Bi-dimensional packing: a Constraint Programming approach

Marco Gavanelli[1], Andrea Lodi[2], Michela Milano[2], and Fabio Parisini[2]

[1] Dept of Engineering, University of Ferrara
Via Saragat 1, 44100, Ferrara, Italy
[2] DEIS, University of Bologna
V.le Risorgimento 2, 40136, Bologna, Italy

The problem of packing a set of rectangles into a given bi-dimensional container is NP-hard in the strong sense; it is common in many practical applications, such as scheduling, placement and cutting problems. We consider a Constraint Programming based approach to the rectangle packing problem; we define a novel implementation of the non-overlapping rectangles constraint, in conjunction with the definition of ad-hoc search strategies.

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; using that approach it is very difficult to perform any pruning until at least one of the coordinates of the rectangles of interest is fixed. The key of our approach, instead, lies in the fact that we define native bi-dimensional domains for the rectangular variables to exploit the innate peculiarities of the problem; in fact we make use of a region quad-tree representation of the area which is available for rectangle placement.

A region quad-tree (in the following, simply *quad-tree*) partitions a bi-dimensional space in surface patches, equivalent to tree nodes, which are either black, gray or white. A node is gray when its area cannot be considered completely black or white; the node is a non-leaf node and it has got 4 children nodes. Leaf nodes are either black or white, and represent areas which are homogeneous in color. When we use the quad-tree to represent the area which is still available to place a rectangle within the given container, we use white nodes for surface patches which can be used from the rectangle, black nodes for the others.

This quad-tree data structure allows us to perform a propagation which is much tighter than the one performed in classical approach; we are going to illustrate 3 filtering algorithms we have included in our non-overlapping rectangles constraint implementation.

The first filtering algorithm we have implemented is a very simple one; it takes into account a single rectangular variable at a time and explores the quad-tree looking for surface patches which are not suitable for hosting the rectangle of interest because of their dimensions. This algorithm is usually triggered when an area removal operation takes place because of the other filtering algorithms or because of the labeling.

Also the second filtering algorithm we have included takes into account a single rectangular variable at a time, looking for its compulsory parts. The concept of compulsory parts is well-known and widely used in the context of scheduling problems; in our environment a compulsory part for a rectangle R is a surface patch which R is forced

to occupy, so we reserve that area for R by removing it from the domain of all the other rectangular variables.

Our approach to compulsory parts has got a major advantage over the classical ones; due to the bi-dimensional nature of our quad-tree representation we are able to effectively take in account any compulsory part and successfully update the domains accordingly.

The third filtering algorithm takes into account occupancy information from all the rectangular variables involved in the constraint. The algorithm builds, similarly to the Global Cardinality Constraint (GCC), a bipartite graph where the rectangles which still have to be placed are linked to the surface patches which are available for them on the quad-tree.

All of these filtering algorithms are included within our implementation of the non-overlapping rectangles constraint; we have to check their efficiency and effectiveness in a testing environment. Recent works by Simonis and O'Sullivan show that combining existing constraints, such as cumulative and sweep constraint, in a Constraint Programming environment to solve bi-dimensional packing problems turns out to be a very powerful strategy.

We have defined a number of different configurations, taking into account our non-overlapping rectangles constraint, cumulative and sweep constraint. We are collecting results for different search strategies, trying to determine in which situations our constraint is able to perform a tighter propagation in shorter execution time; we aim to define search strategies which take advantage of the information which is available from the non-overlapping rectangles constraint during the search. An interesting path we are going to follow in this direction regards the possibility of studying statistical connections between solution density information and the state of the quad-tree data structure during the search for the purpose of defining ad-hoc search strategies.

We are also studying the possibility to use our constraint as part of a hybrid framework to solve allocation and placement problems of application components on a Field-Programmable Gate Array, FPGA, minimizing a communication cost related objective.