Génération automatique de voisinages de grande taille pour la recherche locale autonome

Thierry Benoist[1], Bertrand Estellon[2], Frédéric Gardi[1], Romain Megel[1], Karim Nouioua[2]

[1] Bouygues e-lab
40, rue de Washington, 75 008 Paris, France
{tbenoist,fgardi,rmegel}@bouygues.com

[2] Laboratoire d'Informatique Fondamentale - CNRS UMR 6166,
Faculté des Sciences de Luminy - Université Aix-Marseille II, Marseille, France
{bertrand.estellon,karim.nouioua}@lif.univ-mrs.fr

Mots-clés *: Recherche Locale Autonome, Couplages, Programmation par recherche locale*

1 Introduction

Le fonctionnement idéal d'un solveur de problèmes d'optimisation est celui dans lequel l'utilisateur modélise simplement son problème et se repose ensuite sur le solveur pour y trouver des solutions de bonne qualité. C'est ce paradigme *model&run* qui a fait le succès des solveurs de programmation linéaire en nombres entiers. Dans le domaine de la recherche locale, *LocalSolver* [1] poursuit ce même but (http://www.localsolver.com).

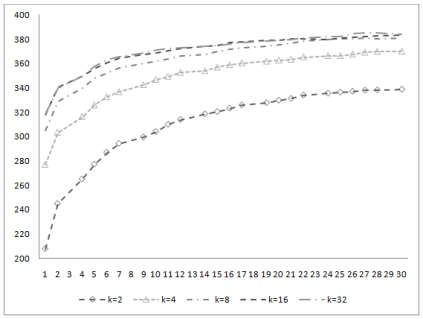
Nous nous intéressons ici aux voisinages de grande taille (*Very Large Scale Neighborhoods*). En particulier certains voisinages de taille exponentielle peuvent être explorés en un temps polynomial et sont parfois utiles pour améliorer la convergence d'une recherche locale sur des instances difficiles. Par exemple pour le problème du *Car Sequencing*, Estellon et al.ont montré dans [2] qu'en choisissant K positions suffisamment distantes les unes des autres, il était possible de repositionner optimalement ces K véhicules via la résolution d'un problème de transport.

Nous montrerons que certains grands voisinages partagent des bases communes et peuvent alors être implémenté de façon unifiée dans un solveur autonome. Plus précisément nous montrerons que certains voisinages basés sur une structure de couplage (ou plus généralement de transport) peuvent être conçu automatiquement en cherchant des ensembles stables dans un certain graphe. Nous proposerons également une procédure simple permettant de précalculer implicitement des milliards de stables de ce type.

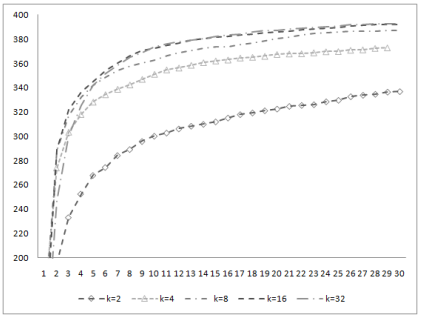
Nous illustrerons ces idées sur le puzzle Eternity II (http://us.eternityii.com), qui consiste à placer 256 pièces carrées sur une grille de taille 16x16, de façon à maximiser le nombre d'arêtes « accordées », c'est-à-dire sur lesquelles les deux pièces adjacentes présentent la même couleur. Enfin nous présenterons les résultats obtenus sur différents problèmes en implémentant ce mécanisme dans Localsolver.

2 Résultats

Ce mécanisme a été implémenté dans le cadre du solveur LocalSolver et testé sur un certain nombre de problèmes. Pour le puzzle Eternity II nous obtenons les résultats suivants. Notons que la phase de détection et de calcul des stables prend moins d'une seconde sur ce problème à plus de 250 000 variables binaires.



Schaus&Deville Intel Xeon(TM) 2.80GHz
(kindly provided by Pierre Schaus)



Automatic LNS, Intel Xeon(TM) 3 GHz

Ces courbes représentent des descentes de 30 secondes sur ce puzzle, avec différentes tailles de voisinages (de 2 à 32). Le graphique de gauche est extrait de [3] alors que celui de droite a été obtenu avec notre algorithme automatique. La similarité de ces courbes confirme que, sans connaissances a priori sur la structure du problème, nous parvenons à explorer le même voisinage avec la même efficacité.

Nous avons testé cette implémentation générique sur différents problèmes. Pour la plus grande instance de chaque problème, la tableau ci-dessous donne la taille de la structure bipartie détectée, le nombre d'ensembles stables implicitement générés, le temps total de cette analyse initiale et le nombre de mouvements par seconde pendant la recherche locale. Dans ces expériences la taille des voisinages était réglée à 8.

Problem	*Bipartite size*	*# of (implicit) stable sets*	*Moves per second*
Car sequencing with colors	1319x284	10^{24} in 3.2s	590
Car Sequencing CspLib	500x20	10^{17} in 0.1s	545
Eternity II	256x256	10^{13} in 0.5s	990
Social Golfer	30x10 (13 times)	1 in 1.6s	285

Ces résultats montrent qu'il est possible de générer automatiquement de grands voisinages grâce à une analyse du modèle. Tout comme les voisinages offerts par défaut dans LocalSolver, ces grands voisinages préservent la faisabilité de la solution et sont similaires à ce qu'un ingénieur programmerait.

Références

[1] Thierry Benoist, Bertrand Estellon, Frederic Gardi, Karim Nouioua, *Toward Local Search Programming: LocalSolver 1.0*. In CPAIOR Workshop : Open Source Tools for Constraint Programming and Mathematical Programming (2010).

[2] Estellon B., Gardi F., Nouioua K. , *Large neighborhood improvements for solving car sequencing problems*. RAIRO Operations Research 40(4) pp 355-379 (2006).

[3] Shaus P., Deville Y, *Hybridization of CP and VLNS for Eternity II*. JFPC'08 Quatrième Journées Francophones de Programmation par Contraintes, Nantes, (2008).