Linéarisation de variables ensemblistes dans LocalSolver

Nicolas Blandamour

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

Mots-clés : *optimisation combinatoire, linéarisation, recherche locale, programmation linéaire mixte.*

1 Contexte

LocalSolver est un solveur d'optimisation mathématique de type « model & run ». Il permet de modéliser un problème d'optimisation avec des opérateurs mathématiques usuels et produit des solutions rapidement. Combinant différentes techniques dans le cadre d'une approche heuristique, LocalSolver parvient à traiter des problèmes non linéaires ou combinatoires de grande taille, sur des ordinateurs standards.

LocalSolver permet de modéliser des problèmes d'optimisation avec les variables de décision classiques de la programmation linéaire en nombres entiers : variables booléennes, entières et flottantes. Cependant, depuis les versions 5.5 [1] et 8.0 [2], deux nouvelles variables de décision ont été ajouté aux variables de décisions classiques : les listes et les ensembles. Ces variables de collection (structures permettant de stocker un ensemble de valeurs) ont permis de faciliter la modélisation et d'améliorer les performances du solveur pour de nombreux types de problèmes, tel que les problèmes de tournées de véhicules, d'ordonnancement ou de remplissage.

2 Variable de *set* et problème de *bin-packing*

Une variable ensembliste, appelée *set* dans le formalisme de LocalSolver, est définie sur un ensemble discret (les entiers de 0 à $n-1$) et représente un sous-ensemble (non-ordonné) de cet ensemble. Ce type de variable permet de faciliter la modélisation des problèmes de remplissage, tel que le *bin-packing*.

Le problème de *bin-packing* consiste à ranger un ensemble d'objets dans un minimum de boîtes. La modélisation LocalSolver de ce problème consiste à créer un *set* par boîte, pouvant contenir l'ensemble des objets. On s'assure ensuite que le contenu de chaque boîte ne dépasse la capacité, et que chaque objet est présent dans un et un seul *set* (avec l'opérateur de modélisation `partition` spécifique aux variables de collection) :

```
bins[k in 0..nbMaxBins-1] <- set(nbItems);
for [k in 0..nbMaxBins-1] {
    binWeights[k] <- sum(bins[k], i => itemWeights[i]);
    constraint binWeights[k] <= binCapacity;
    binsUsed[k] <- (count(bins[k]) > 0);
}
constraint partition[k in 0..nbMaxBins-1](bins[k]);
totalBinsUsed <- sum[k in 0..nbMaxBins-1](binsUsed[k]);
minimize totalBinsUsed;
```

1. *Release notes* de LocalSolver 5.5 : https://www.localsolver.com/news.html?id=60
2. *Release notes* de LocalSolver 8.0 : https://www.localsolver.com/news.html?id=81

3 Linéarisation de modèles utilisant des *sets*

Dans l'optique d'offrir à l'utilisateur la meilleure solution possible à son problème, LocalSolver combine différentes techniques avancées d'optimisation autres que la recherche locale. L'une d'entre elles consiste à résoudre le problème avec des techniques de programmation linéaire mixte.

Pour cela, la première étape consiste à transformer la représentation non-linéaire du problème, issue du formalisme de modélisation LocalSolver, en une représentation linéaire adaptée à une résolution par un solveur MIP.

Au cours de l'exposé, nous détaillerons les différentes étapes nécessaires à la linéarisation d'un problème utilisant des variables de *sets*, ainsi que les raffinements permettant d'obtenir un meilleur modèle linéarisé pour le problème de *bin-packing* (formulation de Kantorovich [1]).

Nous présenterons par la suite les résultats pratiques de telles linéarisations, en détaillant la complémentarité d'une résolution du problème par un solveur MIP avec une résolution basée sur des techniques de recherche locale, tant du point de vue de la génération de solutions que sur la remontée de bornes. Nous illustrerons ces résultats avec deux exemples :

— le problème de *bin-packing* pour lequel la linéarisation permet d'obtenir une borne inférieure grâce au solveur MIP, ce qui couplé aux solutions obtenues par la recherche locale, permet de quantifier de façon précise le *gap* d'optimisation.

— le problème de couverture par ensembles (*set-cover*), pour lequel le solveur MIP vient cette fois suppléer la recherche locale sur la partie primale.

Références

[1] Leonid Vitaliyevich Kantorovich. Mathematical methods of organizing and planning production. 6 :366–422, jul 1960.