Vers un solveur de programmation mathématique généralisée basé sur la recherche locale

Thierry Benoist[1], Julien Darlay[1], Bertrand Estellon[2], Frédéric Gardi[1], Romain Megel[1]

[1] LocalSolver, Paris, France
http://www.localsolver.com

[2] LIF CNRS - Aix-Marseille Université, Marseille, France
bertrand.estellon@lif.univ-mrs.fr

Mots-clés : *LocalSolver, programmation mathématique, recherche locale.*

Débuté en 2007 comme pur projet de R&D, LocalSolver [1] nous a permis de montrer qu'une approche par recherche locale est possible pour la programmation non-linéaire en variables 0-1. Mieux, la puissance qu'offre une recherche locale pure et directe nous permet d'attaquer des problèmes combinatoires de très grande taille, hors d'atteinte des solveurs actuels parce qu'essentiellement basés sur des techniques arborescentes. Suite à ce succès technique, une version commerciale de LocalSolver a été lancée début 2012. Toutefois, le produit reste gratuit pour l'enseignement et la recherche fondamentale (visitez `localsolver.com` pour plus de détails).

Notre approche par recherche locale repose sur des mouvements très efficaces en temps tendant à maintenir la faisabilité des solutions (c'est-à-dire respectant toutes les contraintes du problème). L'idée maîtresse consiste à modifier la valeur courante de certaines décisions tout en réparant les contraintes violées suite à ces modifications. Dans cette note, nous donnons un aperçu de nos travaux en cours pour étendre cette technique à l'optimisation en variables continues ou mixtes. Plus généralement, nous présentons la feuille de route du projet LocalSolver à court et moyen terme.

1 Vers une optimisation en variables continues ou mixtes

La recherche locale est une technique que l'on peut en fait retrouver en optimisation continue, sous une autre appellation : recherche directe (*direct search* en anglais). Cette technique a été introduite dans les années soixante pour la programmation non-linéaire sans contrainte (avec variables continues) à travers "l'autre méthode du simplexe", l'algorithme du simplexe de Nelder-Mead [4, 7]. La recherche directe [3] se rapporte aux méthodes sans dérivée aussi appelées d'ordre zéro, par opposition aux méthodes de premier ordre requérant le calcul du gradient comme les méthodes de quasi-Newton, ou encore aux méthodes de second ordre requérant le calcul de la hessienne comme les méthodes de Newton (voir [5] pour plus de détails). À la lecture du récent article de synthèse de Kolda et al. [3], le parallèle entre recherche locale en optimisation combinatoire et recherche directe en optimisation continue est flagrant : la simplicité conceptuelle de l'approche, sa nature heuristique, la notion de "mouvements" appliqués à chaque itération [3, p. 389]), l'évaluation rapide de chaque itération parce que directement basé sur l'objectif. De façon amusante, la similarité ne s'arrête pas aux aspects techniques : "Mathematicians hate it because you can't prove convergence ; engineers seem to love it because it often works" disait lui-même John Nelder [7, p. 274].

D'un autre côté, de façon similaire à la recherche locale en optimisation combinatoire, la recherche directe apparaît comme limitée pour attaquer des problèmes contraints, même si de sérieux progrès ont été réalisés ces dernières années [3]. Nous travaillons actuellement à *unifier*

recherche locale en optimisation combinatoire et recherche directe en optimisation continue. En particulier, nous adaptons les mouvements de recherche locale décrits précédemment pour la programmation 0-1 à la gestion de décisions continues. Les idées principales derrière ces mouvements locaux sur variables continues (ou même mixtes) restent les mêmes : nous tendons à maintenir la faisabilité du mouvement par réparation des contraintes violées, et nous évaluons le mouvement d'une façon très efficace par des techniques d'algorithmique incrémentale. Ces fonctionnalités seront disponibles dans la version 4.0 de LocalSolver. En résumé, notre but est de permettre aux utilisateurs de LocalSolver d'attaquer de *très grands problèmes d'optimisation non-linéaire en variables mixtes*, impliquant des décisions 0-1 et continues.

2 Un solveur de nouvelle génération

Trouver de bonnes solutions (c'est-à-dire calculer des bornes supérieures) et prouver l'optimalité (c'est-à-dire calculer des bornes inférieures) sont deux choses très différentes, mettant en jeu des algorithmes très différents. Par exemple, les solveurs de PLNE modernes utilisent des heuristiques pour produire rapidement de bonnes solutions avant de débuter la recherche arborescente. Ainsi, ces deux tâches sont traitées *séparément* dans LocalSolver : de bonnes solutions faisables sont produites (rapidement) par la recherche locale, tandis que les bornes inférieures sont produites par des techniques d'inférence. Bien entendu, des informations sont échangées entre les deux processus au cours de l'exécution.

Pour les prochaines versions de LocalSolver, nous prévoyons d'améliorer chacun des processus comme suit : exploiter la recherche arborescente dans le calcul de solutions admissibles, notamment comme mouvement de recherche locale à voisinage très large ; améliorer les bornes inférieures en exploitant de meilleures techniques d'inférence ainsi que des relaxations (primales ou duales, linéaires ou convexes), éventuellement renforcées par recherche arborescente (se référer [2, 6] pour des idées sur chacun de ces thèmes).

Sur le long terme, nous souhaitons ajouter des décisions entières ainsi que des opérateurs ensemblistes dans le formalisme de LocalSolver, de façon à faciliter la modélisation des problèmes de routage et d'ordonnancement (et permettre une résolution plus efficace de ces problèmes, notamment en complexité mémoire). Néanmoins, mixer ce formalisme plus riche avec le formalisme classique aux décisions 0-1 est délicat (pour l'utilisateur notamment). Un premier prototype offrant des résultats concluants a été réalisé sur ce sujet.

Références

[1] T. Benoist, B. Estellon, F. Gardi, R. Megel, K. Nouioua (2011). LocalSolver 1.x : a black-box local-search solver for 0-1 programming. *4OR* 9(3), pp. 299-316.

[2] T. Berthold, S. Heinz, M.E. Pfetsch (2009). Nonlinear pseudo-Boolean optimization : relaxation or propagation ?. In *Proceedings of SAT 2009*, *LNCS 5584*, pp. 441–446. Springer.

[3] T.G. Kolda, R.M. Lewis, V. Torczon (2003). Optimization by direct search : new perspectives on some classical and modern methods. *SIAM Review* 45(3), pp. 385–482.

[4] J.A. Nelder, R. Mead (1965). A simplex method for function minimization. *Computer Journal* 7, pp. 308–313.

[5] M. Minoux (2007). *Programmation Mathématique : Théorie et Algorithmes*. Éditions Tec & Doc, Lavoisier. (2ème edition)

[6] E. Rothberg (2007). An evolutionary algorithm for polishing mixed integer programming solutions. *INFORMS Journal on Computing* 19(4), pp. 534–541.

[7] M.H. Wright (2012). Nelder, Mead, and the other simplex method. In *Optimization Stories, 21st ISMP Berlin 2012*, pp. 271–276. Documenta Mathematica.