Une version Multithread du solveur BiqCrunch

Franck Butelle[1] Camille Coti[1] Etienne Leclercq[1] Frédéric Roupin[1]

LIPN, UMR 7030, Université Paris 13, Sorbonne Paris Cité
{butelle,coti,leclercq,roupin}@lipn.univ-paris13.fr

Mots-clés : *Calcul parallèle, Branch-and-bound, Optimisation quadratique en variables 0-1*

1 Introduction

BiqCrunch [3] permet de résoudre de manière exacte tout programme quadratique ou linéaire en variables 0-1. Ce logiciel est open-source et son code ainsi que de nombreux résultats expérimentaux sont disponibles à l'adresse `http://www-lipn.univ-paris13.fr/BiqCrunch/`.

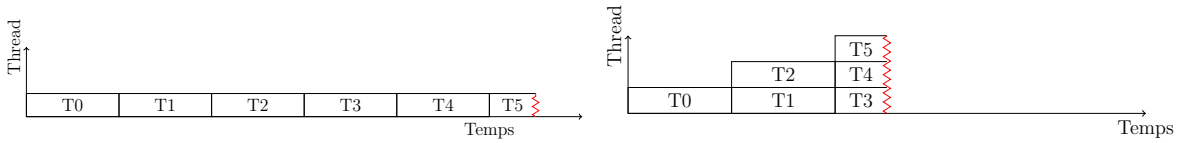
BiqCrunch est un algorithme de branch-and-bound fondé sur une famille de relaxations non-linéaires originales [6] permettant d'ajuster dynamiquement la qualité et donc le temps de calcul de l'évaluation de chaque nœud de l'arbre de recherche. De plus, l'approche duale adoptée garantit la validité des bornes pendant toute la durée de l'évaluation et donc cette dernière peut être interrompue à tout instant, en particulier lorsque l'élagage d'un nœud est possible. Ces caractéristiques permettent à BiqCrunch d'être très compétitif pour la résolution de nombreux problèmes difficiles de l'optimisation combinatoire tels que max-cut ou k-cluster.

BiqCrunch s'appuie sur deux briques logicielles essentielles : L-BFGS-B [1], implémentant une méthode quasi-Newtonienne robuste et peu consommatrice de mémoire, et une version modifiée de BOB [5], un framework pour développer des branch-and-bounds en environnements séquentiels et parallèles. Cependant, la version actuelle de BiqCrunch n'exploite pas la présence éventuelle de plusieurs coeurs dont sont pourvues les machines actuelles. L'objectif de cette contribution est donc de proposer une version multithread en mémoire partagée, ce qui pour un algorithme de branch-and-bound tel que BiqCrunch peut s'avérer particulièrement rentable, l'obtention précoce de la solution optimale permettant éventuellement d'écourter l'évaluation de certains nœuds dans l'arbre de recherche voire d'en énumérer un nombre inférieur.

2 Approche suivie pour la parallélisation

Le framework BOB adapté et intégré dans BiqCrunch permet d'effectuer plusieurs choix concernant les structures de données, la file d'attente des nœuds à développer, ainsi que plusieurs stratégies (types de parcours, équilibrage,...). Le niveau de granularité de la parallélisation de BiqCrunch choisie ici se situe au niveau des nœuds de l'arbre de recherche. Une fois ces nœuds générés, ils peuvent donc être traités de façon indépendante les uns des autres par un thread différent chacun, une file d'attente partagée permettant de les prioriser.

L'avantage de notre approche réside dans cette indépendance entre les calculs : on évite alors les interactions, et donc les synchronisations, entre les threads. Cependant, cette granularité, relativement grosse, pourrait entraîner un déséquilibrage de charge. En effet, le calcul d'un nœud qui prendrait beaucoup de temps pourrait faire durer le calcul sur un seul thread alors que les autres threads ont terminé tous les calculs disponibles. Cependant, en pratique, ce cas extrême ne survient pas ici, plusieurs mécanismes mis en place dans BiqCrunch garantissant des temps d'évaluation bornés et le plus souvent du même ordre de grandeur.



(a) Exécution des calculs en séquentiel.

(b) Exécution des calculs sur 3 threads.

FIG. 1 – Exécution des calculs, l'optimum étant trouvé pendant l'exécution de T5.

3 Résultats expérimentaux

La Figure 1 présente une exécution possible du calcul en séquentiel (Figure 1a) et en parallèle sur plusieurs threads (Figure 1b). Un premier nœud T0 génère deux nœuds T1 et T2 : ces deux nœuds peuvent être calculés en parallèle sur deux threads dans la version parallèle, mais ils sont calculés l'un après l'autre dans la version séquentielle. Lorsque davantage de nœuds sont générés, on peut augmenter le parallélisme du calcul. Ici, l'optimum est trouvé pour le nœud T5 : dans la version parallèle, il est alors possible d'arrêter les calculs effectués sur les autres threads (les nœuds sont élagués), tandis que dans la version séquentielle, les temps d'évaluation des noeuds T3 et T4 sont plus longs car la meilleure solution courante n'a pas encore été mise à jour (elle ne l'est qu'au moment de l'évaluation de T5).

La Figure 2 présente la comparaison des profils de performance (il s'agit de fonctions de distribution cumulatives : voir [2] pour la définition de ces profils), entre la version séquentielle de BiqCrunch 2.0 et la nouvelle version multi-thread, avec un nombre de threads variant de 2 à 8 (sur une machine 8 cœurs). Ces profils correspondent aux 45 instances du problème k-cluster de taille 100 utilisées en particulier dans [4]. En théorie, le gain relatif, par rapport à BiqCrunch 2.0, est de l'ordre du nombre de threads utilisés, mais ce gain peut parfois être plus important.

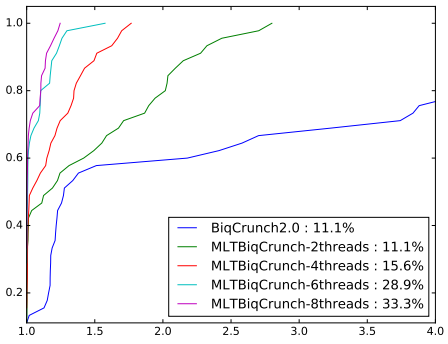


FIG. 2 – Profils de performance

Références

[1] Byrd R. H. ; Nocedal J. et Zhu C. *L-BFGS-B : Algorithm 778 : L-BFGS-B, FORTRAN routines for large scale bound constrained optimization.* ACM Transactions on Mathematical Software, 23(4) : 550-560, 1997.

[2] Dolan, Elizabeth D. et Moré, Jorge J. *Benchmarking optimization software with performance profiles.* In Math. Prog., 91(2) : 201-213, 2002.

[3] Krislock, N. ; Malick, J. et Roupin, F. *BiqCrunch : A semidefinite branch-and-bound method for solving binary quadratic problems.* ACM Transactions on Mathematical Software, 43(4) : n°32, 2016.

[4] Krislock, N. ; Malick, J. et Roupin, F. *Computational results of a semidefinite branch-and-bound algorithm for k-cluster.* Computers and Operations Research, 66 : 153-159, 2016

[5] Le Cun B. ; Roucairol C., et The Pnn Team. *BOB : a Unified Platform for Implementing Branch-and- Bound like Algorithms.* Technical Report. Laboratoire Prism, 1995.

[6] Malick, J. et Roupin, F. *On the bridge between combinatorial optimization and nonlinear optimization : new semidefinite bounds for 0-1 quadratic problems leading to quasi-Newton methods.* In Math. Prog. B, 140(1) : 99-124, 2013.