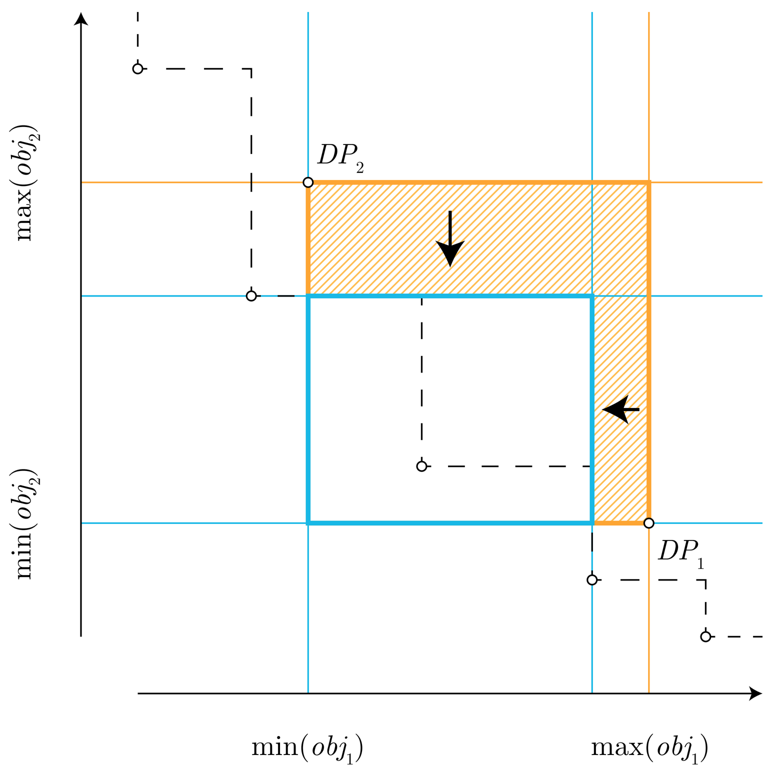
The Bi-Objective Pareto Constraint

Renaud Hartert and Pierre Schaus

ICTEAM, UCLouvain
Louvain-la-Neuve, Belgium

renaud.hartert@uclouvain.be
pierre.schaus@uclouvain.be

The Bi-Objective Pareto Constraint prunes branches that cannot lead to an efficient solution



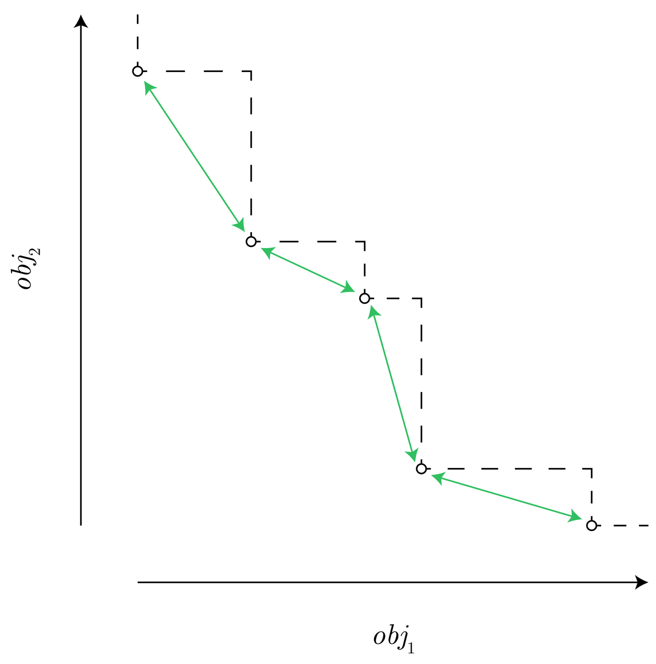
The Bi-Objective Pareto constraint is defined over two objective variables to minimize obj_1, obj_2 and an archive $\mathcal{A}$ i.e. a set of previously discovered solutions such that no solution in the archive dominates an other solution in the archive.

The Bi-Objective Pareto constraint forces the following filtering rule where the vector DP_i corresponds to an extremity of the cartesian product of the objective variables:

$$\max(obj_i) \leftarrow \min(\{\max(obj_i)\} \cup \{sol_i - 1 \mid sol \in \mathcal{A}, sol \preceq DP_i\})$$

Each time a new solution is discovered, it is added in the archive to tighten the bounds of the objective variables. Solutions in the archive that are dominated by this new solution have to be removed from the archive to maintain its *domination-free* property.

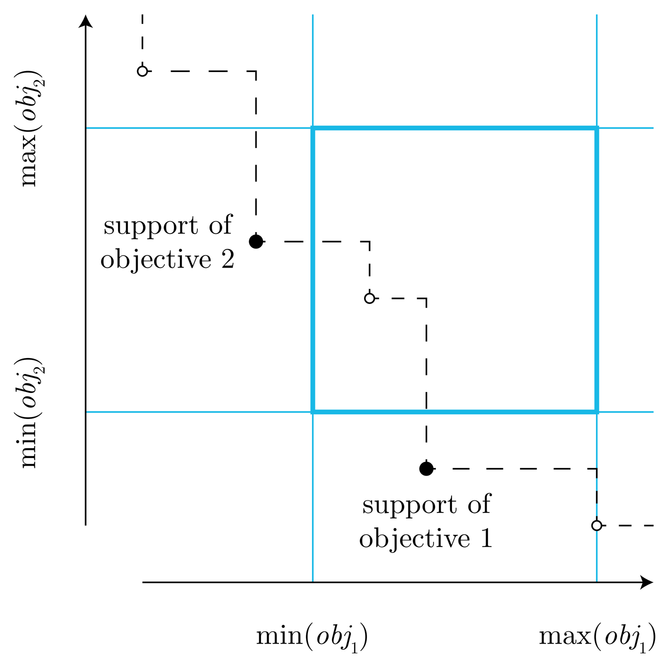
The bi-objective ordering property



Sorting the solutions of a bi-objective archive $\mathcal{A}$ in increasing order with regard to one objective amounts to sort these solutions in decreasing order with regard to the other objective.

In the following, we denote $\mathcal{A}^{>_i}$, the archive ordered by decreasing value of objective i.

Support of objective i

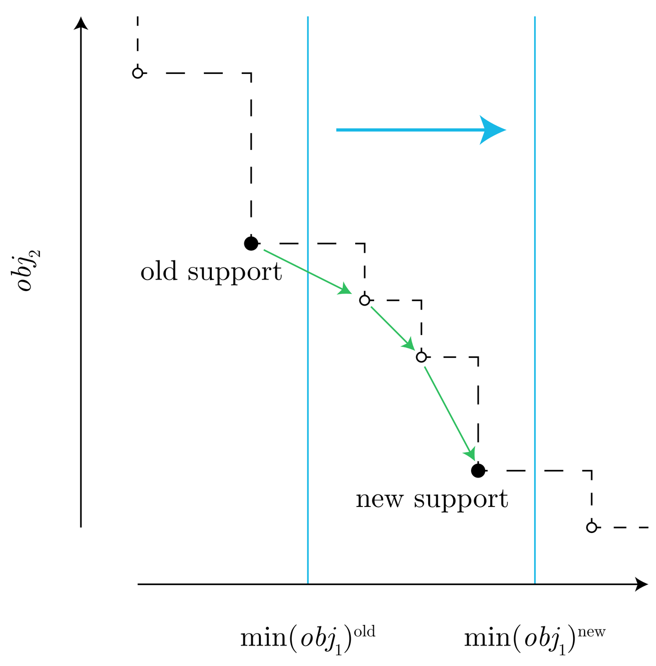


Support are used to adjust the upper bounds of the objective variables. The support of objective 1 is as follows:

$$\arg\min_{sol' \in \mathcal{A}}(\{sol'_2 \mid sol'_2 < \min(obj_2)\})$$

Supports are never included in the cartesian product of the domain of the objective variables i.e. supports cannot be dominated by a new solution.

Supports are easy to maintain with a trail-based CP Solver

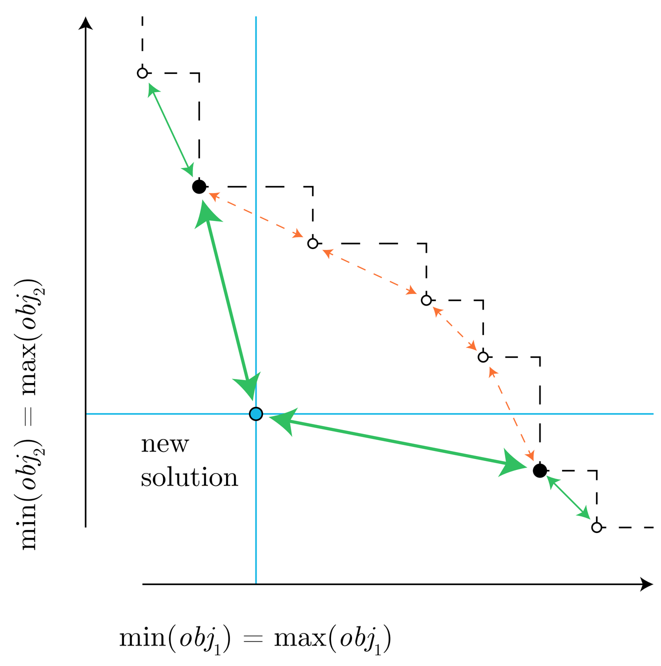


When the lower bound of objective i is adjusted, it is easy to access its next support by iterating on the successors, in $\mathcal{A}^{>_i}$, of its previous support.

The sum of these iterations, along a branch of the search tree, is bound by the number of solutions in the archive $\mathcal{A}$.

Reversible pointers are used to restore the supports when a bracktrack occurs.

New solutions are inserted in the archive in constant time



If the archive is ordered with regard to one objective (e.g. $\mathcal{A}^{>_1}$ or $\mathcal{A}^{>_2}$), then all the solutions contained between the supports of a newly discovered solution are dominated by this new solution.

The new solution is inserted in constant time by updating the pointers of its supports. The archive remains ordered and is still *domination-free.*

19th International Conference on Principles and Practice of Constraint Programming, 2013.