JaCoP
Java Constraint Programming libraray and its applications

Krzysztof Kuchcinski
Krzysztof.Kuchcinski@cs.lth.se

Department of Computer Science
Lund University

Outline

Why Java?

- Mature language that is portable between many platforms.
- Strongly typed and safe language.
- Efficient memory management and garbage collection.
- Easy to develop new applications since it has rich standard libraries.
- Efficient implementations with on-time compilations.
- Support for multi-threading.
- ...

JaCoP library

- Finite domain constraint programming paradigm implemented in Java 1.5 (ca. 40,000 lines of code).
- Provides different type of constraints
  - most commonly used primitive constraints, such as arithmetical constraints, equalities and inequalities,
  - logical, reified and conditional constraints,
  - combinatorial (global) constraints.
- Provides a number of standard search methods.
- It is used as usual Java API, either by providing it as a JAR file or specifying access to a class directory containing all JaCoP classes.
- Used in several research projects at several places.

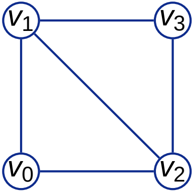
JaCoP Example

```
import JaCoP.*;
import java.util.*;
public class Main {

    static Main m = new Main ();

    public static void main (String[] args) {
        FDstore store = new FDstore();  // define FD store
        int size = 4;
        // define FDVs
        FDV[] v = new FDV[size];
        for (int i=0; i<size; i++)
            v[i] = new FDV(store, "v"+i, 1, size);
        // define constraints
        store.impose( new XneqY(v[0], v[1]) );
        store.impose( new XneqY(v[0], v[2]) );
        store.impose( new XneqY(v[1], v[2]) );
        store.impose( new XneqY(v[1], v[3]) );
        store.impose( new XneqY(v[2], v[3]) );

        // search for a solution and print results
        ArrayList<FDV> list = new ArrayList<FDV>();
        for(FDV var : v) list.add(var);
        boolean result = Solver.searchOne(store, list, new SearchOne(), new IndomainMin(), new Delete());
        if ( result )
            System.out.println("Solution: " + list + "\n*** Yes");
        else
            System.out.println("*** No");
    }
}
```



JaCoP Example (cont'd)

The program produces the following output

```
Solution: [v0=1, v1=2, v2=3, v3=1]
*** Yes
```

JaCoP Constraints

- primitive
  - $X + Y = Z$,
  - $X >= Y$,
  - $X \times Y = Z$,
  - etc.
- conditional
  - IF $(X = Y)$ THEN $B > 10$ ELSE $C <= 7$
  - $X = Y \Leftrightarrow A > C$
- reified
  - $X = Y \Leftrightarrow B$
- logical
  - $(X = Y) \vee (X = Z)$
  - $\neg(A = B)$

JaCoP Combinatorial Constraints

- `Alldifferent`, `Alldiff`, `Alldistinct`– all varibales have different values.
- `Diff2`– non-overlapping rectangles in 2-D space.
- `Cumulative`– cumulative use of resources.
- `Circuit`– Hamiltonian circuit in the graph.
- `Element`– finite relation between I and V, $V = List[I]$.
- `Assignment`– $X_i = j \Leftrightarrow Y_j = i$
- `ExtensionalSupport`, `ExtensionalConflict`– relations between variables defined by tuples.
- `Sum`, `SumWeighted`– summation of finite domain variables.
- `Min`, `Max`– min and max value form the list of variables.

Constraints Implementation

- All necessary data structures for constraint consistency are built in FDstore.
- All constraints extend abstract class `Constraint` that defines, among other methods, two most important methods
  - `consistency`
  - `satisfied`
- Primitive constraints extend this class to `PrimitiveConstraint`
  - `notConsistency`
  - `notSatisfied`
- Propagation loop calls consistency methods for the constraints in the evaluation queues.

Constraints Implementation (cont'd)

Bounds consistency method for X + Y = Z

```
public void consistency (FDstore S) {
  while ( S.newPropagation ) {
    S.newPropagation = false;
    in(S, X, Z.min() - Y.max(), Z.max() - Y.min());
    in(S, Y, Z.min() - X.max(), Z.max() - X.min());
    in(S, Z, X.min() + Y.min(), X.max() + Y.max());
  }
}
```

Basic features of the solver

- Evaluation of constraints triggeret by events.
- Satisfied constraints are removed from evaluation.
- New constraint can be posed during search to build new search methods.
- Constraints can have a state that changes during search and backtracking in a similar way as variables.
- Easy to add new constraints with different consistency methods; extend abstract class.
- Can run large examples, e.g. ca. 180 000 constraints.

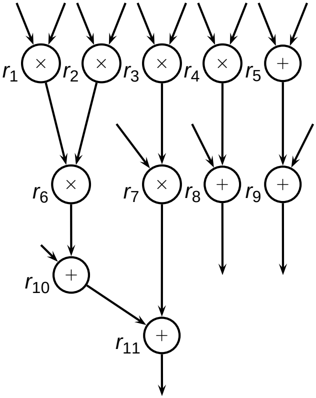
JaCoP Search

- JaCoP offers a number of search methods
  - search for a single solution,
  - find all solutions, and
  - find a solution that minimizes/maximizes a given cost function.
- Search is achieved using depth-first-search together with consistency checking.
- Search is parametrized (different classes for labeling, delete, and indomain).
- There are complete search methods and heuristics
  - depth-first-search and branch-and-bound,
  - credit search,
  - “limited discrepancy search”,
  - hierarchical search.

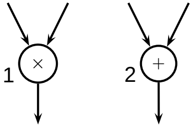
Design Automation Area

- Scheduling and resource assignment in high-level synthesis.
- Partial task assignment of task graphs under heterogeneous resource constraints.
- Time-energy design space exploration for multi-layer memory architectures.
- Synthesis of SoPC (System on Programmable Chip)– cell synthesis and communication synthesis.
- New synthesis method based on graph matching constraint.
- Parallel search.

Assignment and Scheduling Based on Graph Matching

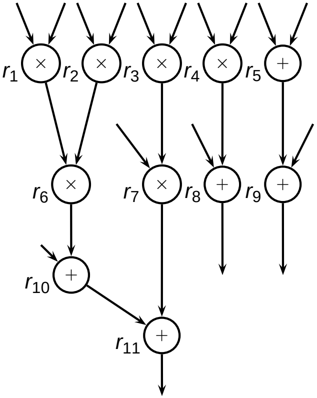


data-flow graph

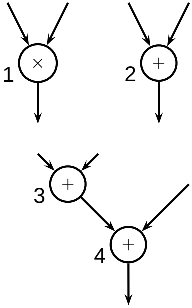


components

Assignment and Scheduling Based on Graph Matching



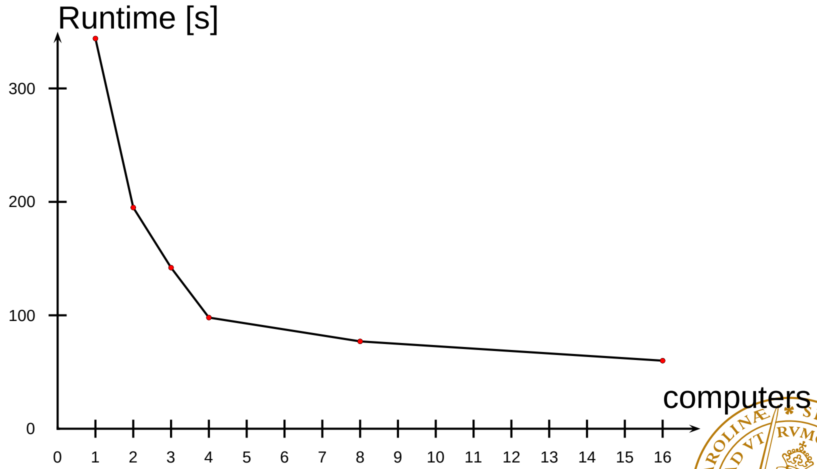
data-flow graph



components

Distributed Ssearch– Golomb 11

Search tree is distributed into a number of computers or cores.



Conclusions

- Ease and intuitive to use while efficient,
- Implementation based on the state-of-the-art constraint programming algorithms backed by few years of experimentation with different designs.
- Not tailored to any specific domains, acceptable efficiency for a wide range of different applications.
- Successfully used in a number of research projects at different places.

Reference

- K. Kuchcinski and R. Szymanek.
  JaCoP Library. User's Guide.
  Technical Report, Lund University, 2006.

- K. Kuchcinski.
  Constraints-driven scheduling and resource assignment.
  ACM Transactions on Design Automation of Electronic Systems (TODAES), 8(3):355–383, July 2003.

- K. Kuchcinski.
  Constraint programming for embedded system design: Principles and practice.
  Lecture notes, 2005.