Discussion about Constraint Programming Bin Packing Models

Jean-Charles Régin and Mohamed Rezgui
Université de Nice-Sophia Antipolis, I3S CNRS
2000, route des Lucioles - Les Algorithmes - BP 121
06903 Sophia Antipolis Cedex - France
jcregin@gmail.com; mohamed.rezgui@etu.unice.fr

Abstract

The bin packing problem is one of the core problems of cloud computing management. It corresponds to the problem of assigning virtual machines to servers. However, cloud computing also imposes a huge variety of constraints on this problem and most of them can not be expressed a priori. Constraint Programming (CP) has been proved efficient for solving bin packing instances and for its capability to deal with unexpected constraints. Nevertheless, CP approaches have never been tested on large scale instances coming from cloud computing. In this paper, we describe some CP models for solving bin packing instances and we expect that some experiments could identify precisely the advantages and the drawbacks of these models for solving real life instances.

Introduction

Cloud computing is an emerging trend bringing together various kinds of virtualization technologies to offer on-demand computing resources. There is widespread consensus that the Future Internet will be heavily based on some kind of successful Cloud technology. However, to master the deployment of Cloud-based infrastructures, some hard scientific problems need to be properly addressed first.

Several problems of the cloud computing management are configuration problems where group of objects must be assigned to containers. The objects may be virtual machines and the containers may be servers, but the objects can also be the servers and the containers are power units.

These configuration problems can be seen as bin packing problems, which are defined as follows in their classical version: objects of different capacities must be packed into a finite number of bins of capacity C in a way that minimizes the number of bins used. However, the problem is more general than the classical bin packing problem: minimizing the number of used servers costs less energy but several side constraints must also be considered (e.g., agility, reliability, sustainability). Computing optimized deployment plans requires to solve a hard planning problem and that plan may be subject to the addition of side constraints that cannot be defined in advance. Therefore, a robust method is required.

Constraint Programming is such a method having the capabilities to be easily adapted for considering new constraints. In addition, several recent studies have shown that this method is quite competitive for solving bin packing instances [10, 8].

Unfortunately, the proposed methods have two main drawbacks: there are mainly focused on the resolution of the pure version of the bin packing (all the bins are equivalent and we want to minimize the number of used bins and there is only one dimension) and only few specific instances have been considered. Notably, the size of the considered instance are two or three orders of magnitude smaller than the cloud computing based instances. A well known internet company has data center with 20,000 servers and millions of virtual machines.

In order to obtain a more robust and more general method capable to deal with large scale problems, we need first to clearly identify the advantages and the drawbacks of the current constraint programming models. Mainly, we need to identify what parts of the model are really important and what other parts are secondary. Then, we would like to study the scalability of the current models and identify the current limits. Therefore, we propose to consider all existing CP models in order to answer to these questions.

First, we will discuss about the instances that are available in the literature and present more pertinent ones. Then, we will recall the basic model. Next, we will show that we can consider two methods for reaching the optimality (and proving it). We will also present some lower bounds that have been proposed. The classical strategies (first fit decreasing, best fit decreasing, ...) for assigning items will be detailed. After, we will investigate the dominance rules and the subset sum constraint (also called knapsack constraints). In addition, two other models will be introduced. At last, we will propose to introduce additional constraints.

Instances

The quality of the instances we consider are quite important for deriving some general conclusions. Unfortunately it appears that most of the instances proposed in the literature have some major drawbacks. Gent [3] criticized the well known Falkenauer's benchmarks [2]. He closed five benchmark problems left open by cpu-intensive methods using a genetic algorithm and an exhaustive search method by using a very simple heuristic methods requiring only seconds of cpu time and hours of manual work. He questioned the underlying hardness of test data sets. One reason of the problem is the kind of data sets. Most of the time, the bins are filled in with only 3 items or less.

Unfortunately, the same kind of data sets have been proposed by others [9, 6, 4]. Korf explicitly considered triplets instances.

We think it is quite important to be compared to other existing works, therefore we propose to use these benchmarks that we can mainly found on the web page of A. Scholl. However, if it is certainly interesting to consider such hard instances, we should also be careful, as noted by I. Gent, and we should use some other data sets.

We propose to use two other kinds of data:

• First we use the data defined for the multidimensional knapsack problems [1]. The advantage is that the capacities of the items are representative of real life instances, and we can have several items per bins. Since the data have been defined for a knapsack problem, we just need to repeat or to mix several data for obtaining bin packing instances.

• Then, we propose to generate instances in regards to the criterion number of items per bin. The advantage of this approach is that we can study the scalability of some models. The drawback is that it is difficult to avoid having some randomness.

Basic model

Let I be the set of items and B be the set of bins. Each item k is associated with the capacity ci_k and each bin j is associated with the capacity cb_j. In the basic model membership variables are used, that is for each item i and each bin j, a 0-1 variable x_{ij} is defined. If the variable is equal to 1 then it means that item i is assigned to bin j, else the variable is equal to 0. The variable y_j is a 0-1 variables stating that the bin j has been used. The problem becomes:

$$\min \sum_{j=1}^{m} y_j$$

$$\forall j \sum_{i=1}^{n} ci_i x_{ij} \leq cb_j y_j \tag{1}$$

$$\forall i \sum_{j=1}^{m} x_{ij} = 1 \tag{2}$$

$$x \in \{0,1\}, y \in \{0,1\}$$

Constraint (1) ensures that the capacity of a bin is not exceeded by the sum of the capacities of the items put in that bin. Constraint (2) states that an item is put in exactly one bin.

Slack Variables

Some slack variables may be introduced in order to add some implicit constraints. Constraint (1) can be refined in

$$\forall j \sum_{i=1}^{m} ci_i x_{ij} + s_j = cb_j y_j$$

where s_j is a variable expressing the capacity of bin j that has not been used. In this case we can introduce the following constraint:

$$\sum_{i=1}^{n} ci_i + \sum_{j=1}^{m} s_j = \sum_{j=1}^{m} cb_j$$

The advantage of this approach is that the first and the last part of this constraint are constant.

Increasing or Decreasing?

In order to find the minimal number of bins we have two solutions:

• either we start with a convenient number of bins and we try to reduce this number each time we find a solution. That is, we successively decrease the number of required bins until there is no more solution.

• or we compute a lower bound on the number of bins, for instance by applying a greedy algorithm which breaks items into several parts. Then, we increase this number while there is no solution. The first found solution is the optimal one.

Lower Bounds and Optimality

First, it could be interesting to measure the cost of reaching the optimality and the cost of proving the optimality. We could also consider integrating or not the classical lower bounds proposed by Martello [6] or by Labbé [5]. We do not speak about their methods here but simply about their lower bounds $\mathcal{L}_2$ and $\mathcal{L}_3$.

Strategies

Well known strategies have been developed for bin packing. Notably:

- first fit decreasing,
- best fit decreasing
- worst fit decreasing.

For all these strategies, the items are fit into bins by considering them by non increasing capacities. For the first decreasing strategy, the first bin in which an item can be put is selected. For the best fit decreasing strategy, we select the bin such that the remaining capacity after setting the item is minimal. For the worst fit decreasing strategy, we select the bin leaving the largest capacity after setting the item.

Symmetry breaking

Symmetries are quite frequent in bin packing problems. Several symmetry methods or dominance rules must be considered. Note that we should be really careful about the combinations of these rules. Some of them cannot be easily combined. There are all based on the interchangeability of objects. Thus, we have to ensure that the object are effectively interchangeable before applying one of these rules. Particularly we have to check that no other rule currently applied could interfere with the application of the rule we consider. These rules have been proposed by [10, 8, 4]:

• Items having the same capacity are interchangeable. Therefore, we should state that if item i and j have the same capacity and $i < j$ then the assigned bin of item i is less than or equal to the assigned bin of item j.

• If there is no constraint between bins and if a bin j has a capacity greater than a bin k then the bin j should be more fill than bin k

• If an item can be fit into several interchangeable bins then we fit it into the first one and we do not try the other interchangeable bins for this item.

• If a bin can contain only one more item, we fit into that bin the item having the largest capacity.

• If an item can totally fill a bin then we fit this item into that bin and we do not try any other bin for this item.

We have to be careful about the fact that the additional constraints of real life instances may prevent from the application of these rules.

Sum constraint

The difficulty of the bin packing is twofold: we need to solve a simple packing problem, in other words a knapsack problem with unit cost also named subset sum problem, and we need to repeat that resolution for a set of bins in order to minimize the number of used bins. Therefore, we can either focus our attention on the independent subset sum problems or on the combinations of bins.

Several filtering algorithms have been proposed for the subset sum problem:

• the classical one establishing bound consistency

• the algorithm of M. Trick which establishes arc consistency but with a pseudo polynomial complexity [11]

• the algorithm described by JC. Régin and based on a reformulation [7]

On the other hand there are two other works that have been carried out:

• P. Shaw designed an interesting filtering algorithm [10]. It is difficult to characterize but it seems to be worthwhile.

• P. Schaus proposed another algorithm based on the preemption relaxation of the underlined assignment problem of the bin packing [8].

All these algorithms deserve to be compared.

Capacitated Variables

In the basic model, we deal with membership variables. With this model it is quite important to break some symmetries because items having the same capacity are interchangeable. In addition, the drawback of this model is that it is difficult to avoid a quadratic memory complexity. If we have n items and m bins then we need nm variables. With capacitated variables we expect to overcome this difficultly. The idea is to gather the items having the same capacity. Then instead of membership variables, we will have variables counting the number of times an item of a given capacity is taken. So, if we have p items having different capacities then we need to have only pm variables. Moreover, we avoid the interchangeability problem between items having the same capacity.

Subset Sum Variables

For each bin we propose to generate all the possible combinations of the subset sum constraint associated with the bin (i.e. Constraint (1) of the basic model). Then, in a solution a bin takes only one of these combinations. Therefore, we propose to use for each bin a variable whose domain is one of the solutions of the subset sum constraint.

Introduction of additional constraints

In order to simulate some problems arising in real life we propose to introduce some constraints perturbing some models and notably some symmetry breaking rules:

• we restrict the assignment of some items to some bins.

• we impose that some items must be fit into the same bins than some other ones

• we impose that some items cannot be fit into the same bin than some others

Conclusion

We have presented the most well known CP models for solving bin packing problems. We expect that some experiments could give us a better understanding of the strengths and the weaknesses of each CP model used for solving large scale bin packing instances.

References

[1] *SAC94 Suite: Collection of Multiple Knapsack Problems*, 1994.

[2] E. Falkenauer. A hybrid grouping genetic algorithm for bin packing. *Journal of Heuristics*, 2:5–30, 1996.

[3] I. Gent. Heuristic solution of open bin packing problems. *Journal of Heuristics*, 3:299–304, 1998.

[4] R. Korf. An improved algorithm for optimal bin packing. In *Proc. IJCAI-03*, pages 1252–1258. 2003.

[5] M. Labbé, G. Laporte, and S. Martello. Upper bounds and algorithms for the maximum cardinality bin packing problem. *European Journal of Operational Research*, 149(3):490–498, 2003.

[6] S. Martello and P. Toth. *Knapsack Problems*. John Wiley and Sons Inc, 1990.

[7] J-C. Régin. *Hybrid Optimization - The 10 years of CPAIOR*, chapter Global Constraints: a survey. Springer, 2011.

[8] P. Schaus. *Solving Balancing and Bin-Packing problems with Constraint Programming*. PhD thesis, Université catholique de Louvain-la-Neuve, 2009.

[9] A. Scholl, R. Klein, and C. Jürgens. Bison: a fast hybrid procedure for exactly solving the one-dimensional bin packing problem. *Computers & Operations Research*, 24:627–645, 1997.

[10] P. Shaw. A constraint for bin packing. In *CP'04*, pages 648–662, 2004.

[11] M. Trick. A dynamic programming approach for consistency and propagation for knapsack constraints. *Annals of Operations Research*, 118:73 – 84, 2003.