A Tree Search Method for the ROADEF/EURO Challenge 2018: Cutting Optimization Problem

Jialu Zhang, Dong Liu, Yuan Fang (Team J29)
SMART, School of Computer Science and Technology, Huazhong University of Science and Technology, China.

1 Introduction

The cutting optimization problem can be considered as a strip packing problem with extra constraints. We try to solve the problem with the general tree search method, and several pruning strategies are used to improve the efficiency of the algorithm.

2 Algorithm Description

2.1 Search method

Our algorithm uses a tree search method and the main two parts are branching and pruning. We estimate the upper bound of the used area for every branch and explore the branch with a better upper bound firstly. Furthermore, when exploring a tree node, we estimate its lower bound to decide whether to branch it or not. At the same time, we consider the handling of special items in the algorithm, such as how to place the same shape items and oversized items. Our algorithm will continuously explore the search tree until the preset time is exceeded.

2.2 Branching

We construct a complete feasible solution by selecting one of the remaining items and putting it into the plate each time. For each non-leaf nodes, we consider all the items on the top of the batch stack and its possible rotation state to produce possible branches. There are some rules for branching. First, the item should be placed next to the cut or the edge of defect. Second, the waste area and cut width/height should satisfy the corresponding constraints. Third, item should be placed from bottom left to top right between two 1-cut.

2.3 Estimate the lower bound

A tighter lower bound is important for pruning. we use Gurobi 8.0 to optimize the linear relaxation to a mixed integer programming model to get the approximate lower bound, which is the minimal area of the waste. We will randomly estimate the lower bound on the tree nodes and compare it with the best objective to decide whether prune or not.

2.4 Estimate the upper bound

For each partial solution, there is an approximate upper bound. We use a greedy strategy to construct a poor feasible solution, and take the obtained objective function value as the upper bound. The upper bound will be estimated for every branch node and the node with the best upper bound will be considered first. Since the upper bound is only a guide to the search branch order, we use a greedy strategy to quickly estimate it without deliberately requiring its accuracy.

3 Computational Environment

The required environment for our executable program is Ubuntu 18.04.1 operation system with g++ and Gurobi 8.0 is necessary.

Our experiment result was tested on a virtual machine with Ubuntu 18.04.1 and the hardware is Intel Core i3-3240 CPU (3.40GHZ) with 4GB RAM.

Reference

1. Dusberger, F. (2015). Solving the 3-Staged 2-Dimensional Cutting Stock Problem by Dynamic Programming and Variable Neighborhood Search, 47, 133–140.
2. Lodi, A., Monaci, M., & Pietrobuoni, E. (2017). Partial enumeration algorithms for Two-Dimensional Bin Packing Problem with guillotine constraints. Discrete Applied Mathematics, 217, 40–47.