

Constraint programming overview

Constraint programming (CP) is a software technology becoming more widespread thanks to many successes with effective solving of large, real-life, particularly combinatorial, problems. It provides a powerful technique for different problems such as scheduling, timetabling, resource allocation, or network configuration.

Research results from different fields (operations research, artificial intelligence, discrete mathematics, graph theory) are all involved in the core of Constraint Programming packages. Constraint Programming allows for representing many problems in a way which is very close to a natural language description, thanks to semantic constraints. Benefits are important: fast prototyping, compact code, easy modification, and good performance. It allows for specifying powerful decision support systems.

Basic concepts of Constraint Programming

A *Constraint Programming (CP) problem* is defined by its *decision variables* with their *domains* and *constraints* over these variables. The problem definition is usually completed by a *branching strategy* (also referred to as *enumeration* or *search strategy*).

CP makes active use of the concept of *variable domains*, that is, the set out of which a decision variable takes its values. In *finite domain Constraint Programming* these are sets or intervals of integer numbers.

Each *constraint* in CP comes with its own (set of) solution algorithm(s), typically based on results from other areas, such as graph theory. Once a constraint has been established it maintains its set of variables in a solved state, *i.e.*, its solution algorithm removes any values that it finds infeasible from the domains of the variables.

The constraints in a CP problem are linked by a mechanism called *constraint propagation*: whenever the domain of a variable is modified this triggers a re-evaluation of all constraints on this variable which in turn may cause modifications to other variables or further reduction of the domain of the first variable as shown in the example in Figure *Example of constraint propagation* (the original domains of the variables are reduced by the addition of two constraints; in the last step the effect of the second constraint is propagated to the first constraint, triggering its re-evaluation).

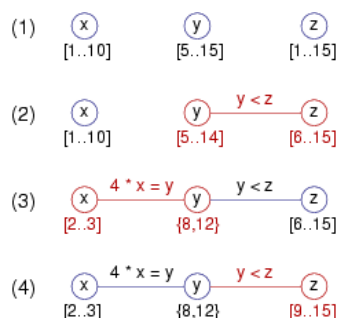


Figure 1.1: Example of constraint propagation

Solutions of a CP problem can be found by a systematic search on the set of possible assignments of values to variables.

One may wish to find:

- just *one* feasible solution, without any choice preference;
- *all* the solutions;
- an *optimal* solution (or at least a nearly optimal solution) that optimizes a certain objective function defined on a subset of the variables of the problem.