Utilisation automatique de la génération de colonnes dans Hexaly

Noémie Cartier[1], Julien Darlay[1], Bienvenu Bambi[1]

Hexaly, 251 Boulevard Pereire, Paris, France
{ncartier, jdarlay, bbambi}@hexaly.com

Mots-clés : *recherche opérationnelle, optimisation, génération de colonnes, routing.*

1 Introduction

Hexaly est un solveur d'optimisation mathématique basé sur différentes techniques de recherche opérationnelle, combinant des méthodes exactes, telles que la programmation linéaire, non linéaire et par contraintes, et heuristiques, comme la recherche locale. Son but est d'offrir une approche de type « model-and-run » à des problèmes d'optimisation (combinatoires, continus, mixtes...), y compris sur de grandes instances [1].

Nous présentons ici comment Hexaly parvient à utiliser automatiquement la génération de colonnes pour donner des bornes inférieures sur un grand ensemble de problèmes de tournées de véhicules industriels, grâce notamment à son formalisme ensembliste flexible.

2 Modélisation des problèmes de routing avec des listes

Une solution d'un problème de tournées de véhicules (type CVRPTW) s'exprime naturellement sous la forme d'un ensemble de listes formant une partition de l'ensemble des clients et respectant un certain nombre de contraintes. Cette modélisation est bien adaptée aux algorithmes heuristiques de recherche de solutions, mais plus difficile à intégrer à une approche mathématique visant à calculer des bornes inférieures sur l'objectif. Une reformulation directe introduit un nombre quadratique de variables de décisions binaires ainsi que plusieurs contraintes « big M », ce qui pose des problèmes de passage à l'échelle et peut donner des mauvaises bornes.

Cependant, l'approche par liste s'intègre bien dans le paradigme de la génération de colonnes, ou *branch cut and price*, où chaque variable correspond à une tournée possible. Les variables représentant des tournées sont ajoutées au fur et à mesure par un algorithme de pricing qui résout un problème de plus court chemin sous contrainte de ressources par la programmation dynamique [2]. Cette méthode de résolution a donné toutes les nouvelles preuves d'optimalité sur les instances CVRPLIB des dix dernières années [3].

Néanmoins, l'intégration des contraintes de problèmes industriels – souvent très éloignés des problèmes académiques – au pricing est peu accessible pour des utilisateurs non-experts.

3 Approche d'Hexaly

Les variables de type liste des modèles d'Hexaly permettent une modélisation efficace et compacte des problèmes de tournées de véhicules, et se prêtent bien à la détection automatique de ce type de problèmes. De plus, l'utilisation en parallèle d'algorithmes de recherche de solutions basés sur des heuristiques permet de relâcher certaines contraintes pour le calcul de bornes sans perte de qualité ou de justesse sur les solutions obtenues. Tout ceci rend possible la détection de structures de problèmes de routing et l'exécution d'un solveur de type branch cut

price dans de nombreux cas concrets potentiellement très exotiques, sans ajout de complexité lors de la modélisation.

Hexaly 13.5 détecte automatiquement les problèmes de type CVRPTW multi-dépôts et/ou avec des camions hétérogènes. D'autres extensions ultérieures sont possibles, par exemple aux problèmes de type *pick-up and delivery*, *prize collecting*, ou même à des modèles ensemblistes.

4 Résultats

Pour des raisons de complexité spatiale et temporelle, la génération de colonnes ne se lance que pour des instances de taille et de complexité raisonnable (au plus 200 clients). Le reste du temps, on calcule des bornes grâce à un modèle linéaire dont les variables correspondent aux arcs du graphe. On comparera donc les résultats de la génération de colonne à ceux de cet autre modèle uniquement sur les instances traitées par la génération de colonne.

La proportion que représentent ces instances dans chaque set est donnée en troisième colonne du tableau. Il s'agit de l'intégralité des sets A et B de la CVRPLIB, de presque trois quarts des instances à moins de 200 clients du set X, de plus de la moitié du set Solomon pour le CVRPTW et de près d'un tiers des instances du set HVRP DLP.

Les résultats suivants sont tous obtenus avec une limite de temps de 60 secondes.

Type	Instances	Traitées	Gap moyen	Ancient gap
CVRP	Set A	100%	0.11%	9.24%
	Set B	100%	0.06%	0.93%
	Set X ($n \leqslant 200$)	72.73%	3.74%	9.64%
CVRPTW	Solomon	59%	0.6%	9.03%
HVRP	DLP set	32.29%	10.9%	86.03%

TAB. 1 – Résultats

On constate que la génération de colonnes donne de bien meilleures bornes que l'approche par arc, surtout pour les instances de HVRP, notamment parce qu'elle permet une prise en compte beaucoup plus fine des caractéristiques des camions. Elle permet de plus de fermer la plupart des instances des sets A (63%) et B (78%) très rapidement.

Elle reste cependant limitée par la complexité du problème de pricing qui augmente très rapidement avec certaines caractéristiques du problème, et le solveur préférera donc l'autre approche, moins coûteuse, si il estime que cette complexité sera trop grande.

Références

[1] F. Gardi, T. Benoist, J. Darlay, B. Estellon, et R. Megel. *Mathematical Programming Solver Based on Local Search*, Wiley, 2014.

[2] Artur Alves Pessoa, Ruslan Sadykov, Eduardo Uchoa, and François Vanderbeck. A Generic Exact Solver for Vehicle Routing and Related Problems. *Mathematical programming*, 183 : 483–523, 2020.

[3] http://vrp.galgos.inf.puc-rio.br/index.php/en/updates