Calcul de bornes dans LocalSolver 9.5

Nikolas Stott

LocalSolver, 36 avenue Hoche, Paris, France
nstott@localsolver.com

Mots-clés : *Solveur, Optimisation globale, variables mixtes*

1 Contexte

LocalSolver est un solveur d'optimisation mathématique tout terrain qui permet de modéliser des problèmes formulés avec les variables classiques de la programmation linéaire en nombres entiers (booléens, entiers et réels). Il permet également d'utiliser des variables de type collection (listes et ensembles) pour modéliser des problèmes de tournées de véhicules, de rangement et d'ordonnancement.

La recherche de solution repose sur le coeur historique de méthodes heuristiques, en utilisant des techniques de recherche locale, et sur un module d'optimisation globale pour traiter les problèmes de type MINLP.

Le formalisme de LocalSolver accepte naturellement plusieurs objectifs qui sont traités lexicographiquement. Afin d'attester de la qualité des solutions obtenues, LocalSolver calcule des bornes inférieures (resp. supérieures) pour chacun des objectifs minimisés (resp. maximisés). L'objectif de LocalSolver est de produire des solutions de qualité rapidement sur tout l'éventail des problèmes modélisables. Ces solutions sont ensuite raffinées. La même stratégie est adoptée pour le calcul de bornes sur les valeurs des objectifs : obtenir de premières bornes de qualité rapidement, puis les améliorer itérativement.

Cet exposé présente l'ensemble de techniques utilisées dans LocalSolver pour produire des bornes et illustre leur application sur un ensemble de benchmarks de la littérature et de cas industriels.

2 Techniques de calcul de bornes

2.1 Inférence et propagation lors du preprocessing

LocalSolver représente un problème d'optimisation sous la forme d'un graphe dirigé acyclique dans lequel les feuilles sont les variables de décision et les racines sont les contraintes et les objectifs. Il stocke pour chaque noeud du graphe les valeurs qu'il peut prendre sous la forme d'un intervalle. Les bornes associées à chaque noeud peuvent être déduites des bornes de ses enfants, ou inférées à partir des bornes de ses parents. Cette arithmétique d'intervalle permet d'obtenir des premières bornes dès le preprocessing du modèle sur tous les objectifs.

2.2 Problèmes entiers mixtes linéaires et non-linéaires

Dans le cas d'un seul objectif, le module d'optimisation globale reformule le problème et produit des bornes valides au cours de la résolution à l'aide de relaxations linéaires/convexes couplées avec un mécanisme de *branch-and-reduce*. Si le problème est purement linéaire, il est résolu directement par le solveur linéaire sous-jacent.

LocalSolver trouve ainsi une solution optimale à 40% des problèmes de la MINLPLib (et des solutions faisables pour 70% d'entre eux) en moins d'une minute.

Dans le cas multi-objectif, le calcul des bornes traite les objectifs un par un. Chaque objectif doit être résolu jusqu'à l'optimum global en contraignant chacun des objectifs précédents à leur valeur optimale avant de traiter le suivant. Nous illustrons les apports de cette approche sur un ensemble de problèmes industriels, sur lesquels la résolution rapide des premiers objectifs entraîne des améliorations significatives sur la résolution des objectifs suivants.

2.3 Problèmes avec collections

Le sous-solveur MINLP permet également d'obtenir des solutions et des bornes lorsque les variables de décisions sont des ensembles, par linéarisation booléenne exacte de la structure ensemblistes et des contraintes associées.

Sur les problèmes de type *bin-packing*, la formulation renforcée de Kantorovich permet d'obtenir très rapidement une borne inférieure de grande qualité, qui ne diffère en pratique souvent que de 1 de la solution optimale.

Les problèmes de tournées sont modélisés avec des listes dans LocalSolver. Le calcul de bornes utilisant seulement des arguments de propagation obtient des bornes trop faibles, et il est donc traité avec une technique dédiée : la résolution de la relaxation lagrangienne de Held-Karp pour les TSP symétriques et asymétriques. Nous obtenons ainsi rapidement des solutions avec des gaps inférieurs à 3% sur les instances de la TSPLib comprenant moins de 1000 villes.