Bi-dimensional Domains for the Non-overlapping Rectangles Constraint

Fabio Parisini

DEIS, University of Bologna
V.le Risorgimento 2, 40136, Bologna, Italy

Given a set of rectangles and a bi-dimensional container, the non-overlapping rectangles constraint aims to obtain consistency such that all the rectangles can be placed without intersection inside the box. So, the $nonOverlapping([R_1, \ldots, R_n], Box)$ holds iff all rectangles are placed inside the Box and no two rectangles R_i and R_j overlap.

The n dimensional version of this constraint, called *diffn* [1], has been used in many applications. In the context of two dimensions, it has an obviously prominent role in various flavors of placement problems (such as 2-dim packing and cutting problems) and scheduling problems to model resource constraints [2].

Many propagation algorithms have been proposed in the literature for the bi-dimensional version of the constraint: to handle its decomposition (pairwise non-overlapping constraints), constructive disjunction or the cardinality operator can be used [3]; the most effective algorithm so far is the specialization of the general sweep pruning technique [4].

Much effort has been recently spent on optimal rectangle packing problem, which is strictly related to the non-overlapping rectangles constraint. Korf [5] exploited relaxations inherited from the bin-packing area and, more recently, a meta-CSP approach has been proposed [6]. Much interest has arisen for techniques using global constraints [7], such as the *cumulative* and the *sweep* constraint, in conjunction with interval-based heuristics to solve the problem.

The traditional way of modeling the non-overlapping rectangles constraint involves the definition of a pair of variables per rectangle, standing for the coordinates of the left-bottom corner of the rectangle itself. However, in this way, the non-overlapping rectangles constraint is expressed through a disjunction of inequality constraints, which cannot perform propagation even when one of the rectangles' position is fixed. Instead, using a bi-dimensional representation of domains, it would be possible to store more precise information about occupied/free surface patches.

Following this idea, I have identified two main issues to face in my research activity:

- The choice of the data structure to be used to represent bi-dimensional domains, which has to be simple to mantain during search.
- The exploitation of the data structure to implement efficient and effective propagation algorithms on its top; the new available information can be used to perform innovative reasoning about rectangles which have already been placed and sum of areas globally available for rectangles still to be placed.

As for the first issue, the *region quad-tree* [8] seems to suite best to the need of representing rectangular domains; in the current constraint implementation a single quad-tree

M. Garcia de la Banda and E. Pontelli (Eds.): ICLP 2008, LNCS 5366, pp. 811–812, 2008.

is used for storing all rectangles domains by labeling its nodes with different "colors", representing free or occupied surfaces patches.

Whenever any rectangle placement decision is taken, the quad-tree data structure is updated, i.e. the relevant surface patches are set as occupied. As a consequence, addressing the second issue, ad hoc routines are triggered to check if the remaining area is wide and tall enough to host uninstantiated rectangles.

In addition to that, the whole area remainder is exploited to perform global propagation. Using a graph structure which is similar to the one defined for the global cardinality constraint [9], together with its filtering algorithm, a connection is created between rectangles still to be placed, corresponding to GCC variables, and the available surface patches, corresponding to GCC domain values.

The constraint implementation is being tested on a set of rectangle packing instances, in *SICStus Prolog*, and its perfomances are compared to the *sweep* algorithm [4]. The first results show that the pruning algorithms included into our implementation of the non-overlapping rectangles constraint are effective; a higher number of backtracks is requested to the competitor [4]. In spite of that the algorithm which employs the non-overlapping rectangles constraint needs more time per backtrack, and more total time to solve the problem instances. Preliminary results show that search strategies tailored on the data structure could lead to important performance improvements, so the development of such strategies is one of the main directions in which the research work is heading to.

In general, we can say that the current implementation is still prototipal; we aim to improve its efficiency by defining smart search strategies and combining the copious sub-routines which compose the filtering algorithm in a more clever way. The fact that the number of backtracks is less than the one of the sweep algorithm is a good starting point; the integration of external constraints and related propagation algorithms, such as the *sweep* and $cumulative$ constraint, could lead to even better resuts.

References

1. Beldiceanu, N., Contejean, E.: Introducing global constraints in CHIP. Mathl. Comput. Modelling 20(12), 97–123 (1994)
2. Aggoun, A., Beldiceanu, N.: Extending CHIP in order to solve complex scheduling and placement problems. Mathl. Comput. Modelling 17(7), 57–73 (1993)
3. Hentenryck, P.V., Saraswat, V., Deville, Y.: Design, implementation and evaluation of the constraint language cc(fd). Constraints: Basic and Trends 910 (1995)
4. Beldiceanu, N., Carlsson, M.: Sweep as a generic pruning technique applied to the non-overlapping rectangles constraint. In: Walsh, T. (ed.) CP 2001. LNCS, vol. 2239, pp. 377–391. Springer, Heidelberg (2001)
5. Korf, R.E.: Optimal rectangle packing: New results. In: ICAPS 2004, pp. 142–149 (2004)
6. Moffitt, M.D., Pollack, M.E.: Optimal rectangle packing: A meta-csp approach. In: ICAPS 2006, pp. 93–102 (2006)
7. Simonis, H., Sullivan, B.O.: Using global constraints for rectangle packing. In: CP-AIOR (2008)
8. Samet, H.: The design and analisys of Spatial Data Structures. Addison-Wesley, Reading (1989)
9. Regin, J.: Global constraints and filtering algorithms. In: Milano, M. (ed.) Constraint and Integer Programming. Kluwer Academic Publisher, Dordrecht (2004)