INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

Advanced Mathematical Programmation

Branch-and-Price for the Bin-Packing Problem

March 31, 2020

Théo Guyard
theo.guyard@insa-rennes.fr
4GM, Mathematics Department, INSA Rennes

Contents

1 Introduction

1.1 Bin-Packing Problem

In this report, we focus on the Bin-Packing Problem. It consists of putting objects with a given size in the minimum number of bins with the same capacity. Consider that set if items is given by $I = \{1, \dots, N\}$, their size by $S = \{s_1, \dots, s_n\}$ and B bins with a capacity C are available. The number of bins is supposed large enough to store all the n objects. Thus, the Bin-Packing Problem can be written under the following form :

$$\begin{cases} \min & \displaystyle\sum_{b=1}^{B} y_b & & \\ \text{s.t.} & \displaystyle\sum_{i=1}^{N} s_i x_{ij} \leq C y_b & \forall\ b = 1, \dots, B & \\ & \displaystyle\sum_{b=1}^{N} x_{ib} = 1 & \forall\ i = 1, \dots, N & \\ & x_{ib} \in \{0,1\} & \forall\ i = 1, \dots, N & \forall\ b = 1, \dots, B \\ & y_b \in \{0,1\} & \forall\ b = 1, \dots, B & \end{cases} \tag{$\mathbb{BPP}$}$$

Where $x_{ib} = 1$ if the item i is packed in the bin number b ($x_{ib} = 0$ otherwise) and $y_b = 1$ if the bin number b is non-empty ($y_b = 0$ otherwise). The first constraint ensure that the bin capacity is not exceeded and the constraint number two ensure that each item is packed in a bin. This problem is known to be NP-hard.

1.2 Branch-and-Price approach

To address the NP-hardness of the problem, we can solve this problem with a Branch-and-Price algorithm.

1.2.1 Set covering formulation

We choose an other formulation of the ($\mathbb{BPP}$). Let $\mathcal{P}$ the set of all combination of items in I such that their cumulative size doesn't exceed the bin capacity C. Each element of $\mathcal{P}$ is called a pattern. Let $P = |\mathcal{P}|$, we have

$$\mathcal{P} = \left\{ \mathbf{x} \in \{0,1\}^{|N|}, \quad \sum_{i=1}^{N} x_i s_i \leq C \right\} = \left\{ \mathbf{x} = \sum_{p=1}^{P} \alpha^p \bar{x}^p, \quad \sum_{p=1}^{P} \alpha^p = 1, \alpha \in \{0,1\}^{|P|} \right\}$$

Here, $\bar{x}^p$ is a pattern and $\bar{x}^p_i = 1$ if the item i is used in the pattern p. Using this Dantzig reformulation of the set $\mathcal{P}$, we can write ($\mathbb{BPP}$) under its set covering formulation :

$$\begin{cases} \min & \displaystyle\sum_{p=1}^{P} \alpha^p & & \\ \text{s.t.} & \displaystyle\sum_{p=1}^{P} x_i^p \alpha^p = 1 & \forall\ i = 1, \dots, N & \\ & \alpha_p \in \{0,1\} & \forall\ p = 1, \dots, P & \end{cases} \tag{$\mathbb{SCBPP}$}$$

For this formulation, $x_i^p = 1$ if the item i is within the pattern p and $\alpha^p = 1$ is the pattern p is used in the solution. Thus, the solution is a set of patterns and each patter correspond to a bin. The constraint ensure that each item is packed in a pattern.

On thing to notice is that $|P|$ (the number of variables) is huge. At most, $|P|$ is the number of combination available with the N items, namely $\sum_{n=1}^{P} \binom{P}{n}$. Even for small instances, the number of variable is not trackable. Furthermore, it is needed to enumerate and store all the patterns possible in order to solve the problem which could involve a huge amount of memory.

1.2.2 Restricted master problem

To handle this problem, we introduce a new set $\mathcal{P}' \subset \mathcal{P}$ or cardinal P' containing only a fraction of the available patterns. We can solve ($\mathbb{SCBPP}$) on this restricted number of pattern, which is more tracktable. This leads to a sub-optimal bound. This new problem is called the restricted master problem :

$$\begin{cases} \min & \displaystyle\sum_{p=1}^{P'} \alpha^p & \\ \text{s.t.} & \displaystyle\sum_{p=1}^{P'} x_i^p \alpha^p = 1 & \forall\ i = 1, \dots, N \\ & \displaystyle\sum_{p=1}^{P'} \alpha^p \leq B & \\ & \alpha_p \in [0,1] & \forall\ p = 1, \dots, P' \end{cases} \tag{$\mathbb{RMBPP}$}$$

If the set $\mathcal{P}'$ is well chosen and contained the optimal patterns, then the optimal solution of ($\mathbb{RMBPP}$) will be the same as the optimal solution of ($\mathbb{SCBPP}$) (and also the solution of ($\mathbb{BPP}$)). The problem is to choose the rights patterns to include in $\mathcal{P}'$. As $\mathcal{P}'$ will also have a big cardinal, we rather solve a relaxation of the master problem and that is why we have that $\alpha_p \in [0,1]\ \forall\ p = 1, \dots, P'$. We also add a constraint on the number of pattern allowed as we know that there are only B bins available. Solving the relaxation will lead to an supra-optimal solution for the problem. We will see later how to obtain an integer solution in order to have the same solution as the optimal solution of ($\mathbb{BPP}$).

1.2.3 Subproblem

We can write an other optimization problem called the subproblem (or priceing problem) which will generate "interesting" patterns to include in ($\mathbb{RMBPP}$). If we note π the optimal dual variables corresponding to the constraint of ($\mathbb{RMBPP}$), this pricing problem takes the following form :

$$\begin{cases} \min & 1 - \displaystyle\sum_{i=1}^{N} \pi_i y_i & \\ \text{s.t.} & \displaystyle\sum_{i=1}^{N} y_i s_i \leq C & \\ & y_i \in \{0,1\} & \forall\ i = 1, \dots, N \end{cases} \tag{$\mathbb{SPBPP}$}$$

The solution of the pricing problem is a feasible pattern (*i.e.*, which doesn't exceed the bin capacity). The optimal cost is called the reduced cost. We can see that usually, the cost of a pattern is 1 (one pattern used correspond to one bin used) but here, the cost is penalized by a term containing π. This second term will penalize patterns which can not improve the set $\mathcal{P}'$ in the sense that if a pattern p has a high reduced cost, the solution of ($\mathbb{RMBPP}$) with $\mathcal{P}'$ or with $\mathcal{P}' \cup p$ is the same. In fact, if ${z_{sp}}^\star$ denotes the optimal cost of ($\mathbb{SPBPP}$) for a given π, the the optimal patter $\mathbf{y}^\star$ can improve the solution of ($\mathbb{RMBPP}$) if and only if ${z_{sp}}^\star < 0$.

On thing very important to notice is that this pricing problem is equivalent to a knapsack problem with weights π, sizes s and capacity C. A knapsack problem is easy to solve. The key mechanism of the Branch-and-Price algorithm will be to solve sequentially ($\mathbb{RMBPP}$) and ($\mathbb{SPBPP}$) until a solution ${z_{sp}}^\star \geq 0$ is obtained. At this stage, ($\mathbb{RMBPP}$) will be solved at optimality.

1.2.4 Branching

Know, let's remember that ($\mathbb{RMBPP}$) is a relaxation of ($\mathbb{SCBPP}$) so it gives a supra-optimal solution. In order to have the optimal integer solution, we will introduce branching rules (new constraints) to break the functionality of the variables. Let's consider a tree where each node correspond to a problem with its local branching rules. When a node is solved to optimality (*i.e.*, ($\mathbb{RMBPP}$) and ($\mathbb{SPBPP}$) are solved sequentially until optimality), we will select a fractional variables to branch on. Then, child node will be created under the node just solved with new constraints in order to have a non-fractional variable for the rest of the branch. The tree will be explored node after

node until all the nodes are explored. The key idea will be to cut non-improving branches during the explorations so as to explore only a part of the tree.

We can not branch only on the fractional α^p. Instead, we will select a pair (i, j) of items such that the variable

$$w_{ij} = \sum_{p \in \mathcal{P}' | x_i^p = x_j^p = 1} \alpha^p$$

is fractional. In one branch, we will impose that $w_{ij} \geq 1$ (items i and j always together) and in the other branch, we will impose that $w_{ij} \leq 0$ (items i and j always separated).

1.3 Resolution method

To solve the Bin-Packing Problem, we will use two different branching rules. The first one was proposed by Ryan & Foster in 1981 [RF81]. This branching rule allow to keep a single subproblem at each node but doesn't preserve the knapsack structure of the problem. The subproblems will rather be knapsack with conflicts problems, harder to solve than the usual knapsack problem. The second branching rule which will be used is a generic branching scheme introduced by Vance in 1994 [Van+94]. This branching rule preserve the knapsack structure of the subproblems but at each node, multiple subproblems will have to be solved.

For the Ryan & Foster branching rule, two method will be proposed for the subproblem resolution. The first one is to use a classical solver so as to model and solve the subproblem with the new branching constraints. We will also see how to solve the subproblems with dynamic programming [Tot80]. For the Generic branching rule, we will also propose to solve the subproblem either with a solver or with a dynamic programming algorithm [SV13].

In addition, we will see how to construct an heuristic solution before the tree exploration in order to set better initial bounds. Three variations of a decreasing-size-order packing algorithm will be proposed [BHB09].

2 Ryan-Foster branching rule

2.1 Node processing

2.2 Subproblem resolution

3 Generic branching constraints

3.1 Node processing

3.2 Subproblem resolution

4 Heuristics

4.1 Root heuristics

5 Numerical applications

5.1 Implementation detail

5.2 Numerical results

5.3 Result analysis

References

[Tot80] Paolo Toth. "Dynamic programming algorithms for the zero-one knapsack problem". In: *Computing* 25.1 (1980), pp. 29–45.

[RF81] DM Ryan and EA Foster. "Rn integer programming approach to scheduling". In: (1981).

[Van+94] Pamela H Vance et al. "Solving binary cutting stock problems by column generation and branch-and-bound". In: *Computational optimization and applications* 3.2 (1994), pp. 111–130.

[BHB09] AK Bhatia, M Hazra, and SK Basu. "Better-fit heuristic for one-dimensional bin-packing problem". In: *2009 IEEE International Advance Computing Conference*. IEEE. 2009, pp. 193–196.

[SV13] Ruslan Sadykov and Fran**c**cois Vanderbeck. "Bin packing with conflicts: a generic branch-and-price algorithm". In: *INFORMS Journal on Computing* 25.2 (2013), pp. 244–255.