390	73	7207	**600.6**	1.03
r210	636.1	18.716	46	8166	667.3	636.1	18.012	52	7661	672.4	1.04
r211	528.6	759.98	51	12915	567.1	528.6	435.73	51	11292	562.3	1.74
rc101	850.0	0.3473	26	955	971.1	850.0	0.2145	45	648	986.6	1.62
rc102	721.8	0.8344	32	1829	841	721.8	0.6850	36	1340	850.3	1.22
rc103	645.3	2.8273	39	2460	734.7	645.3	2.7074	39	1973	734.7	1.04
rc104	**545.8**	15.396	55	4094		**545.8**	9.7646	43	2898		1.58
rc105	761.6	0.5466	28	1653	886.2	761.6	0.4983	33	1092	880.8	1.10
rc106	664.4	1.3005	42	1787	**723.2**	664.4	1.1901	46	1160	**723.2**	1.09
rc107	603.6	5.2944	51	2788	668.1	603.6	4.9380	60	2425	668.1	1.07
rc108	541.2	26.166	46	3196	**598.1**	541.2	24.343	49	3488	**598.1**	1.07
rc201	**684.8**	1.2478	77	2818		**684.8**	1.2256	109	1828		1.02
rc202	**613.6**	4.7339	51	5414		**613.6**	3.1968	58	4176		1.48
rc203	**555.3**	400.82	114	23424		**555.3**	253.66	104	19366		1.58
rc204											
rc205	**630.2**	6.0739	39	9400		**630.2**	5.2052	69	6129		1.17
rc206	**610**	3.4529	58	5008		**610**	3.3737	81	4866		1.02
rc207	**558.6**	88.287	77	15018		**558.6**	33.139	87	12967		2.66
rc208											

Tableau 4-7: Résolution du VRPTW (Instances de Solomon avec 100 clients) par DLC-ESPPRC (4.3) et ESPPRC Classique [129].

	ESPPRC					DLC-ESPPRC					Comparison
Instance	L_1^b	T_1	N_1	C_1	U_1	L_2^b	T_2	N_2	C_2	U_2	T_1/T_2
c101	**827.3**	6.4666	131	3855		**827.3**	9.2139	343	2866		0.70
c102	**827.3**	87.821	131	11646		**827.3**	129.60	182	10903		0.68
c103	**826.3**	811.77	154	20127		844.755	665.54	121	18261	872.4	1.22
c104											
c105	**827.3**	8.7654	124	4379		**827.3**	10.373	273	3759		0.85
c106	**827.3**	16.510	148	6511		**827.3**	16.241	240	4788		1.02
c107	**827.3**	12.150	130	5047		828	10.853	196	4100		1.12
c108	**827.3**	26.256	164	7646		831.9	19.673	166	5495		1.33
c109	**827.3**	84.380	205	14353		833.3	67.830	206	9990		1.24
c201	**589.1**	111.32	429	33294		**589.1**	28.080	442	6310		3.96
c202	**589.1**	102694	549	157749		**589.1**	9907.8	500	101473		10.36
c203											
c204											
c205	**586.4**	625.07	794	91715		**586.4**	73.781	482	13338		8.47
c206	**586**	2721.9	1449	221620		**586**	147.00	443	24023		18.52
c207											
c208	**585.8**	5862.0	1657	288142		**585.8**	571.90	677	53597		10.25
r101	1631.15	1.4273	37	1738	1649.7	1631.15	1.1753	63	902	1643.3	1.21
r102	**1466.6**	5.6666	46	3868		**1466.6**	4.9803	55	3017		1.14
r103	1206.8	38.986	70	7347	1220.8	1206.81	52.839	92	7170	1225	0.74
r104	956.973	2145.7	108	15430	1004.7	956.985	3537.1	151	16547	1011.8	0.61
r105	1346.142	4.1300	56	3540	1359.2	1346.144	3.1770	83	2018	1371.5	1.30
r106	1227.112	25.533	79	6850	1250.4	1227.114	25.225	83	5405	1248.6	1.01
r107	1053.596	281.24	100	11407	1082	1053.634	427.44	111	10141	1081.5	0.66
r108	913.529	6599.9	137	20892	963.8	913.532	9746.6	165	22109	977.9	0.68
r109	1134.28	14.494	72	5566	1158	1134.28	10.492	79	3521	1170	1.38
r110	1055.566	74.189	92	8613	1086.9	1055.567	46.017	92	6051	1091.6	1.61
r111	1034.734	105.44	101	11036	1072.2	1034.845	101.56	104	9893	1089.5	1.04
r112	926.753	1982.1	107	13507	974.4	926.836	2329.6	116	11070	973.2	0.85
r201	1140.3	75.666	72	37246	1163.6	1140.3	51.418	230	14094	1195.3	1.47
r202	1022.261	455.13	152	29971	1037.4	1022.236	305.46	166	25394	1038.2	1.49
r203											
r204											
r205	939.243	306.94	156	23160	986.4	939.224	241.10	199	16769	997.8	1.27
r206	866.868	45184	193	56945	905	866.869	42641	203	52493	909.1	1.06
r207											
r208											
r209	841.423	7535.5	152	37355	894.4	841.409	4354.0	176	29744	903.4	1.73
r210	889.37	24625	157	39935	934.1	889.38	23990	157	33884	938.1	1.03
r211											
rc101	1584.094	2.6982	48	3063	1668.7	1584.094	2.1404	63	1660	1679.2	1.26
rc102	1406.262	12.730	65	5402	1505.1	1408.19	10.801	63	4196	1519.1	1.18
rc103	1225.544	110.08	99	9696	1307.9	1225.661	104.41	107	7553	1363.6	1.05
rc104	1101.808	2297.6	116	13395	1176.2	1101.886	3998.2	123	12205	1183.6	0.57
rc105	1471.925	7.8960	54	4299	1555.9	1471.925	7.3293	76	3348	1571.4	1.08
rc106	1318.796	12.726	72	5250	1419.4	1318.796	11.381	72	3630	1437.9	1.12
rc107	1183.37	66.542	88	8140	1259.6	1185.676	63.438	85	6597	1266.1	1.05
rc108	1073.436	720.62	110	10782	1162.2	1084.76	571.78	94	9773	1201.1	1.26
rc201	1255.942	32.889	147	10050	1271.2	1255.942	27.369	224	7036	1282.1	1.20
rc202	1088.085	241.90	165	27259	1102.6	1088.085	133.41	162	18795	1100.2	1.81
rc203											
rc204											
rc205	1147.608	159.23	150	15025	1175.1	1147.608	121.93	214	14318	1184.8	1.31
rc206	1038.6	427.65	205	26875	1072.9	1038.6	273.89	213	18343	1071.9	1.56
rc207	947.314	7779.9	199	45428	1003.4	947.317	5374.9	203	39307	1016	1.45
rc208											

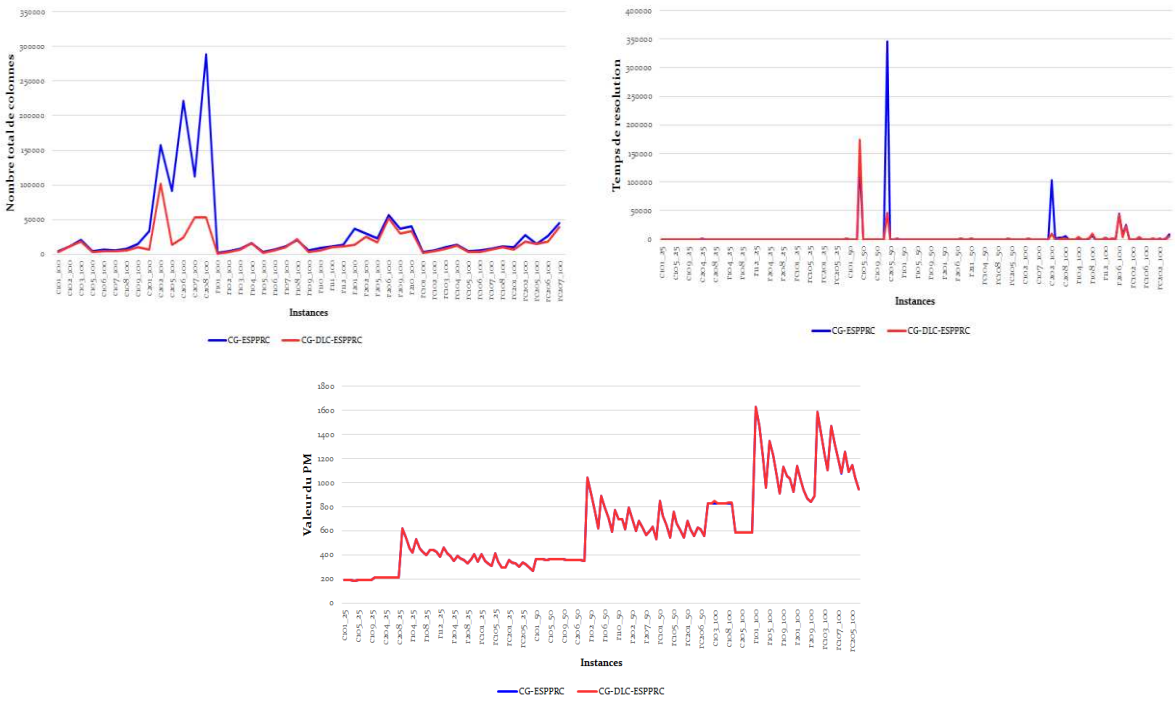

Figure 4.2: Impact de la DLC-ESPPRC sur la qualité des solutions

À partir des tableaux Tableau 4-5, Tableau 4-6 et Tableau 4-7, et la figure 4.2 nous pouvons tirer les conclusions suivantes. Sur un nombre total de 168 cas étudiés, notre algorithme a été deux fois plus rapide dans 124 cas, soit 74% des cas, tout en obtenant des résultats identiques. Nous avons obtenu jusqu'à 18% de gain en temps de calcul. Dans 27 cas (15 %), nous avons obtenu des résultats de précision et de temps de calcul similaires. Dans un cas, l'algorithme de Feillet [129] n'a pas pu obtenir de solution en un temps raisonnable (moins de 36 heures) alors que notre algorithme l'a fait en 12 heures. Dans les 16 cas restants, les deux algorithmes n'ont pas pu obtenir de résultats dans un délai raisonnable.

4.6 Conclusion

Ce chapitre propose un algorithme de résolution approchée pour résoudre le ESPPRC pour les graphes acycliques et cycliques qui repose sur une relaxation lagrangienne. La dominance s'applique uniquement à un sous-ensemble des ressources. Par rapport à l'algorithme exact, notre approche offre un bon compromis entre précision et temps de calcul. À l'avenir, nous prévoyons d'appliquer notre algorithme pour résoudre divers problèmes tels que branch-and-price, branch-and-cut et branch-and-price-and-cut.

CONCLUSION GENERALE

Dans cette thèse consacrée à la résolution du VRPs, comme ces problèmes sont complexes et de grande taille, ce qui les place hors des capacités de résolution des logiciels disponibles aujourd'hui, pour pouvoir les traiter, des méthodes de décomposition de l'espace des solutions sont utilisées. Ces modèles peuvent être résolus par une technique basée sur la méthode de génération de colonnes dont le point fort est de pouvoir caractériser une solution optimale d'un problème en manipulant un ensemble réduit de ses variables. Cette méthode a prouvé son efficacité pour la résolution de divers problèmes d'optimisation combinatoire, cependant, elle est connue pour ses problèmes de convergence.

Nous avons principalement développé les approches de génération de colonnes et de décomposition en problème maître et sous-problème. La difficulté de la résolution du sous-problème étant directement liée au nombre de ressources, nous avons particulièrement étudié les techniques de réduction de l'espace des ressources, cette notion de réduction étant un élément-clé de l'efficacité de la résolution globale du problème. Les tests effectués sur plusieurs instances ont montré l'efficacité de DLC-SPPRC et DLC-ESPPRC.

Nos perspectives de recherche sur ce problème sont nombreuses. Parmi elles, nous prévoyons d'appliquer notre algorithme pour résoudre divers problèmes tels que branch-and-price, branch-and-cut et branch-and-price-and-cut. De plus, nous explorerons des algorithmes d'optimisation plus avancés (heuristiques, méta-heuristiques, etc.) qui ont été appliqués avec succès dans d'autres domaines, tels que l'apprentissage en ligne, la planification, l'optimisation multi-objectifs, le transport, la médecine et la classification des données., aussi les problèmes de reconstruction d'une solution robuste suite à la perturbation, par un quelconque aléa, du planning initialement construit. Ces problématiques de ré-optimisation suscitent un intérêt croissant chez les ingénieurs chargés de la planification dans les grandes entreprises de transport et ouvrent des voies de recherche particulièrement intéressantes et prometteuses.

Bibliographie

[1] G. B. Dantzig et J. H. Ramser, «The truck dispatching problem,» *Management science,* vol. 6, p. 80–91, 1959.

[2] M. Haj Rachid, «Les problèmes de tournées de véhicules en planification industrielle : classification et comparaison d'opérateurs évolutionnaires,» 2010.

[3] C. H. Cheng, Y. P. Gupta, W. H. Lee et K. F. Wong, «A TSP-based heuristic for forming machine groups and part families,» *International Journal of Production Research,* vol. 36, p. 1325–1337, 1998.

[4] R. Kumar et Z. Luo, «Optimizing the operation sequence of a chip placement machine using TSP model,» *IEEE Transactions on Electronics Packaging Manufacturing,* vol. 26, p. 14–21, 2003.

[5] M. Ayob et G. Kendall, «The optimisation of the single surface mount device placement machine in printed circuit board assembly: a survey,» *International Journal of Systems Science,* vol. 40, p. 553–569, 2009.

[6] B. a. C. D. a. D. J. R. a. V. O. D. Verlinden, «Modeling Sheet Metal Integrated Production Planning for Laser Cutting and Air Bending,» vol. 344, pp. 913--920, 2007.

[7] J.-F. Cordeau, G. Laporte, M. W. P. Savelsbergh et D. Vigo, «Vehicle routing,» *Handbooks in operations research and management science,* vol. 14, p. 367–428, 2007.

[8] C. H. Christiansen, «Elements of Vehicle Routing under uncertainty,» *Aarhus School of Business, Department of Business Studies,* 2007.

[9] A.-l. Chen, G.-k. Yang et Z.-m. Wu, «Hybrid discrete particle swarm optimization algorithm for capacitated vehicle routing problem,» *Journal of Zhejiang University-Science A,* vol. 7, p. 607–614, 2006.

[10] S.-Y. Tan et W.-C. Yeh, «The Vehicle Routing Problem: State-of-the-Art Classification and Review,» *Applied Sciences,* vol. 11, p. 10295, 2021.

[11] G. Dantzig, R. Fulkerson et S. Johnson, «Solution of a large-scale traveling-salesman problem,» *Journal of the operations research society of America,* vol. 2, p. 393–410, 1954.

[12] T. K. Ralphs, «Parallel branch and cut for capacitated vehicle routing,» *Parallel Computing,* vol. 29, p. 607–629, 2003.

[13] K. Q. Zhu, «Heuristic Methods For Vehicle Routing Problem with Time Windows,» 1999.

[14] G. Laporte, «The vehicle routing problem: An overview of exact and approximate algorithms,»

European journal of operational research, vol. 59, p. 345–358, 1992.

[15] M. Fisher, «Vehicle routing,» *Handbooks in operations research and management science,* vol. 8, p. 1–33, 1995.

[16] G. Laporte et I. H. Osman, «Routing problems: A bibliography,» *Annals of operations research,* vol. 61, p. 227–262, 1995.

[17] M. Gendreau, «Vehicle routing: modern heuristics,» *Local search in combinatorial optimization,* 1997.

[18] G. Laporte, M. Gendreau, J.-Y. Potvin et F. Semet, «Classical and modern heuristics for the vehicle routing problem,» *International transactions in operational research,* vol. 7, p. 285–300, 2000.

[19] M. Gendreau, G. Laporte et J.-Y. Potvin, «Metaheuristics for the Capacitated VRP,» chez *The Vehicle Routing Problem*, 2002, pp. 129-154.

[20] P. Toth et D. Vigo, Vehicle routing: problems, methods, and applications, SIAM, 2014.

[21] O. Bräysy, W. Dullaert et M. Gendreau, «Evolutionary algorithms for the vehicle routing problem with time windows,» *Journal of Heuristics,* vol. 10, p. 587–611, 2004.

[22] O. Bräysy et M. Gendreau, «Vehicle routing problem with time windows, Part II: Metaheuristics,» *Transportation science,* vol. 39, p. 119–139, 2005.

[23] M. Gendreau, J.-Y. Potvin, O. Bräumlaysy, G. Hasle et A. Løkketangen, «Metaheuristics for the Vehicle Routing Problem and Its Extensions: A Categorized Bibliography,» chez *The Vehicle Routing Problem: Latest Advances and New Challenges*, B. Golden, S. Raghavan et E. Wasil, Éds., Boston, MA: Springer US, 2008, p. 143–169.

[24] J.-Y. Potvin, «State-of-the art review—Evolutionary algorithms for vehicle routing,» *INFORMS Journal on computing,* vol. 21, p. 518–548, 2009.

[25] A. H. L. a. A. G. Doig, «An Automatic Method of Solving Discrete Programming Problems,» *Econometrica,* vol. 28, n° %13, pp. 497--520, 1960.

[26] M. Padberg et G. Rinaldi, «Optimization of a 532-city symmetric traveling salesman problem by branch and cut,» *Operations research letters,* vol. 6, p. 1–7, 1987.

[27] C. Rego et C. Roucairol, «Le problème de tournées de vehicules: Étude et résolution approchée,» 1994.

[28] G. Clarke et J. W. Wright, «Scheduling of vehicles from a central depot to a number of delivery points,» *Operations research,* vol. 12, p. 568–581, 1964.

[29] M. M. Solomon, «Algorithms for the vehicle routing and scheduling problems with time window constraints,» *Operations research,* vol. 35, p. 254–265, 1987.

[30] J. D. a. A. P. P. Siarry, Métaheuristiques pour l'optimisation difficile, EYROLLES, Éd., 2003, p. 355.

[31] J. E. Beasley, «Route first—cluster second methods for vehicle routing,» *Omega,* vol. 11, p. 403–408, 1983.

[32] B. L. Golden, A. Assad, L. Levy et F. Gheysens, «The fleet size and mix routing problem,» *Management Science and Statistics. Working paper, University of Maryland at college park,* 1982.

[33] A. Le Bouthillier, Modélisation UML pour une architecture coopérative appliquée au problème de tournées de véhicules avec fenêtres de temps, Université de Montréal, Centre de recherche sur les transports, 2000.

[34] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller et E. Teller, «Equation of state calculations by fast computing machines,» *The journal of chemical physics,* vol. 21, p. 1087–1092, 1953.

[35] K. C. Tan, L. H. Lee, Q. L. Zhu et K. Ou, «Heuristic methods for vehicle routing problem with time windows,» *Artificial intelligence in Engineering,* vol. 15, p. 281–295, 2001.

[36] R. Bent et P. Van Hentenryck, «A two-stage hybrid local search for the vehicle routing problem with time windows,» *Transportation Science,* vol. 38, p. 515–530, 2004.

[37] F. Glover, «Future paths for integer programming and links to artificial intelligence,» *Computers & operations research,* vol. 13, p. 533–549, 1986.

[38] É. Taillard, «Parallel iterative search methods for vehicle routing problems,» *Networks,* vol. 23, p. 661–673, 1993.

[39] J. Xu et J. P. Kelly, «A network flow-based tabu search heuristic for the vehicle routing problem,» *Transportation science,* vol. 30, p. 379–393, 1996.

[40] C. Rego et C. Roucairol, «A parallel tabu search algorithm using ejection chains for the vehicle routing problem,» chez *Meta-Heuristics*, Springer, 1996, p. 661–675.

[41] É. Taillard, P. Badeau, M. Gendreau, F. Guertin et J.-Y. Potvin, «A tabu search heuristic for the vehicle routing problem with soft time windows,» *Transportation science,* vol. 31, p. 170–186, 1997.

[42] P. Hansen et N. Mladenović, «An introduction to variable neighborhood search,» chez *Meta-heuristics*, Springer, 1999, p. 433–458.

[43] K. Fleszar, I. H. Osman et K. S. Hindi, «A variable neighbourhood search algorithm for the open vehicle routing problem,» *European Journal of Operational Research,* vol. 195, p. 803–809, 2009.

[44] L. S. Pitsoulis et M. G. C. Resende, «Greedy randomized adaptive search procedures,» *Handbook of applied optimization,* p. 168–183, 2002.

[45] M. G. C. Resende et C. C. Ribeiro, «Greedy randomized adaptive search procedures: Advances and applications,» *Handbook of metaheuristics,* vol. 146, p. 281–317, 2010.

[46] D. B. Fogel, «An introduction to simulated evolutionary optimization,» *IEEE transactions on neural networks,* vol. 5, p. 3–14, 1994.

[47] W. M. Spears, K. A. De Jong, T. Bäck, D. B. Fogel et H. De Garis, «An overview of evolutionary computation,» chez *European conference on machine learning*, 1993.

[48] Z. Michalewicz et M. Michalewicz, «Evolutionary computation techniques and their applications,» chez *1997 IEEE International Conference on Intelligent Processing Systems (Cat. No. 97TH8335)*, 1997.

[49] P. Calégari, G. Coray, A. Hertz, D. Kobler et P. Kuonen, «A taxonomy of evolutionary algorithms in combinatorial optimization,» *Journal of Heuristics,* vol. 5, p. 145–158, 1999.

[50] J.-L. Deneubourg, S. Aron, S. Goss et J. M. Pasteels, «The self-organizing exploratory pattern of the argentine ant,» *Journal of insect behavior,* vol. 3, p. 159–168, 1990.

[51] M. Dorigo, «Optimization, learning and natural algorithms,» *Ph. D. Thesis, Politecnico di Milano,* 1992.

[52] M. Dorigo, V. Maniezzo et A. Colorni, «Ant system: optimization by a colony of cooperating agents,» *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics),* vol. 26, p. 29–41, 1996.

[53] F. Glover, «Heuristics for integer programming using surrogate constraints,» *Decision sciences,* vol. 8, p. 156–166, 1977.

[54] M. G. A. Verhoeven et E. H. L. Aarts, «Parallel local search,» *Journal of heuristics,* vol. 1, p. 43–65, 1995.

[55] E. G. Talbi, Z. Hafidi et J. M. Geib, «Parallel adaptive tabu search for large optimization problems,» 1997.

[56] P. Jog et D. Van Gucht, «Parallelisation of probabilistic sequential search algorithms,» chez *Genetic algorithms and their applications*, 2013.

[57] H. Mühlenbein, M. Gorges-Schleuter et O. Krämer, «Evolution algorithms in combinatorial optimization,» *Parallel computing,* vol. 7, p. 65–85, 1988.

[58] P. Jog, J. Y. Suh et D. Van Gucht, «Parallel genetic algorithms applied to the traveling salesman problem,» *SIAM Journal on Optimization,* vol. 1, p. 515–529, 1991.

[59] N. L. J. Ulder, E. H. L. Aarts, H.-J. Bandelt, P. J. M. Van Laarhoven et E. Pesch, «Genetic local search algorithms for the traveling salesman problem,» chez *International Conference on Parallel Problem Solving from Nature*, 1991.

[60] B. Freisleben et P. Merz, «New genetic local search operators for the traveling salesman problem,» chez *International Conference on Parallel Problem Solving from Nature*, 1996.

[61] C. Prins, «A simple and effective evolutionary algorithm for the vehicle routing problem,» *Computers & operations research,* vol. 31, p. 1985–2002, 2004.

[62] C. Erbao et L. Mingyong, «A hybrid differential evolution algorithm to vehicle routing problem with fuzzy demands,» *Journal of computational and applied mathematics,* vol. 231, p. 302–310, 2009.

[63] T. Zhen et Q. Zhang, «A hybrid metaheuristic algorithm for the multi-depot vehicle routing problem with time windows,» chez *2009 International Conference on Networks Security, Wireless Communications and Trusted Computing*, 2009.

[64] F. Glover, A. Løkketangen et D. L. Woodruff, «Scatter search to generate diverse MIP solutions,» chez *Computing Tools for Modeling, Optimization and Simulation*, Springer, 2000, p. 299–317.

[65] M. Laguna, *Scatter Search In Handbook of Applied Optimization, PM Pardalos and M. GC Resende,* Oxford University Press, 2002.

[66] A. Wade et S. Salhi, «An ant system algorithm for the mixed vehicle routing problem with backhauls,» chez *Metaheuristics: computer decision-making*, Kluwer Academic Publishers, 2004, p. 699–719.

[67] Y. Gajpal et P. Abad, «An ant colony system (ACS) for vehicle routing problem with simultaneous delivery and pickup,» *Computers & Operations Research,* vol. 36, p. 3215–3223, 2009.

[68] J. Yang et Y. Zhuang, «An improved ant colony optimization algorithm for solving a complex combinatorial optimization problem,» *Applied Soft Computing,* vol. 10, p. 653–660, 2010.

[69] A. E. Rizzoli, R. Montemanni, E. Lucibello et L. M. Gambardella, «Ant colony optimization for real-world vehicle routing problems,» *Swarm Intelligence,* vol. 1, p. 135–151, 2007.

[70] Y. A. N. G. Yu, H. Zhu, J. M. Frantz, M. E. Reding, K. C. Chan et H. E. Ozkan, «Evaporation and coverage area of pesticide droplets on hairy and waxy leaves,» *biosystems engineering,* vol. 104, p. 324–334, 2009.

[71] Y. Zhang, T.-T. Tang, C. Girit, Z. Hao, M. C. Martin, A. Zettl, M. F. Crommie, Y. R. Shen et F. Wang, «Direct observation of a widely tunable bandgap in bilayer graphene,» *Nature,* vol. 459, p. 820–823, 2009.

[72] J. W. Tang, F. Y. L. Lai, P. Nymadawa, Y.-M. Deng, M. Ratnamohan, M. Petric, T. P. Loh, N. W. S. Tee, D. E. Dwyer, I. G. Barr et others, «Comparison of the incidence of influenza in relation to climate factors during 2000–2007 in five countries,» *Journal of medical virology,* vol. 82, p. 1958–1965, 2010.

[73] V. Klee et G. J. Minty, «How good is the simplex algorithm,» *Inequalities,* vol. 3, p. 159–175, 1972.

[74] L. G. Khachiyan, «A polynomial algorithm in linear programming,» chez *Doklady Akademii Nauk*, 1979.

[75] N. Karmarkar, «A new polynomial-time algorithm for linear programming,» chez *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, 1984.

[76] M. Lebbar, «Résolution de problèmes combinatoires dans l'industrie: apport de la programmation mathématique et des techniques de décomposition,» 2000.

[77] V.Chvatal, Linear Programming, New York: W. H. Freeman and Company, 1983.

[78] E. L. Lawler et D. E. Wood, «Branch-and-bound methods: A survey,» *Operations research,* vol. 14, p. 699–719, 1966.

[79] H. Marchand, A. Martin, R. Weismantel et L. Wolsey, «Cutting planes in integer and mixed integer programming,» *Discrete Applied Mathematics,* vol. 123, p. 397–446, 2002.

[80] G. B. Dantzig, «Programming in a linear structure,» 1948.

[81] G. Dantzig, Linear Programming and Extensions, Princeton: Princeton University Press, 1963.

[82] G. B. Dantzig et R. M. Van Slyke, «Generalized upper bounding techniques,» *Journal of Computer and System Sciences,* vol. 1, p. 213–226, 1967.

[83] M. L. Fisher, «The Lagrangian relaxation method for solving integer programming problems,» *Management science,* vol. 27, p. 1–18, 1981.

[84] F. Glover, «A multiphase-dual algorithm for the zero-one integer programming problem,» *Operations Research,* vol. 13, p. 879–919, 1965.

[85] H. J. Greenberg et W. P. Pierskalla, «Surrogate mathematical programming,» *Operations Research,* vol. 18, p. 924–939, 1970.

[86] J. F. BENDERS, «Partitioning procedures for solving mixed-variables programming problems.,» *Numerische Mathematik,* vol. 4, pp. 238-252, 1962/63.

[87] G. B. Dantzig et P. Wolfe, «Decomposition principle for linear programs,» *Operations research,* vol. 8, p. 101–111, 1960.

[88] M. Minoux, Programmation mathématique. Théorie et algorithmes, Lavoisier, 2008.

[89] M. Guignard et M. B. Rosenwein, «An application of lagrangean decomposition to the resource-constrained minimum weighted arborescence problem,» *Networks,* vol. 20, p. 345–359, 1990.

[90] M. Guignard et S. Kim, «Lagrangean decomposition: A model yielding stronger Lagrangean bounds,» *Mathematical programming,* vol. 39, p. 215–228, 1987.

[91] H. Reinoso et N. Maculan, «Lagrangean decomposition in integer linear programming: a new scheme,» *INFOR: Information Systems and Operational Research,* vol. 30, p. 1–5, 1992.

[92] B. Chena et M. Guignard, «Polyhedral analysis and decompositions for capacitated plant location-type problems,» *Discrete Applied Mathematics,* vol. 82, p. 79–91, 1998.

[93] L. S. Lasdon, «Optimization theory for large systems,» *M acmillan Company, London, E ngland,* vol. 1, p. 7, 1970.

[94] L. A. Wolsey et G. L. Nemhauser, Integer and combinatorial optimization, vol. 55, John Wiley & Sons, 1999.

[95] M. Florian, G. Bushell, J. Ferland, G. Guerin et L. Nastansky, «The engine scheduling problem in a railway network,» *INFOR: Information Systems and Operational Research,* vol. 14, p. 121–138, 1976.

[96] R. Richardson, «An optimization approach to routing aircraft,» *Transportation Science,* vol. 10, p. 52–71, 1976.

[97] A. M. Geoffrion et G. W. Graves, «Multicommodity distribution system design by Benders decomposition,» *Management science,* vol. 20, p. 822–844, 1974.

[98] M. L. Fisher et R. Jaikumar, A decomposition algorithm for large-scale vehicle routing, Department of Decision Sciences, Wharton School, University of Pennsylvania, 1978.

[99] R. T. Wong, «Accelerating Benders decomposition for network design.,» 1978.

[100] T. K. Ralphs et M. V. Galati, «Decomposition in integer linear programming,» chez *Integer Programming*, CRC Press, 2005, p. 73–126.

[101] G. Belov et G. Scheithauer, «A branch-and-cut-and-price algorithm for one-dimensional stock cutting and two-dimensional two-stage cutting,» *European Journal of Operational Research,* vol. 171, pp. 85-106, 2006.

[102] A. S. da Cunha, A. Lucena, N. Maculan et M. G. C. Resende, «A relax-and-cut algorithm for the prize-collecting Steiner problem in graphs,» *Discrete Applied Mathematics,* vol. 157, pp. 1198-1217, 2009.

[103] C. Lemaréchal, «The omnipresence of Lagrange,» *Annals of Operations Research,* vol. 153, p. 9–27, 2007.

[104] P. C. Gilmore et R. E. Gomory, «A linear programming approach to the cutting-stock problem,» *Operations research,* vol. 9, p. 849–859, 1961.

[105] G. Desaulniers, J. Desrosiers et M. M. Solomon, Column generation, vol. 5, Springer Science & Business Media, 2006.

[106] J. E. Kelley, «The cutting-plane method for solving convex programs,» *Journal of the society for Industrial and Applied Mathematics,* vol. 8, p. 703–712, 1960.

[107] I. Loiseau, J. M. Brito, N. Maculan et M. Passini, *Génération de colonnes en programmation linéaire en nombres entiers, Optimisation Combinatoire 1-Concepts fondamentaux,* Hermes Science Publications, 2005, p. 237–262.

[108] A. Nagih et F. Soumis, «Nodal aggregation of resource constraints in a shortest path problem,» *European Journal of Operational Research,* vol. 172, p. 500–514, 2006.

[109] R. T. Rockafellar, «Monotone operators and the proximal point algorithm,» *SIAM journal on control and optimization,* vol. 14, p. 877–898, 1976.

[110] P. Wolfe, «A method of conjugate subgradients for minimizing nondifferentiable functions,» chez *Nondifferentiable optimization*, Springer, 1975, p. 145–173.

[111] C. Lemaréchal, «An Algorithm for Minimizing Convex Functions.,» chez *IFIP Congress*, 1974.

[112] H. Ben Amor, J. Desrosiers et A. Frangioni, *Stabilization in Column Generation,* GERAD, Montréal QC H3T 2A7, Canada, 2004, pp. 1-31.

[113] S. Kim, K.-N. Chang et J.-Y. Lee, «A descent method with linear programming subproblems for nondifferentiable convex optimization,» *Mathematical programming,* vol. 71, p. 17–28, 1995.

[114] R. E. Marsten, W. W. Hogan et J. W. Blankenship, «The boxstep method for large-scale optimization,» *Operations Research,* vol. 23, p. 389–405, 1975.

[115] O. Du Merle, D. Villeneuve, J. Desrosiers et P. Hansen, «Stabilized column generation,» *Discrete Mathematics,* vol. 194, p. 229–237, 1999.

[116] P. Wentges, «Weighted Dantzig-Wolfe decomposition for linear mixed-integer programming,» *International Transactions in Operational Research,* vol. 4, p. 151–162, 1997.

[117] D. Huisman, R. Jans, M. Peeters et A. P. M. Wagelmans, «Combining column generation and Lagrangian relaxation,» chez *Column generation*, Springer, 2005, p. 247–270.

[118] L. A. N. Lorena, M. A. Pereira et S. N. A. Salomão, «The lagrangean/surrogate relaxation and the column generation: New bounds and new columns,» 2001.

[119] V. V. Rodionov, «The parametric problem of shortest distances,» *USSR Computational Mathematics and Mathematical Physics,* vol. 8, p. 336–343, 1968.

[120] J. Murchland, «The effect of increasing or decreasing the length of a single arc on all shortest distances in a graph,» 1967.

[121] R. Dionne, «Etude et extension d'un algorithme de Murghland,» *INFOR: Information Systems and Operational Research,* vol. 16, p. 132–146, 1978.

[122] U. Pape, «CHANGES IN NETWORKS AND ADJUSTMENT OF LENGTHS OF SHORTEST PATHS,» *Computing,* vol. 12, p. 357–362, 1974.

[123] M. Desrochers et F. Soumis, «A reoptimization algorithm for the shortest path problem with time windows,» *European Journal of Operational Research,* vol. 35, p. 242–254, 1988.

[124] R. Bellman, «On a Routing Problem,» *Quarterly of Applied Mathematics,* vol. 16, p. 87–90, 1958.

[125] E. W. Dijkstra, «A note on two problems in connexion with graphs:,» *Numerische Mathematik,* vol. 1, pp. 269-271, 1959.

[126] M. Desrochers et U. de Montréal. Centre de recherche sur les transports, La fabrication d'horaires de travail pour les conducteurs d'autobus par une méthode de génération de colonnes, Université de Montréal, Centre de recherche sur les transports, 1986.

[127] J. Desrosiers, Y. Dumas, M. M. Solomon et F. Soumis, «Time constrained routing and scheduling,» *Handbooks in operations research and management science,* vol. 8, p. 35–139, 1995.

[128] G. Desaulniers et D. Villeneuve, «The shortest path problem with time windows and linear waiting costs,» *Transportation Science,* vol. 34, p. 312–319, 2000.

[129] D. Feillet, P. Dejax, M. Gendreau et C. Gueguen, «An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems,» *Networks: An International Journal,* vol. 44, p. 216–229, 2004.

[130] M. Behnke, T. Kirschstein et C. Bierwirth, «A column generation approach for an emission-oriented vehicle routing problem on a multigraph,» *European Journal of Operational Research,* vol. 288, p. 794–809, 2021.

[131] H. Ben Ticha, N. Absi, D. Feillet, A. Quilliot et T. Van Woensel, «A branch-and-price algorithm for the vehicle routing problem with time windows on a road network,» *Networks,* vol. 73, p. 401–417, 2019.

[132] N. Kohl, J. Desrosiers, O. B. G. Madsen, M. M. Solomon et F. Soumis, «2-path cuts for the vehicle routing problem with time windows,» *Transportation Science,* vol. 33, p. 101–116, 1999.

[133] S. Irnich et D. Villeneuve, «The shortest-path problem with resource constraints and k-cycle elimination for k >=3,» *INFORMS Journal on Computing,* vol. 18, p. 391–406, 2006.

[134] R. Fukasawa, H. Longo, J. Lysgaard, M. P. De Aragão, M. Reis, E. Uchoa et R. F. Werneck, «Robust branch-and-cut-and-price for the capacitated vehicle routing problem,» *Mathematical programming,* vol. 106, p. 491–511, 2006.

[135] R. Baldacci, N. Christofides et A. Mingozzi, «An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts,» *Mathematical Programming,* vol. 115, p. 351–385, 2008.

[136] R. Baldacci, E. Bartolini, A. Mingozzi et R. Roberti, «An exact solution framework for a broad class of vehicle routing problems,» *Computational Management Science,* vol. 7, p. 229–268, 2010.

[137] R. Baldacci, A. Mingozzi et R. Roberti, «New route relaxation and pricing strategies for the vehicle routing problem,» *Operations research,* vol. 59, p. 1269–1283, 2011.

[138] S. Dabia, S. Ropke, T. Van Woensel et T. De Kok, «Branch and price for the time-dependent vehicle routing problem with time windows,» *Transportation science,* vol. 47, p. 380–396, 2013.

[139] R. Fukasawa, Q. He et Y. Song, «A branch-cut-and-price algorithm for the energy minimization vehicle routing problem,» *Transportation Science,* vol. 50, p. 23–34, 2016.

[140] D. Pecin, A. Pessoa, M. Poggi et E. Uchoa, «Improved branch-cut-and-price for capacitated vehicle routing,» *Mathematical Programming Computation,* vol. 9, p. 61–100, 2017.

[141] I. Himmich, H. Ben Amor, I. El Hallaoui et F. Soumis, «A primal adjacency-based algorithm for the shortest path problem with resource constraints,» *Transportation Science,* vol. 54, p. 1153–1169, 2020.

[142] R. Sadykov, E. Uchoa et A. Pessoa, «A bucket graph–based labeling algorithm with application to vehicle routing,» *Transportation Science,* vol. 55, p. 4–28, 2021.

[143] I. Mathlouthi, M. Gendreau et J.-Y. Potvin, «Branch-and-price for a multi-attribute technician routing and scheduling problem,» vol. 2, p. 1–35, 2021.

[144] W.-C. Yeh et S.-Y. Tan, «Simplified Swarm Optimization for the Heterogeneous Fleet Vehicle Routing Problem with Time-Varying Continuous Speed Function,» *Electronics,* vol. 10, 2021.

[145] H. Park, D. Son, B. Koo et B. Jeong, «Waiting strategy for the vehicle routing problem with simultaneous pickup and delivery using genetic algorithm,» *Expert Systems with Applications,* vol. 165, p. 113959, 2021.

[146] H. Fan, Y. Zhang, P. Tian, Y. Lv et H. Fan, «Time-dependent multi-depot green vehicle routing problem with time windows considering temporal-spatial distance,» *Computers and Operations Research,* vol. 129, p. 105211, 2021.

[147] G. Srivastava, A. Singh et R. Mallipeddi, «NSGA-II with objective-specific variation operators for multiobjective vehicle routing problem with time windows,» *Expert Systems with Applications,* vol. 176, p. 114779, 2021.

[148] M. R. Garey et D. S. Johnson, «Computers and intractability,» *A Guide to the,* 1979.

[149] M. Dror, «Note on the complexity of the shortest path models for column generation in VRPTW,» *Operations Research,* vol. 42, p. 977–978, 1994.

[150] P. Sun, L. P. Veelenturf, S. Dabia et T. Van Woensel, «The time-dependent capacitated profitable tour problem with time windows and precedence constraints,» *European Journal of Operational*

Research, vol. 264, p. 1058–1073, 2018.

[151] P. Sun, L. P. Veelenturf, M. Hewitt et T. Van Woensel, «The time-dependent pickup and delivery problem with time windows,» *Transportation Research Part B: Methodological,* vol. 116, p. 1–24, 2018.

[152] M. M. Solomon, «Vehicle routing and scheduling with time window constraints: models and algorithms (heuristics),» 1984.

[153] J. E. Beasley et N. Christofides, «An algorithm for the resource constrained shortest path problem,» *Networks,* vol. 19, p. 379–394, 1989.

[154] J. Desrosiers, F. Soumis, M. Desrochers et M. SauveGerad, «Methods for routing with time windows,» *European Journal of Operational Research,* vol. 23, p. 236–245, 1986.

[155] M. Desrochers, J. Desrosiers et M. Solomon, «A new optimization algorithm for the vehicle routing problem with time windows,» *Operations research,* vol. 40, p. 342–354, 1992.

[156] G. Righini et M. Salani, «Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints,» *Discrete Optimization,* vol. 3, p. 255–273, 2006.

[157] A. Chabrier, «Vehicle routing problem with elementary shortest path based column generation,» *Computers & Operations Research,* vol. 33, p. 2972–2990, 2006.

[158] D. Feillet, M. Gendreau et L.-M. Rousseau, «New refinements for the solution of vehicle routing problems with branch and price,» *INFOR: Information Systems and Operational Research,* vol. 45, p. 239–256, 2007.

[159] M. Tagmouti, M. Gendreau et J.-Y. Potvin, «Arc routing problems with time-dependent service costs,» *European Journal of Operational Research,* vol. 181, p. 30–39, 2007.

[160] M. Jepsen, B. Petersen, S. Spoorendonk et D. Pisinger, «Subset-row inequalities applied to the vehicle-routing problem with time windows,» *Operations Research,* vol. 56, p. 497–511, 2008.

[161] G. Righini et M. Salani, «New dynamic programming algorithms for the resource constrained elementary shortest path problem,» *Networks: An International Journal,* vol. 51, p. 155–170, 2008.

[162] G. Desaulniers, F. Lessard et A. Hadjar, «Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows,» *Transportation Science,* vol. 42, p. 387–404, 2008.

[163] A. G. Qureshi, E. Taniguchi et T. Yamada, «An exact solution approach for vehicle routing and scheduling problems with soft time windows,» *Transportation Research Part E: Logistics and Transportation Review,* vol. 45, p. 960–977, 2009.

[164] A. Bettinelli, A. Ceselli et G. Righini, «A branch-and-cut-and-price algorithm for the multi-depot heterogeneous vehicle routing problem with time windows,» *Transportation Research Part C: Emerging Technologies,* vol. 19, p. 723–740, 2011.

[165] F. Liberatore, G. Righini et M. Salani, «A column generation algorithm for the vehicle routing problem with soft time windows,» *4OR,* vol. 9, p. 49–82, 2011.

[166] D. Duque, L. Lozano et A. L. Medaglia, «Solving the orienteering problem with time windows via the pulse framework,» *Computers & Operations Research,* vol. 54, p. 168–176, 2015.

[167] L. Lozano, D. Duque et A. L. Medaglia, «An exact algorithm for the elementary shortest path problem with resource constraints,» *Transportation Science,* vol. 50, p. 348–357, 2016.

[168] D. Pecin, C. Contardo, G. Desaulniers et E. Uchoa, «New enhancements for the exact solution of the vehicle routing problem with time windows,» *INFORMS Journal on Computing,* vol. 29, p. 489–502, 2017.

[169] G. Lera-Romero et J. J. Miranda-Bront, «Integer programming formulations for the time-dependent elementary shortest path problem with resource constraints,» *Electronic Notes in Discrete Mathematics,* vol. 69, p. 53–60, 2018.

[170] K. Dalmeijer et G. Desaulniers, «Addressing Orientation Symmetry in the Time Window Assignment Vehicle Routing Problem,» *INFORMS Journal on Computing,* vol. 33, p. 495–510, 2021.

[171] D. Taş, «Electric vehicle routing with flexible time windows: a column generation solution approach,» *Transportation Letters,* vol. 13, p. 97–103, 2021.

[172] D. J. White, «The set of efficient solutions for multiple objective shortest path problems,» *Computers & Operations Research,* vol. 9, p. 101–107, 1982.

[173] A. Warburton, «Approximation of Pareto optima in multiple-objective, shortest-path problems,» *Operations research,* vol. 35, p. 70–79, 1987.

[174] B. Verlinden, D. Cattrysse, J. R. Duflou et D. Van Oudheusden, «Modeling sheet metal integrated production planning for laser cutting and air bending,» chez *Key Engineering Materials*, 2007.

[175] N. Touati, L. Létocart et A. Nagih, «Solutions diversification in a column generation algorithm,» *Algorithmic Operations Research,* vol. 5, p. 86–95, 2010.

[176] H. B. Ticha, N. Absi, D. Feillet et A. Quilliot, «Empirical analysis for the VRPTW with a multigraph representation for the road network,» *Computers & Operations Research,* vol. 88, p. 103–116, 2017.

[177] L. Taccari, «Integer programming formulations for the elementary shortest path problem,» *European Journal of Operational Research,* vol. 252, p. 122–130, 2016.

[178] T. Stützle et H. H. Hoos, «MAX–MIN ant system,» *Future generation computer systems,* vol. 16, p.

889–914, 2000.

[179] S. Scheuerer et R. Wendolsky, «A scatter search heuristic for the capacitated clustering problem,» *European Journal of Operational Research,* vol. 169, p. 533–547, 2006.

[180] A. E. Rizzoli, F. Oliverio, R. Montemanni et L. M. Gambardella, «Ant Colony Optimisation for vehicle routing problems: from theory to applications,» *Galleria Rassegna Bimestrale Di Cultura,* vol. 9, p. 1–50, 2004.

[181] B. T. Polyak, «A General Method of Solving Extremum Problems,» chez *Doklady Akademii Nauk*, 1967.

[182] J. Pasha, A. L. Nwodu, A. M. Fathollahi-Fard, G. Tian, Z. Li, H. Wang et M. A. Dulebenets, «Exact and Metaheuristic Algorithms for the Vehicle Routing Problem with a Factory-in-a-Box in Multi-Objective Settings,» *Adv. Eng. Inform.,* vol. 52, April 2022.

[183] V. Paschos, Livre, optimisation combinatoire 3: applications, 2005.

[184] M. A. Nguyen, G. T.-H. Dang, M. H. Hà et M.-T. Pham, «The min-cost parallel drone scheduling vehicle routing problem,» *European Journal of Operational Research,* vol. 299, pp. 910-930, 2022.

[185] M. Minoux, «Plus court chemin avec contraintes: algorithmes et applications,» chez *Annales des télécommunications*, 1975.

[186] E. Manousakis, P. Repoussis, E. Zachariadis et C. Tarantilis, «Improved branch-and-cut for the Inventory Routing Problem based on a two-commodity flow formulation,» *European Journal of Operational Research,* vol. 290, p. 870–885, 2021.

[187] M. E. Lübbecke et J. Desrosiers, «Selected topics in column generation,» *Operations research,* vol. 53, p. 1007–1023, 2005.

[188] T. Liao, T. Stützle, M. A. M. de Oca et M. Dorigo, «A unified ant colony optimization algorithm for continuous optimization,» *European Journal of Operational Research,* vol. 234, p. 597–609, 2014.

[189] G. Lera-Romero et J. J. Miranda-Bront, «A branch and cut algorithm for the time-dependent profitable tour problem with resource constraints,» *European Journal of Operational Research,* vol. 289, p. 879–896, 2021.

[190] J. K. Lenstra et A. R. Kan, «Some simple applications of the travelling salesman problem,» *Journal of the Operational Research Society,* vol. 26, p. 717–733, 1975.

[191] A. LAMAMRI, H. A. I. T. HADDADENE et A. NAGIH, «Une Approche pour l'accélération de la génération de colonnes appliquées au problème de rotations d'équipages,» chez *COSI'2010*, 2010.

[192] A. LAMAMRI, H. A. I. T. HADDADENE et A. NAGIH, «An approach combining re-optimization and reduction of state space for problem solving construction crew rotations,» chez *CIRO'10*, 2010.

[193] M. K. Jepsen, B. Petersen, S. Spoorendonk et D. Pisinger, «A branch-and-cut algorithm for the capacitated profitable tour problem,» *Discrete Optimization,* vol. 14, p. 78–96, 2014.

[194] Y. Huang, L. Zhao, T. Van Woensel et J.-P. Gross, «Time-dependent vehicle routing problem with path flexibility,» *Transportation Research Part B: Methodological,* vol. 95, p. 169–195, 2017.

[195] M. I. Henig, «The shortest path problem with two objective functions,» *European Journal of Operational Research,* vol. 25, p. 281–291, 1986.

[196] G. Y. Handler et I. Zang, «A dual algorithm for the constrained shortest path problem,» *Networks,* vol. 10, p. 293–309, 1980.

[197] F. Glover, «A template for scatter search and path relinking,» chez *European conference on artificial evolution*, 1997.

[198] G. D. Finlayson, M. S. Drew et C. Lu, «Entropy Minimization for Shadow Removal,» *Int J Comput Vis,* vol. 85, p. 35–37, 2009.

[199] M. Drexl et S. Irnich, «Solving elementary shortest-path problems as mixed-integer programs,» *OR spectrum,* vol. 36, p. 281–296, 2014.

[200] M. Drexl, «A note on the separation of subtour elimination constraints in elementary shortest path problems,» *European Journal of Operational Research,* vol. 229, p. 595–598, 2013.

[201] J. Dréo, A. Pétrowski, P. Siarry et E. Taillard, Métaheuristiques pour l'optimisation difficile, Eyrolles, 2003.

[202] C. H. A. N. DONALD et D. Mercier, «IC insertion: an application of the travelling salesman problem,» *The International Journal of Production Research,* vol. 27, p. 1837–1841, 1989.

[203] E. W. Dijkstra, «A note on two problems in connexion with graphs:(Numerische Mathematik, 1 (1959), p 269-271),» 1959.

[204] J. Desrosiers, P. Pelletier et F. Soumis, «Plus court chemin avec contraintes d'horaires,» *RAIRO-Operations Research,* vol. 17, p. 357–377, 1983.

[205] M. Desrochers et F. Soumis, «A generalized permanent labelling algorithm for the shortest path problem with time windows,» *INFOR: Information Systems and Operational Research,* vol. 26, p. 191–212, 1988.

[206] C. M. Damião, J. M. P. Silva et E. Uchoa, «A branch-cut-and-price algorithm for the cumulative capacitated vehicle routing problem,» *4OR,* p. 1–25, 2021.

[207] N. Christofides, A. Mingozzi et P. Toth, «Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations,» *Mathematical programming,* vol. 20, p. 255–282, 1981.

[208] H. Ben Ticha, N. Absi, D. Feillet et A. Quilliot, «Vehicle routing problems with road-network information: State of the art,» *Networks,* vol. 72, p. 393–406, 2018.

[209] R. Bellman, Dynamic Programming, 1 éd., Princeton, NJ: Princeton University Press, 1957.

[210] P. Avella, M. Boccia et A. Sforza, «A penalty function heuristic for the resource constrained shortest path problem,» *European Journal of Operational Research,* vol. 142, pp. 221-230, 2002.

[211] C. Archetti et M. G. Speranza, «A survey on matheuristics for routing problems,» *EURO Journal on Computational Optimization,* vol. 2, p. 223–246, 2014.

[212] T. S. Abdul-Razaq et C. N. Potts, «Dynamic Programming State-Space Relaxation for Single-Machine Scheduling,» *Journal of the Operational Research Society,* vol. 39, p. 141–152, 1988.