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

RiCeRcA 2008

The Rectangle Packing Problem

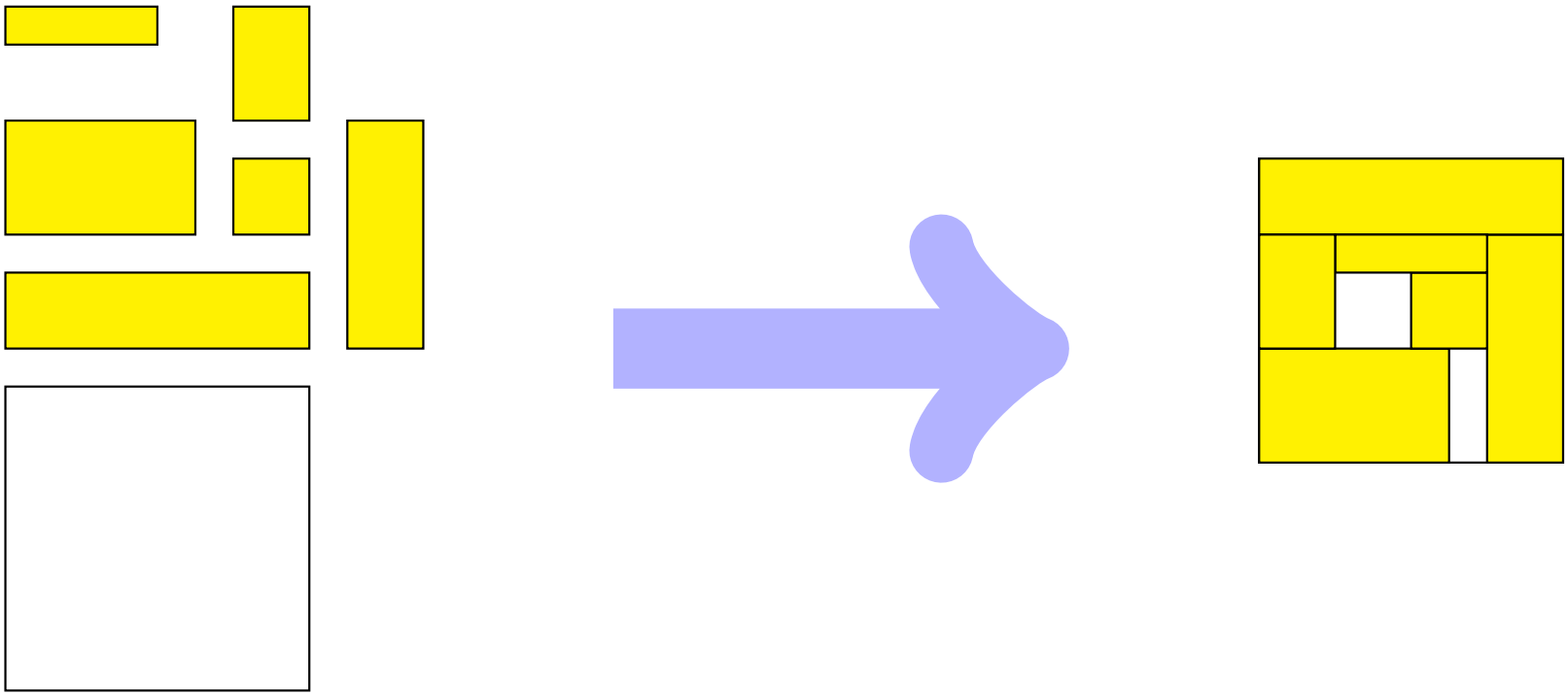
The Rectangle Packing Problem is the problem of positioning a set of rectangular items wihout overlap within a given container.

Non overlapping rectangles constraint

$$nonOverlapping([R_1, \ldots, R_n], Box) \leftrightarrow \forall i, j \ (i \neq j \rightarrow R_i \text{ non overlap } R_j)$$

- Two-Dimensional Packing problems are common in many real life situations such as VLSI Design, Scheduling, Placement and Cutting problems.

The Rectangle Packing Problem



Modeling the problem

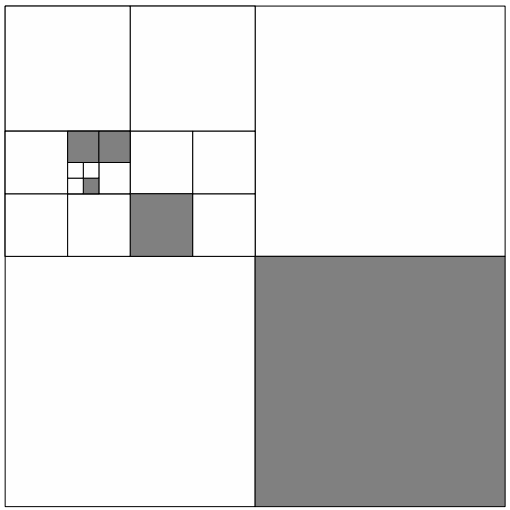
Traditional problem modeling:

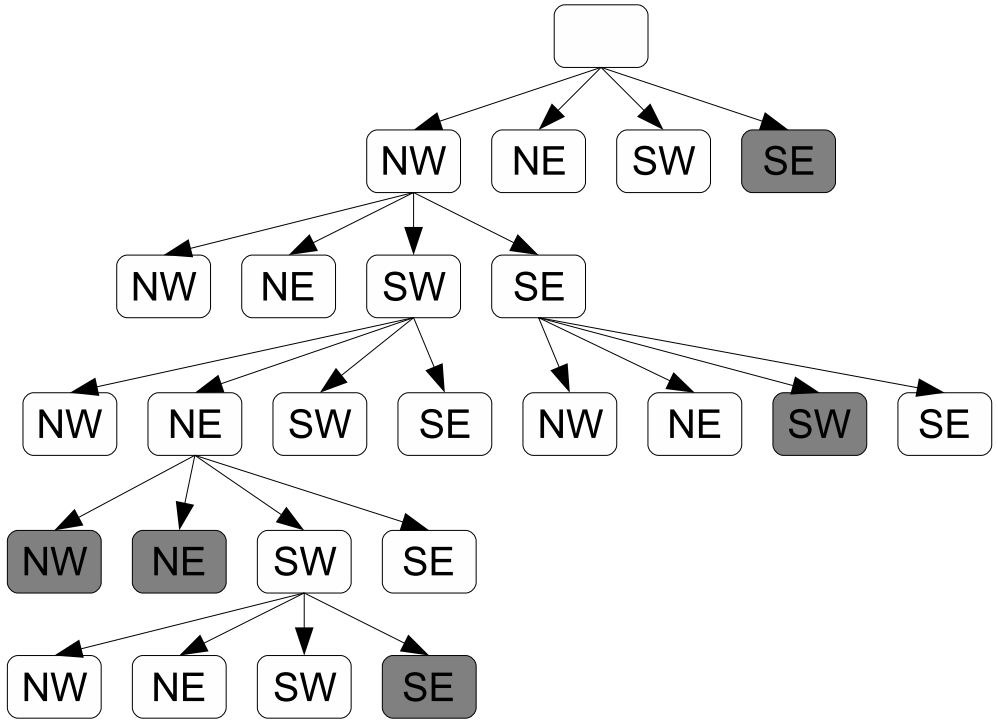
- Definition of a *pair of origin variables per rectangle*, standing for the coordinates of the left-bottom corner of the rectangle.
- Difficult to perform any pruning until at least one of the coordinates is fixed.

Our modeling approach:

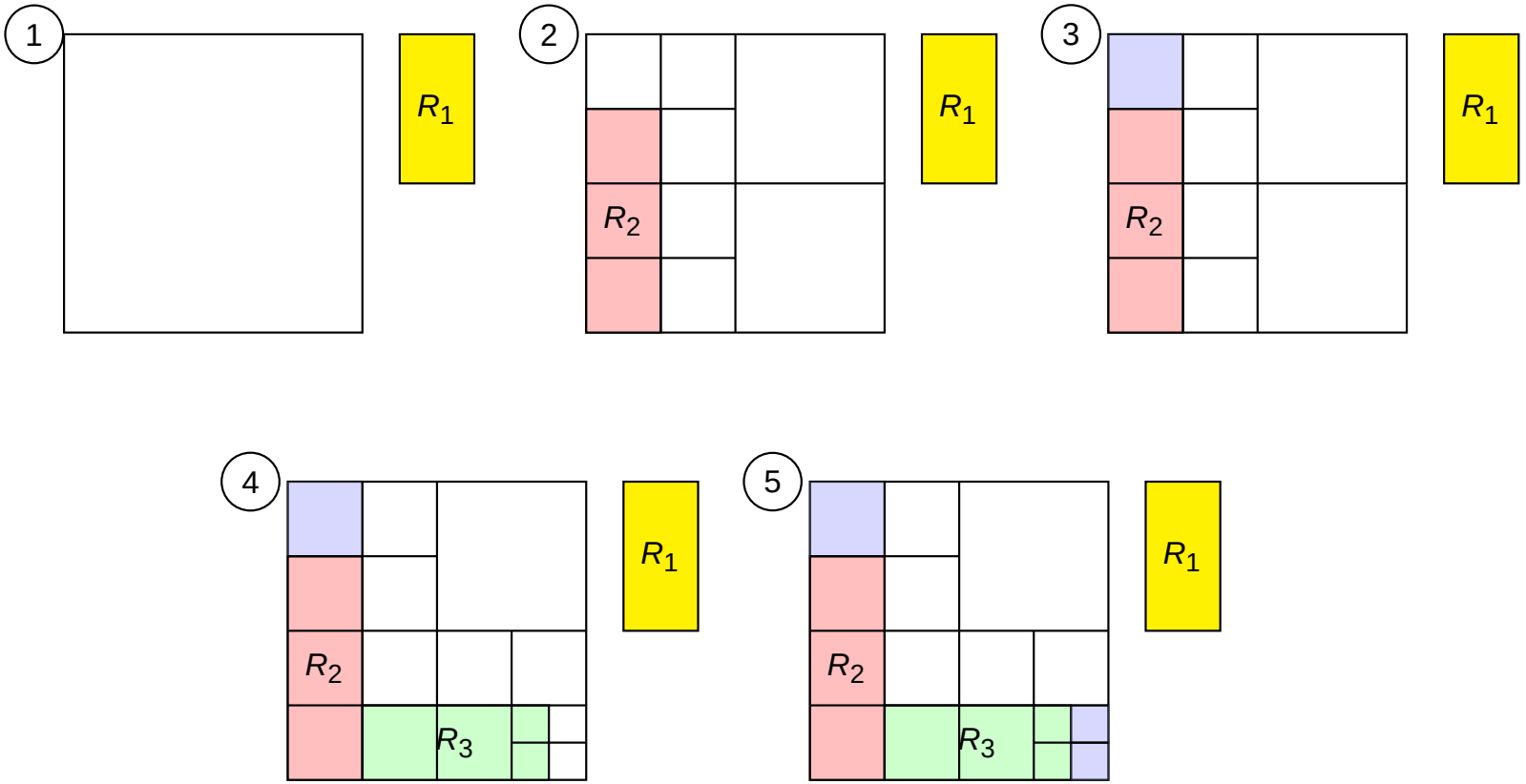
- *Bi-dimensional domains* for the rectangular variables making use of a region quad-tree representation.
- A leaf node is either `white` or `black`; a white node stands for an area which is available for hosting the rectangle, a black node stands for an occupied area.

Region Quad-Tree Representation

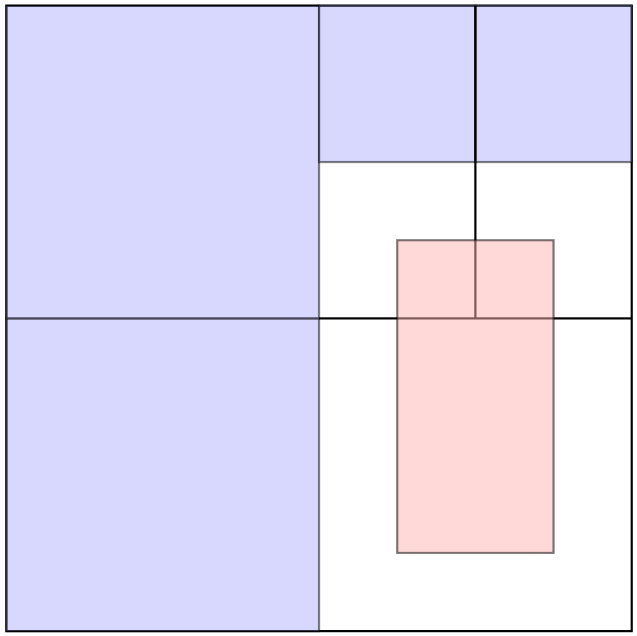


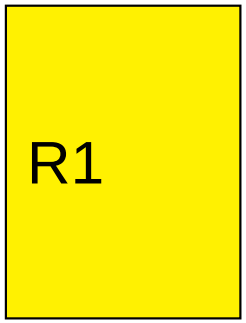


Quad-Tree Evolution During Labeling



Compulsory Parts





Quad-Tree seen as R1 domain:

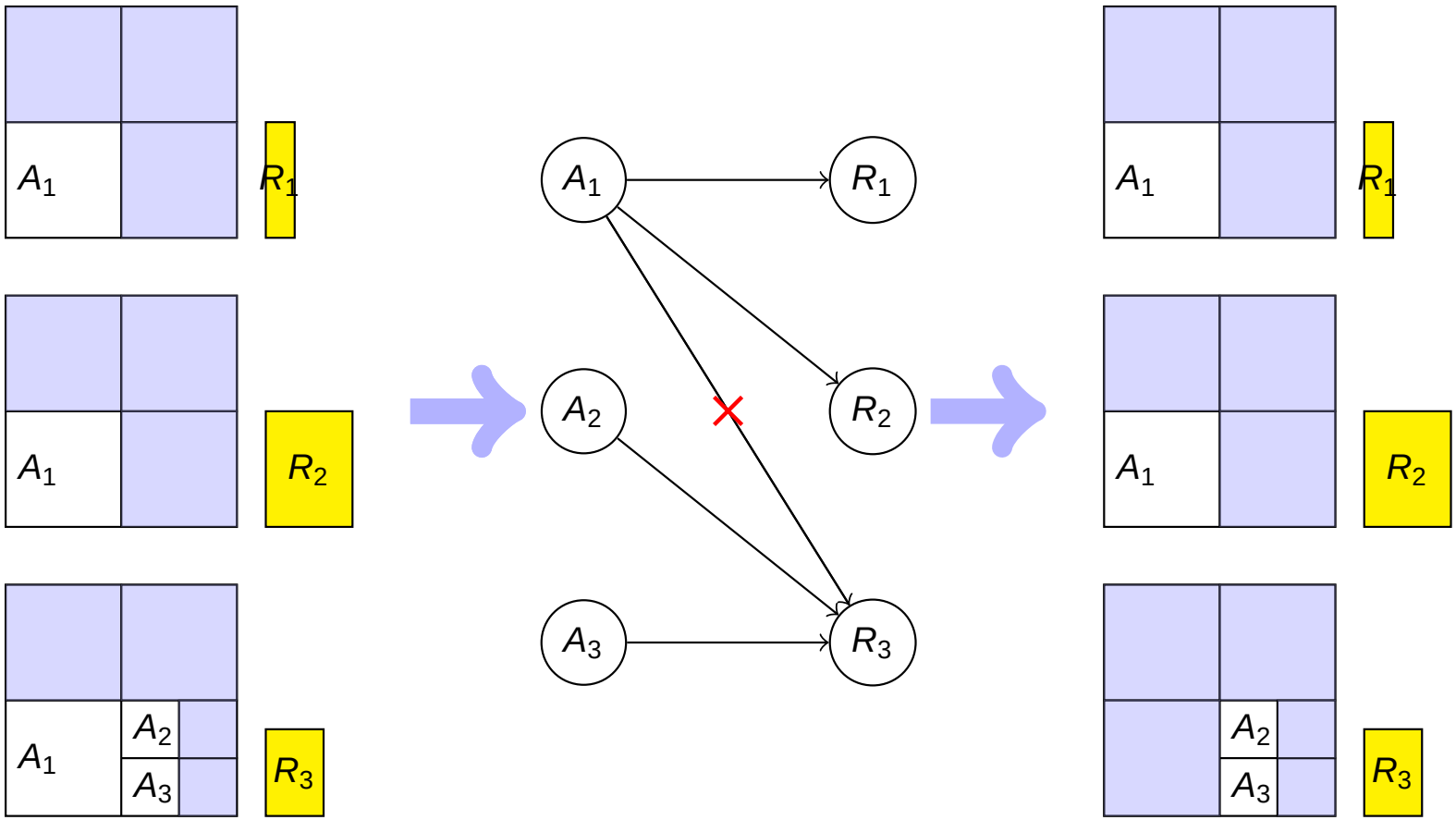
- A compulsory part is detected: that area is going to be removed from all the domains of the other rectangles of the problem.

Compulsory Parts

The concept of compulsory parts is widely used in the context of scheduling problems.

- A compulsory part for a rectangle R is a surface patch which R is *forced to occupy*, so we can remove it from the domain of all the other rectangular variables.
- We are able to *exploit the bi-dimensional nature* of the quad-tree representation to effectively take into account all the compulsory parts and operate on the domain accordingly.

GCC Like Filtering Algorithm



The Global Cardinality Constraint

- In the Global Cardinality Constraint, a set of variables $X = \{x_1, \ldots, x_p\}$ take their values in a subset of $V = \{v_1, \ldots, v_d\}$; GCC constrains the number of times a value $v_i \in V$ is assigned to a variable in X to be in an interval $[l_i, c_i]$.
- The GCC builds a bipartite graph of values and variables, computes the *maximum flow* and detects strongly connected components. Such information is used to reduce variables' domains and detect infeasible solutions.

Implementation

- Implemented on top of `SICStus Prolog`'s CLP(FD) library.
- A pair of origin variables is added for each rectangular variable, as the propagation engine needs.
- All the enumerated filtering algorithms are embedded within a `single global constraint`.
- The current implementation is still prototipal.

Search Configurations and Strategies

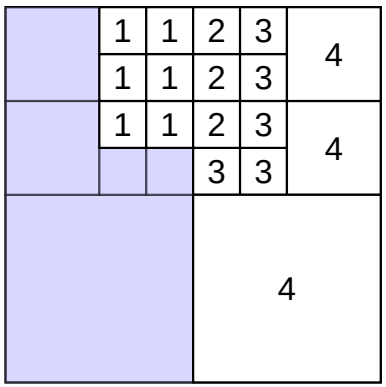
We have defined a number of different configurations, taking into account the following global constraints:

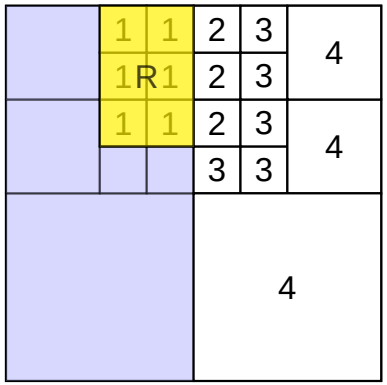
- Our non-overlapping rectangles constraint.
- Sweep based SICStus disjoint2 constraint.
- Cumulative constraint.

Instances and search strategies

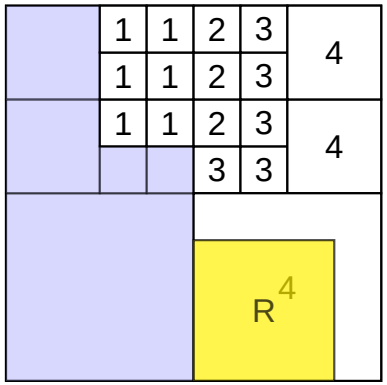
- Tests run on similar instances with little slack.
- Tests run on different search strategies: naive, x then y, disjunctive, semantic4, semantic, dual, interval based search strategies.
- We aim to define search strategies which take advantage of the information which is available from the non-overlapping rectangles constraint during the search.

Search Strategies





most constrained



least constrained

Results

- Our main competitor is SICStus's disjoint2 constraint, which handles the same kind of problems as we do, but with a completely different data structure based on *sweep* technique.
- The pruning algorithms included in the non-overlapping rectangles constraint are effective; using the same search strategies, the non-overlapping rectangles constraint requires a lower number of backtracks than disjoint2.
- The main weakness of our current approach is that the non-overlapping rectangles constraint needs more time per backtrack, and more total time to solve the problem instances.
- Extra filtering algorithms involving pairs of rectangular variables are under development; we work on both increasing pruning power of the constraint and optimizing existing implementation.
- Recent works show how search strategies have a greatest impact on two-dimensional packing problems; we aim to define a search strategy which is tailored to the non overlapping rectangles constraint.

Summary

- Region Quad-Trees are used to exploit the two-dimensionality of the Rectangle Packing Problem.
- Most interesting search configurations make use of more than one global constraint.

Outlook

- Defining a search strategy which exploits the Quad-Tree based domain representation.
- Running comprehensive tests to find out which are the kind of instances the non-overlapping rectangles constraint gets the best results in terms of execution time.