Pourquoi les Branch-and-Bounds sont des meta-heuristiques

Luc Libralesso[1], Florian Fontan[2], Vincent Jost[1]

[1] Univ. Grenoble Alpes, CNRS, Grenoble INP, G-SCOP, 38000 Grenoble, France
[2] Artelys, 75009 Paris, France
luc.libralesso@grenoble-inp.fr

Mots-clés : *Recherches Arborescentes, Meta-heuristiques, Challenge EURO/ROADEF 2018, Sequential Ordering Problem, framework générique*

Face à un problème d'optimisation combinatoire, deux pistes sont généralement considérées :

— Les méthodes exactes qui permettent d'obtenir une solution optimale au prix d'un temps de calcul potentiellement très long. On y retrouve la Programmation linéaire en nombres entiers, la Programmation par contraintes *etc.* Ce genre de méthode utilise généralement des recherches arborescentes qui mettent l'accent sur des bornes et coupes fortes comme le cutting-plane, le Branch-and-Price, *etc.*

— Les meta-heuristiques qui permettent d'obtenir des solutions de très bonnes qualité en un temps réduit. Celles-ci se focalisent sur des opérateurs rapides (voisinages, mutations, croisement *etc.*) et des stratégies de recherche (relativement) élaborées comme le recuit simulé, la recherche tabou, les algorithmes évolutionnaires/mémétiques.

La plupart du temps, les recherches arborescentes sont utilisées pour des méthodes exactes. Dans ce contexte, *Depth First Search* ou *Best First Search* semblent les plus adaptées, la première pour sa simplicité et consommation mémoire limitée et la deuxième pour prouver l'optimalité plus rapidement (on notera que cette hypothèse est parfois remise en question [8]). En revanche, lorsque la taille de l'arbre est trop grande pour trouver des solutions de bonne qualité, ces méthodes n'obtiennent souvent pas de bons résultats. En effet, *Depth First Search* tend à prendre de mauvaises décisions à la racine de l'arbre et ne peut en sortir. *Best First Search* tend à ne trouver de solutions qu'à la fin de la recherche (on notera que diverses techniques comme le restart ou le probing/diving ont été mises au point pour leur permettre de trouver d'avantage de bonnes solutions rapidement. Mais ces dernières ne suffisent généralement pas pour concurrencer les meta-heuristiques actuelles [1]).

Pour ces raisons, il semble couramment admis que les recherches arborescentes ne sont pas adaptées pour des instances de grande taille (voir citation ci-dessous).

Extrait de Mathematical Programming Solver based on Local Search ([5]) :

> " Tree search approaches like branch-and-bound are in essence designed to prove optimality [...] Moreover, tree search has an exponential behavior which makes it not scalable faced with real-world combinatorial problems inducing millions of binary decisions. "

Il se pourrait qu'il soit pertinent de re-questionner la place des méthodes de recherche arborescentes dans la résolution heuristique des problèmes en RO. En effet, il existe d'autres méthodes de recherche arborescente dans les communautés d'IA planning et CP (Beam Search [9], LDS [6], BULB [4] *etc.*). Tout comme la recherche locale, elles trouvent des solutions rapidement et les améliorent continuellement (jusqu'à prouver l'optimalité ou atteindre une restriction de temps ou mémoire). Dans nos travaux, nous montrons que pour certains problèmes, elles battent les meta-heuristiques classiques. En particulier pour le challenge EURO/ROADEF 2018 et le Sequential Ordering Problem [3] (voyageur de commerce asymétrique avec contraintes de

précédence). Ce dernier a été étudié depuis plus de 30 ans et une grande variété de méthodes ont été proposées pour le résoudre. Il est particulièrement connu pour être très difficile au point que même certaines instances de moins de 50 villes sont toujours ouvertes. Nous avons développé une méthode arborescente simple (environ 200 lignes de code C++) qui utilise une inférence faible à chaque nœud de l'arbre (*i.e.* très rapide à calculer). Cette méthode nous a permis d'améliorer les meilleures solutions connues en quelques secondes sur les plus grandes instances de la SOPLIB.

Cet ensemble de méthodes arborescentes présente un certain nombre de propriétés souhaitables de meta-heuristiques :

— Elles sont faciles à implémenter et peuvent s'adapter à des instances de taille importante.
— produisent de très bonnes solutions rapidement et sont capables de les améliorer si elles ont davantage de temps (dans ce cas, on parle d'algorithme *anytime*)
— Il «suffit» de définir un espace de recherche arborescent par un arbre implicite. On peut ensuite appliquer un ensemble de méthodes génériques pour explorer l'arbre efficacement. Ces méthodes sont donc adaptables à plusieurs problèmes distincts.

Pour ces raisons, nous considérons qu'il pourrait être pertinent de considérer les Branch-and-Bounds (et plus spécifiquement les Branch-and-Bounds anytime) comme des meta-heuristiques explorant un arbre implicite (au même titre que les meta-heuristiques basées sur de la recherche locale explorent un graphe implicite défini par les voisinages). On notera qu'un pas dans cette direction a déjà été fait en montrant que l'optimisation par colonies de fourmi peut être vue comme une recherche arborescente [2, 7]. Aussi, comme en recherche locale (et tout paradigme de résolution générique), il est crucial de soigner les implémentations algorithmiques, les structures de données et l'évaluation incrémentale des contraintes et objectifs.

Dans cette présentation nous discuterons des algorithmes mis en place lors du challenge EURO/ROADEF 2018 et pour le Sequential Ordering Problem. Nous présenterons également un framework générique de recherche arboresecnte que nous développons.

Références

[1] Tobias Achterberg. Scip : solving constraint integer programs. *Mathematical Programming Computation*, 1(1) :1–41, 2009.

[2] Christian Blum. Beam-aco—hybridizing ant colony optimization with beam search : An application to open shop scheduling. *Computers & Operations Research*, 32(6) :1565–1591, 2005.

[3] Laureano F Escudero. An inexact algorithm for the sequential ordering problem. *European Journal of Operational Research*, 37(2) :236–249, 1988.

[4] David Furcy and Sven Koenig. Limited discrepancy beam search. In *IJCAI*, pages 125–131, 2005.

[5] Frédéric Gardi, Thierry Benoist, Julien Darlay, Bertrand Estellon, and Romain Megel. *Mathematical programming solver based on local search*. Wiley Online Library, 2014.

[6] William D Harvey and Matthew L Ginsberg. Limited discrepancy search. In *IJCAI (1)*, pages 607–615, 1995.

[7] Vittorio Maniezzo. Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem. *INFORMS journal on computing*, 11(4) :358–369, 1999.

[8] Lei Shang, Vincent T'Kindt, and Federico Della Croce. The memorization paradigm : Branch & memorize algorithms for the efficient solution of sequencing problems. 2018.

[9] Weixiong Zhang. Complete anytime beam search. In *AAAI/IAAI*, pages 425–430, 1998.