One-sheet: Choco-3.1.0

C.Prud'homme, J.G.Fages – August 30, 2013

Solver

The `Solver` is a central object and must be created first: `Solver solver = new Solver();`

Variables

The `VariableFactory` (`VF`) eases the creation of variables. Available variables are: `BoolVar`, `IntVar`, `SetVar`, `GraphVar` and `RealVar`.

Note, that an `IntVar` domain can be bounded (only bounds are stored) or enumerated (all values are stored); a boolean variable is a 0-1 `IntVar`.

Views

A View is a variable whose domain is defined by a function over another variable domain. `VF` available views are: `not`, `offset`, `eq`, `minus`, `scale` and `real`.

Constants

fixed-value integer variables should be created with the specific `VF.fixed(int, Solver)` function.

Constraints

Several constraint factories ease the creation of constraints: `LogicalConstraintFactory` (`LCF`), `IntConstraintFactory` (`ICF`), `SetConstraintsFactory` (`SCF`) and `GraphConstraintFactory` (`GCF`). `RealConstraint` is created with a call to new and to `addFunction` method. It requires the `Ibex` solver.

Constraints hold once *posted*: `solver.post(c);` Reified constraints should not be posted.

Search

Defining a specific way to traverse the search space is made thanks to: `solver.set(AbstractStrategy)`. Predefined strategies are available in `IntStrategyFactory` (`ISF`), `SetStrategyFactory` and `GraphStrategyFactory`.

Large Neighborhood Search (LNS)

Various LNS (random, propagation-guided, etc.) can be created from the `LNSFactory` to improve performance on optimization problems.

Monitors

An `ISearchMonitor` is a callback which enables to react on some resolution events (failure, branching, restarts, solutions, etc.). `SearchMonitorFactory` (`SMF`) lists most useful monitors. User-defined monitors can be added with `solver.getSearchLoop().plugSearchMonitor(...)`.

Limits

A limit may be imposed on the search. The search stops once a limit is reached. Available limits are `SMF.limitTime`, `SMF.limitFail`, etc.

Restarts

Restart policies may also be applied `SMF.geometrical` and `SMF.luby` are available.

Logging

Logging the search is possible. There are variants but the main way to do it is made through the `SMF.log(Solver, boolean, boolean)`. The first boolean indicates whether or not logging solutions, the second indicates wether or not logging search decisions. It also print, by default, main statistics of the search (time, nodes, fails, etc.)

Solving

Finding if a problem has a solution is made through a call to: `solver.findSolution()`. Looking for the next solution is made thanks to `nextSolution()`. `findAllSolutions()` enables to enumerate all solutions of a problem. To optimize an objective function, call `findOptimalSolution(...)`. Resolutions perform a Depth First Search.

Solutions

By default, the last solution is restored at the end of the search. Solutions can be accessed as they are discovered by using an `IMonitorSolution`.

Explanations

Choco natively supports explained constraints to reduce the search space and to give feedback to the user. Explanations are disabled by default.