Calcul de bornes inférieures pour les problèmes de tournées dans LocalSolver

Alexandre Bontems[1]

LocalSolver, 36 avenue Hoche, 75008 Paris
abontems@localsolver.com

Mots-clés : *voyageur de commerce, bornes inférieures.*

1 Introduction

LocalSolver est un solveur d'optimisation mathématique de type « model and run » combinant différentes techniques de recherche opérationnelle [1]. il permet de modéliser des problèmes d'optimisation avec des variables de décision classiques (variables booléennes, entières, et flottantes) mais offre aussi la possibilité d'écrire des modèles avec des variables qui manipulent directement des collections d'entiers. Une variable de type `list` est ainsi définie par le nombre d'éléments maximum n qu'elle peut contenir et a pour domaine de définition tous les sous-ensembles ordonnés de $\{0, ..., n-1\}$. Ce type de variables est particulièrement utile pour modéliser des problèmes de tournées et le solveur peut tirer avantage de cette structure pour améliorer ses performances. Afin de pouvoir mesurer la qualité des solutions produites pour des problèmes de type *voyageur de commerce*, des techniques dédiées aux listes ont été implémentés dans LocalSolver.

2 Calcul de bornes

2.1 Relaxation lagrangienne de Held & Karp

Étant donné un graphe non-orienté $G = (V, E)$ et un coût c_{ij} défini pour chaque arête (i, j), le problème du voyageur de commerce consiste à déterminer un circuit hamiltonien de coût minimum dans G. Pour calculer une borne inférieure on s'intéresse à la relaxation lagrangienne de Held & Karp [2] qui consiste à calculer itérativement des 1-trees.

Soit i un sommet quelconque de G, on appelle 1-tree un arbre couvrant dans le sous-graphe induit par $V \setminus \{i\}$ auquel on ajoute deux arêtes incidentes à i. On remarque qu'un tour est en réalité un 1-tree dans lequel tous les sommets sont de degrés 2 et qu'un 1-tree de coût minimum est donc une borne inférieure de la valeur d'un tour optimal dans G.

Soit π un vecteur poids qui associe à chaque sommet i un poids π_i, si on redéfinit le coût de chaque arête :

$$\bar{c}_{ij} = c_{ij} + \pi_i + \pi_j \tag{1}$$

L'ordre des solutions du voyageur de commerce n'est pas affecté par cette modification car on rajoute exactement $2\sum_{i \in V} \pi_i$ à chaque tour (chaque sommet apparaît exactement deux fois parmi les extrémités des arêtes d'un tour). En revanche la solution au problème du 1-tree sera sensible à cette modification. On peut donc concevoir un algorithme de sous-gradient qui, à chaque itération, perturbe les poids π_i pour essayer de produire des 1-trees qui ressemblent de plus en plus à des tours. Soit $\bar{c}(T)$ le coût d'un 1-tree de coût minimum avec les coûts modifiés $\bar{c}_{ij}$, la valeur $\bar{c}(T) - 2\sum_{i \in V} \pi_i$ est une borne inférieure du problème original.

Dans LocalSolver des modifications de l'algorithme ont été implémentées pour réduire les temps d'exécution, gérer le cas asymétrique et pouvoir calculer une borne du problème du

chemin hamiltonien de coût minimum. À partir de la version 8.5, lorsqu'un voyageur de commerce (modélisé à l'aide d'une variable `list`) est détecté dans un modèle, le solveur calcule automatiquement une borne pour le problème.

2.2 Utilisation de solutions primales issues du solveur

Dans l'espoir d'obtenir de meilleures bornes et de pouvoir potentiellement prouver l'optimalité d'une solution au cours de la résolution, l'utilisation des solutions produites par le solveur a aussi été étudiée. Dans cette partie on s'intéresse à la relaxation du programme linéaire en nombre entiers du voyageur de commerce avec les contraintes d'élimination de sous-tours :

$$\sum_{(i,j)\in\delta(S)} x_{ij} \geq 2 \qquad \forall S \subset V,\, S \neq \emptyset \tag{2}$$

On peut remarquer que dans une solution optimale du voyageur de commerce, seuls les sous-ensembles de sommets adjacents dans la solution vont correspondre à des contraintes serrées. Plusieurs méthodes qui utilisent la structure des solutions produites par le solveur (qu'on suppose de bonne qualité) ont ainsi été testées pour calculer des solutions duales en profitant de cette propriété (algorithme glouton, optimisation par programmation linéaire, etc).

3 Résultats

L'implémentation au sein de LocalSolver s'exécute lors du *preprocessing* et permet de mesurer la qualité des solutions dès le début de la résolution. La table 1 présente les bornes calculées sur quelques instances de la TSPLib. Les instances dans la seconde partie du tableau correspondent à des instances asymétriques.

Instance	**Opt**	**Borne**	**Gap (%)**	**Temps (secs)**
gr48	5 046	4 950	1.9	0.020
st70	675	670	0.7	0.027
ch130	6 110	6 065	0.7	0.054
si175	21 407	21 321	0.4	0.130
a280	2 579	2 562	0.6	0.155
rat575	6 773	6 713	0.8	0.353
d1291	50 801	49 136	3.2	0.989
br17	39	39	0.0	0.005
ftv70	1 950	1 906	2.2	0.037
kro124p	36 230	35 924	0.8	0.053
ftv170	2 755	2 703	1.8	0.072

TAB. 1 – Bornes obtenues par le préprocessing de LocalSolver

Références

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

[2] M. Held et R. M. Karp. *The Traveling-Salesman Problem and Minimum Spanning Trees.* Operations Research 18.6 (1970) p. 1138-1162.