Consistency Check for the Bin Packing Constraint Revisited

Julien Dupuis[1], Pierre Schaus[2], and Yves Deville[1]

[1] Department of Computer Science and Engineering, UCLouvain, Belgium
{julien.dupuis,yves.deville}@uclouvain.be
[2] Dynadec Europe, Belgium
pschaus@dynadec.com

1 Introduction

The bin packing problem (BP) consists in finding the minimum number of bins necessary to pack a set of items so that the total size of the items in each bin does not exceed the bin capacity C. The bin capacity is common for all the bins.

This problem can be solved in Constraint Programming (CP) by introducing one placement variable x_i for each item and one load variable l_j for each bin.

The $\texttt{Pack}([x_1,\ldots,x_n],[w_1,\ldots,w_n],[l_1,\ldots,l_m])$ constraint introduced by Shaw [1] links the placement variables $x_1,\ldots,x_n$ of n items having weights $w_1,\ldots,w_n$ with the load variables of m bins $l_1,\ldots,l_m$ with domains $\{0,\ldots,C\}$. More precisely the constraint ensures that $\forall j \in \{1,\ldots,m\} : l_j = \sum_{i=1}^{n}(x_i = j)\cdot w_i$ where $x_i = j$ is reified to 1 if the equality holds and to 0 otherwise. The Pack constraint was successfully used in several applications.

In addition to the decomposition constraints $\forall j \in \{1,\ldots,m\} : l_j = \sum_{i=1}^{n}(x_i = j)\cdot w_i$ and the redundant constraint $\sum_{i=1}^{n} w_i = \sum_{j=1}^{n} l_j$, Shaw introduced:

1. a filtering algorithm based on a knapsack reasoning inside each bin, and
2. a failure detection algorithm based on a reduction of the partial solution to a bin packing problem.

This work focuses on improvements of the failure detection algorithm.

2 Reductions to Bin Packing Problems

Shaw describes in [1] a fast failure detection procedure for the Pack constraint using a bin packing lower bound (BPLB). The idea is to reduce the current partial solution (i.e. where some items are already assigned to a bin) of the Pack constraint to a bin packing problem. Then a failure is detected if the BPLB is larger than the number of available bins m.

We propose two new reductions of a partial solution to a bin packing problem. The first one can in some cases dominate Shaw's reduction and the second one theoretically dominates the other two.

Paul Shaw's reduction: R0. Shaw's reduction consists in creating a bin packing problem with the following characteristics. The bin capacity is the largest upper bound of the load variables, *i.e.* $c = \max_{j \in \{1,\dots,m\}}(l_j^{\max})$. All items that are not packed in the constraint are part of the items of the reduced problem. Furthermore, for each bin, a virtual item is added to the reduced problem to reflect (1) the upper bound dissimilarities of the load variables and (2) the already packed items. More precisely, the size of the virtual item for a bin j is $(c - l_j^{\max} + \sum_{\{i|x_i=j\}} w_i)$, that is the bin capacity c reduced by the actual capacity of the bin in the constraint plus the total size of the already packed items in this bin. An example is shown in Figure 1(b).

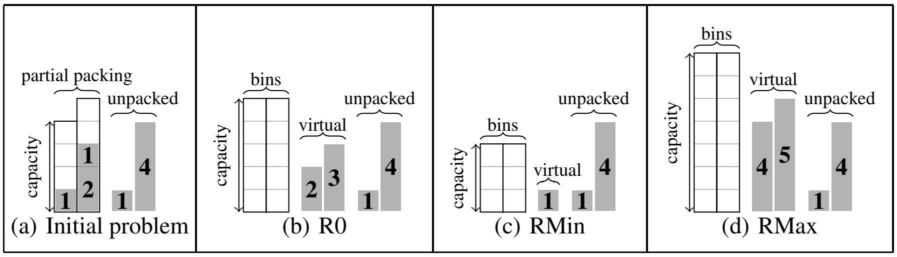

Fig. 1. Example of the three reductions for the bin packing problem

RMin. We introduce RMin that is obtained from R0 by reducing the capacity of the bins and the size of all the virtual items by the size of the smallest virtual item. The virtual items have a size of $(c - l_j^{\max} + \sum_{\{i|x_i=j\}} w_i - min_k(c - l_k^{\max} + \sum_{\{i|x_i=k\}} w_i))$. This reduction is illustrated in Figure 1(c).

RMax. We propose RMax that consists in increasing the capacity and the size of the virtual items by a common quantity, so that, when distributing the items with a bin packing algorithm, it is guaranteed that each virtual item will occupy a different bin. In order to achieve this, each virtual item's size must be larger than half the bin capacity.

In R0, let p be the size of the smallest virtual item, and c the capacity of the bins. The size of the virtual items and the capacity must be increased by $(c - 2p + 1)$. The smallest virtual item will have a size of $s = (c - p - 1)$ and the capacity of the bins will be $(2c - 2p + 1) = 2s - 1$. As one can observe, the smallest virtual item is larger than the half of the capacity. If $c = 2p - 1$, this reduction is equivalent to Shaw's reduction. Note that if $c < 2p - 1$, the capacity and the virtual items will be reduced.

The virtual items have a size of $(2c - 2p + 1 - l_j^{\max} + \sum_{\{i|x_i=j\}} w_i)$. This reduction is illustrated in Figure 1(d).

Generic reduction: Rδ. All these reductions are particular cases of a generic reduction (Rδ) which, based on R0, consists in adding a positive or negative delta (δ) to the capacity and to all the virtual items' sizes.

For R0, $\delta = 0$. For RMin, δ is the minimum possible value that keeps all sizes positive. A smaller δ would create an inconsistency, as the smallest virtual item would have a negative size. δ_{RMin} is always negative or equal to zero. For RMax, δ is the smallest value guaranteeing that virtual items cannot pile up. Note that in some cases, δ_{RMin} or δ_{RMax} can be zero. Also note that δ_{R0} can be larger than the others.

3 Theoretical Comparison of the Three Reductions

Definition 1 (Dominate). *Let A and B be two reductions of the `Pack` constraint to bin packing. We say that A* dominates *B if, for any instance of the `Pack` constraint, the number of bins required in A is larger than the number of bins required in B.*

Theorem 1. *Rδ is a relaxation of the problem of testing the consistency of the `Pack` constraint.*

Proof. If a partial solution of the `Pack` constraint can be extended to a solution with every item placed, then Rδ also has a solution: if each virtual item is placed in its initial bin, then the free space of each bin is equal to its free space in the partial solution, and so all the unplaced items can be placed in the same bin as in the extended solution from the partial assignment.

Theorem 2. *R0 does not dominate RMin and RMin does not dominate R0.*

Proof. Consider the partial packing $\{4, 2\}$ of two bins of capacity 6, and the unpacked items $\{3, 3\}$. R0 only needs two bins, where RMin needs three bins.

Now consider the partial packing $\{2, 3, 1\}$ of three bins of capacity 4, and the unpacked items $\{3, 3\}$. In this case, R0 needs four bins, where RMin only needs three bins.

Theorem 3. *RMax is equivalent to testing the consistency of the `Pack` constraint*

Proof. By Theorem 1, RMax is a relaxation of the partial solution of the BP problem. There remains to show that if there is a solution for RMax, then the partial solution can be extended to a complete solution of the `Pack` constraint. Let's call v the bin from which the virtual item v is from. It is guaranteed by the size of the virtual items that they will each be placed in a different bin b_v. The remaining space in each bin b_v corresponds to the free space in bin v in the original problem. An extended solution of the `Pack` constraint is obtained by packing in v all items packed in b_v.

Corollary 1. *RMax dominates R0 and RMin.*

From a theroretical standpoint, the RMax reduction is always better or equivalent to R0, RMin, and any other instance of Rδ. In practice, though, this is not always the case, as it is shown in the next section.

4 Experimental Comparison

The failure test of Shaw [1] uses the bin packing lower bound $\mathcal{L}_2$ of Martello and Toth [2] that can be computed in linear time. Recently the lower bound $\mathcal{L}_3$ of Labbé [3] has been proved [4] to be always larger than or equal to $\mathcal{L}_2$ and to benefit from a better worst case asymptotic performance ratio ($3/4$ for $\mathcal{L}_3$ [4] and $2/3$ for $\mathcal{L}_2$ [2]), while still having a linear computation time. Experiments show us that $\mathcal{L}_3$ can help detect about 20% more failures than $\mathcal{L}_2$. Throughout the next experiments, we are using $\mathcal{L}_3$.

Although in theory, RMax always outperforms R0 and RMin, the practical results are less systematic. This is because $\mathcal{L}_3$ (as well as $\mathcal{L}_2$) is not monotonic, which means that a BP instance requiring a larger number of bins than a second instance can have a lower bound smaller than the second one. In fact, $\mathcal{L}_3$ is more adapted to instances where most item sizes are larger than the third of the capacity. RMax increases the capacity, making unpacked items proportionally smaller. For each of R0, RMin and RMax, there are instances where they contribute to detecting a failure, while the other two do not.

Table 1 presents the performance of the failure detection using each one of the reductions. It shows the ratio of failures found using each reduction over the total number of failures found by at least one filter. Additional reductions have been experimented, with δ being respectively 25%, 50% and 75% on the way between δ_{RMin} and δ_{RMax}. These results were obtained by generating more than 1,000 random instances and computing $\mathcal{L}_3$ on each of their reductions. Here is how the instances were produced:

Inst1. Number of bins, number of items and capacity C each randomly chosen between 30 and 50. Bins already filled up to $1..C$. Random item sizes in $\{1,\ldots,C\}$.
Inst2. 50 bins. Capacity = 100. Number of items is 100 or 200. Size with normal distribution ($\mu = 5000/n$, $\sigma \in \{3n, 2n, n, n/2, n/3\}$ where n is the number of items). Among these, percentage of items already placed $\in \{10\%, 20\%, 30\%, 40\%, 50\%\}$.
Inst3. Idem as 2, but the number of placed items is 90% or 95%.

Table 1. Comparison of the number of failures found with different reductions

Instances	Number of failures detected (%)					
	RMin	R25	R50	R75	RMax	R0
Inst1	74.16	78.87	86.40	89.53	**99.58**	74.79
Inst2	**99.93**	86.75	87.03	87.8	87.15	**99.93**
Inst3	80.64	86.55	93.37	97.75	**99.39**	98.52

This reveals that some types of instances are more adapted to R0 or RMin, while some are more adapted to RMax. The intermediate reductions R25, R50 and R75 were never better in average than RMin and RMax. Thus, they were not considered in the following experiments.

Comparison on benchmark instances. For the analysis to be more relevant, we compared the behavior of the three proposed reductions on real instances. CP algorithms were run over Scholl's SALBP-1 benchmark [5] and on Scholl's bin packing instances [6] (first data set with n=50 and n=100), and at every change in the domains of the variables, the current partial solution was extracted. We randomly selected 30,000 extracted instances from each. In the second case, only instances for which at least one reduction could detect a failure were selected. The three reductions using $\mathcal{L}_3$ were applied on these selected instances. Figure 2 gives a schema of the results.

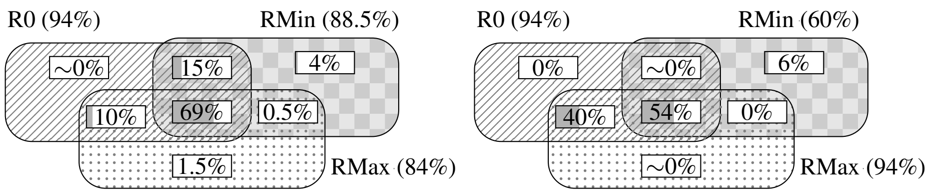

Fig. 2. Proportions of failure detections using each reduction on SALBP-1 instances (left) and BP instances (right)

These results show that R0 detects a larger number of failures. But (almost) all of its failures are also detected by one of the others. Hence, combining RMin and RMax is better than using R0 alone. It is also useless to combine R0 with RMin and RMax.

Impact on a CP search. We compared the effect of applying the failure detection strategy in a CP search on Scholl's bin packing instances N1 and N2 (360 instances), using R0, RMin, RMax and then RMin and RMax combined, with a time limit of five minutes for each instance. For the instances for which all reductions leaded to the same solution, the mean computation time of the searches was computed. All these results are presented in Table 2. One can observe that RMin and Rmax combined find more optimal solutions (though there is no significative difference with R0), and lead faster to the solution than the others (33% speedup compared to R0).

Table 2. Comparison of the reductions on solving the BPLB problem

	No pruning	R0	RMin	RMax	RMin & RMax
Number of optimal solutions	281	317	315	309	**319**
Mean time (s)	5.39	1.88	1.60	3.50	**1.25**

5 Conclusion

We presented two new reductions of a partial solution of the `Pack` constraint to a bin packing problem. Through a CP search, these reductions are submitted to a bin packing lower bound algorithm in order to detect failures of the `Pack` constraint as suggested by Shaw in [1].

We proved that our second reduction (RMax) theoretically provides a better failure detection than the others, assuming a perfect lower-bound algorithm. We conclude that the best strategy is to consider both RMin and RMax filters in a CP search.

Acknowledgments. The first author is supported by the Belgian FNRS (National Fund for Scientific Research). This research is also partially supported by the Interuniversity Attraction Poles Programme (Belgian State, Belgian Science Policy) and the FRFC project 2.4504.10 of the Belgian FNRS.

References

1. Shaw, P.: A constraint for bin packing. In: Wallace, M. (ed.) CP 2004. LNCS, vol. 3258, pp. 648–662. Springer, Heidelberg (2004)
2. Martello, S., Toth, P.: Lower bounds and reduction procedures for the bin packing problem. Discrete Appl. Math. 28(1), 59–70 (1990)
3. Labbé, M., Laporte, G., Mercure, H.: Capacitated vehicle routing on trees. Operations Research 39(4), 616–622 (1991)
4. Bourjolly, J.M., Rebetez, V.: An analysis of lower bound procedures for the bin packing problem. Comput. Oper. Res. 32(3), 395–405 (2005)
5. Scholl, A.: Data of assembly line balancing problems. Technische Universität Darmstadt (93)
6. Scholl, A., Klein, R., Jürgens, C.: Bison: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem. Computers & Operations Research 24(7), 627–645 (1997)