Solving the Sequential Ordering Problem with anytime tree search

Luc Libralesso[1], Abdel-Malik Bouhassoun[1], Hadrien Cambazard[1], and Vincent Jost[1]

Univ. Grenoble Alpes, CNRS, Grenoble INP, G-SCOP, 38000 Grenoble, France
luc.libralesso@grenoble-inp.fr

Abstract. We study several generic tree search techniques applied to the *Sequential Ordering Problem*. This study enables us to propose a simple yet competitive tree search. It consists of an iterative beam search that favors search over inference and integrates prunings that are inspired by dynamic programming. The resulting method proves optimality on half of the SOPLIB instances, 10 to 100 times faster than other existing methods. Furthermore, it finds new best-known solutions on 6 among 7 open instances of the benchmark in a small amount of time. These results highlight that there is a category of problems (containing at least SOP) where an anytime tree search is extremely efficient (compared to classical meta-heuristics) but was underestimated.

Keywords: Tree Search · Beam Search · Dynamic Programming · Sequential Ordering Problem

While facing a hard combinatorial optimization problem, two types of solutions are usually considered:

- *Exact methods* that allow to find optimal solutions at the price of a potentially very long computation time. In this category, we find Mixed Integer Programming methods, Constraint Programming *etc.* This kind of methods generally use tree search techniques that usually rely on strong bounds and cuts like the cutting-plane algorithm or the branch-and-price algorithm *etc.*
- *Meta-heuristics* that allow to find near-optimal solutions. These methods usually rely on fast operators (neighbourhoods, crossovers, mutations, *etc.*) and various search strategies (Simulated Annealing, Tabu Search, Evolutionary Algorithms, Ant Colony Optimization).

Tree search is mainly found in exact methods. In this specific context, *Depth First Search* or *Best First Search* seem to be the most suited methods. The first one for its simplicity, limited memory usage and backtrack-friendliness. The second for its ability to improve quickly bounds and prove optimality in a (relatively) small amount of nodes[1]. However, on large instances, these methods do not usually obtain good quality solutions. Indeed, *Depth First Search* is prone

[1] We note that this hypothesis is sometimes questioned [4]

to chose a bad branch early in the tree search and is not able to escape from it. *Best First Search* is prone to find solutions late in the search (thus not finding any solutions within hours of computation). We also note that work have been done to allow these methods to find good solutions fast (using a restarting strategy for *Depth First Search* and diving for *Best First Search*). But these improvements are usually insufficient to compete with classical meta-heuristics.

A common opinion regarding tree-search is well summarized in [2]:

"*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.*"

It may be interesting to re-evaluate the inefficiency of tree search to solve large-scale/industrial combinatorial optimization problems. Indeed, there are other tree search techniques originally proposed in the sixties in AI or planning conferences. To cite a few, we find *(Complete Anytime) Beam Search* [5], well-known for its success to solve scheduling or packing problems, or, *Limited Discrepancy Search*, used intensively in Constraint Programming solvers. Tree Search can be used to find solutions quickly and continuously improve them similarly to the behavior of classical meta-heuristics (until they reach a stopping criterion or prove optimality by depleting the search tree). We qualify search methods with this property as *Anytime algorithms* (in our case *Anytime Tree Search*).

In this presentation, we discuss a simple anytime tree search algorithm for the *Sequential Ordering Problem* [1] (Asymmetrical Traveling Salesman Problem with precedence constraints). This problem have been studied intensively during the last 30 years and a large variety of methods have been developed to solve it. It is especially well-known because it has instances with less than 50 cities that are still open. We propose a simple anytime tree search algorithm (approximately 200 lines of C++ code) that results from a study over many Branch-and-Bound algorithmic components. This method was able to improve best-known solutions on 6 over 7 open instances from the SOPLIB [3].

References

1. Laureano F Escudero. An inexact algorithm for the sequential ordering problem. *European Journal of Operational Research*, 37(2):236–249, 1988.
2. Frédéric Gardi, Thierry Benoist, Julien Darlay, Bertrand Estellon, and Romain Megel. *Mathematical programming solver based on local search.* Wiley Online Library, 2014.
3. Luc Libralesso, Abdel-Malik Bouhassoun, Hadrien Cambazard, and Vincent Jost. Tree searches for the Sequential Ordering Problem. working paper or preprint, January 2020.
4. Lei Shang, Vincent T'Kindt, and Federico Della Croce. The memorization paradigm: Branch & memorize algorithms for the efficient solution of sequencing problems. *preprint*, 2018.
5. Weixiong Zhang. Complete anytime beam search. In *AAAI/IAAI*, pages 425–430, 1998.