Towards a Heterogeneous Constraint Solving System on the Grid

Mutsunori BANBARA	Kobe University
Shuji OHNISHI	Kobe University
Katsumi INOUE	NII
Naoyuki TAMURA	Kobe University

CP Workshop in NII, 25th October 2004

About this talk

- HECS project, especially two resulting systems:
  - *Cream*: A Java Class Library for CP
  - *Calc/Cream*: A Constraint Spreadsheet
- *g-HECS*: on-going project for developing a constraint solving system on the Grid
- Shuji Ohnishi talks about automatic generation of neighbors by exploiting disjunctions in constraints.

HECS Project (2002–2003)

We have developed a constraint solving system that supports constraint solvers for finite domains of integers and boolean, and our system supports a Mathematica interface.

- *Cream*: A Java class Library for CP
- *Calc/Cream*: A Constraint Spreadsheet
- Multisat: A Parallel Execution System for Multiple SAT Solvers.
- Prolog Cafe: A Prolog to Java Translator
- Maglog: An Extension of Prolog Cafe for Mobile Multi-agent Systems

Cream (2002–)

Cream is a *Java class library* of constraint programming for finite domains of integers.

- *100% Pure Java*
- *Complete Search*: Constraint propagation and Backtracking (branch-and-bound)
- *Local Search*:
  Simulated Annealing, Taboo Search, etc.
- *Parallel Search*:
  Local search solvers can run in parallel
- Related works:
  ILOG JSolver (commercial), Koalog Constraint Solver (commercial), ICS in Java, JACK, JCL

Example program of Cream

Let us consider a simple problem of production plan.

	number	materialA	materialB	profit
product1	x	$2x$	x	$4x$
product2	y	y	y	$3y$
total		$2x+y$	$x+y$	$4x+3y$
stock		100	60	

- Constraints:

$$x \in \mathbf{Z} \qquad y \in \mathbf{Z}$$
$$x \geq 0 \qquad y \geq 0$$
$$2x + y \leq 100 \qquad x + y \leq 60$$

- We want to know the values of x and y so that they maximize the profit, $4x+3y$.

Example program of Cream (Cont.)

1. Create a constraint network

```
Network net = new Network();
```

2. Create variables

```
IntVariable x = new IntVariable(net);  % x
IntVariable y = new IntVariable(net);  % y
```

3. Describe constraints

```
x.ge(0);    % x ≥ 0
y.ge(0);    % y ≥ 0
x.multiply(2).add(y).le(100); % 2x + y ≤ 100
x.add(y).le(60);              %  x + y ≤ 60
```

Example program of Cream (Cont.)

4. Create an objective variable and set it to network.

```
IntVariable p = x.multiply(4).add(y.multiply(3));
net.setObjective(p);  % 4x + 3y
```

5. Create a solver

```
Solver solver =
     new DefaultSolver(net, Solver.MAXIMIZE);
```

6. Find a solution

```
Solution solution = solver.findBest();
```

7. Get values

```
int pv = solution.getIntValue(p);
int xv = solution.getIntValue(x);
int yv = solution.getIntValue(y);
```

Example program of Cream (Cont.)

The whole program is as follows:

```
import jp.ac.kobe_u.cs.cream.*;

public class Plan {
  public static void main(String args[]) {
    Network net = new Network();
    IntVariable x = new IntVariable(net);
    IntVariable y = new IntVariable(net);
    x.ge(0);
    y.ge(0);
    x.multiply(2).add(y).le(100);
    x.add(y).le(60);
    IntVariable p = x.multiply(4).add(y.multiply(3));
    net.setObjective(p);
    Solver solver = new DefaultSolver(net, Solver.MAXIMIZE);
    Solution solution = solver.findBest();
    System.out.println("profit = " + solution.getIntValue(p));
    System.out.println("x = " + solution.getIntValue(x));
    System.out.println("y = " + solution.getIntValue(y));
  }
}
```

Main Features of Cream

- Local search solvers can run without explicitly defining the neighborhood function.
- SA, Taboo, and IBB (Iterative Branch-and-Bound) solvers are executed in parallel.
- These solvers exchange the information of the current optimal solutions.
- Demonstration of Cream
  - FT10 JSSP (Job Shop Scheduling Problem)
  - 10 machines and 10 jobs
  - 930 is the minimum makespan

Calc/Cream (2002–)

Calc/Cream is a constraint spreadsheet developed as an add-in package for *OpenOffice.org Calc*.

- Useful as a front-end of a CSP solving system.
- Users can solve their problems by specifying the constraint variables and constraints on the spreadsheet.
- Easy to install and easy to use.
- Related works:
  Frontline systems (commercial), Knowledgesheet

Example of Calc/Cream

1		A	B	C	D
2		num.	materialA	materialB	profit
3	product1	0	=2*A3	=A3	=4*A3
4	product2	0	=A4	=A4	=3*A4
5	total		=SUM(B3:B4)	=SUM(C3:C4)	=SUM(D3:D4)
6	stock		100	60	
7	condition		=B5<=B6	=C5<=C6	

- Let us consider a previous simple problem.
- The above is a usual description on spreadsheet.
- We add three special functions:
  - CVARIABLES(*RANGE*;*INF*;*SUP*)
  - CONSTRAINTS(*RANGE*)
  - COBJECTIVE(MAX(*CELL*))

Example of Calc/Cream (Cont.)

1		A	B	C	D
2		num.	materialA	materialB	profit
3	product1	0	=2*A3	=A3	=4*A3
4	product2	0	=A4	=A4	=3*A4
5	total		=SUM(B3:B4)	=SUM(C3:C4)	=SUM(D3:D4)
6	stock		100	60	
7	condition		=B5<=B6	=C5<=C6	
8	=CVARIABLES(A3:A4;0;1000)				

- We add three special functions:
  - CVARIABLES(*RANGE*;*INF*;*SUP*)
    Specifies the cell locations of constraint variables.
  - CONSTRAINTS(*RANGE*)
  - COBJECTIVE(MAX(*CELL*))

Example of Calc/Cream (Cont.)

1		A	B	C	D
2		num.	materialA	materialB	profit
3	product1	0	=2*A3	=A3	=4*A3
4	product2	0	=A4	=A4	=3*A4
5	total		=SUM(B3:B4)	=SUM(C3:C4)	=SUM(D3:D4)
6	stock		100	60	
7	condition		=B5<=B6	=C5<=C6	
8	=CVARIABLES(A3:A4;0;1000)				
9	=CONSTRAINTS(B3:D7)				

- We add three special functions:
  - CVARIABLES(*RANGE*;*INF*;*SUP*)
  - CONSTRAINTS(*RANGE*)
    Specifies the cell locations of constraints.
  - COBJECTIVE(MAX(*CELL*))

Example of Calc/Cream (Cont.)

1		A	B	C	D
2		num.	materialA	materialB	profit
3	product1	0	=2*A3	=A3	=4*A3
4	product2	0	=A4	=A4	=3*A4
5	total		=SUM(B3:B4)	=SUM(C3:C4)	=SUM(D3:D4)
6	stock		100	60	
7	condition		=B5<=B6	=C5<=C6	
8	=CVARIABLES(A3:A4;0;100)				
9	=CONSTRAINTS(A3:D7)				
10	=COBJECTIVE(MAX(D5))				

- We add three special functions:
  - CVARIABLES(*RANGE*;*INF*;*SUP*)
  - CONSTRAINTS(*RANGE*)
  - COBJECTIVE(MAX(*CELL*))
    Specifies the cell locations of objective variable.

Example of Calc/Cream (Cont.)

1		A	B	C	D
2		num.	materialA	materialB	profit
3	product1	0	=2*A3	=A3	=4*A3
4	product2	0	=A4	=A4	=3*A4
5	total		=SUM(B3:B4)	=SUM(C3:C4)	=SUM(D3:D4)
6	stock		100	60	
7	condition		=B5<=B6	=C5<=C6	
8	=CVARIABLES(A3:A4;0;100)				
9	=CONSTRAINTS(A3:D7)				
10	=COBJECTIVE(MAX(D5))				

- We add three special functions:
  - CVARIABLES(*RANGE*;*INF*;*SUP*)
  - CONSTRAINTS(*RANGE*)
  - COBJECTIVE(MAX(*CELL*))
- All we have to do is push the ”Start” button.

g-HECS (2004–)

g-HECS is an on-going project developing a constraint solving system on the Grid.

- The HECS system supports heterogeneous constraint solvers, but is not enough for practical use.

- For the feasibility study, we implemented two prototype systems on the Grid using:
  - Globus toolkit
  - Sun Grid Engine
- In prototypes, constraint solver is implemented using only Cream, not heterogeneous.

Performance Results of two prototypes

JSSP (Goal)	Single	SGE	Globus
LA40 (1400)	37.3	35.9 (1.04)	23.6 (1.58)
LA40 (1350)	603.1	129.9 (4.64)	93.4 (6.53)
LA40 (1330)	1381.9	818.6 (1.69)	496.0 (2.79)
TA80 (2850)	85.7	54.1 (1.58)	38.5 (2.23)
TA80 (2800)	347.8	100.8 (3.45)	190.1 (1.83)
TA80 (2750)	1041.1	186.3 (5.59)	312.8 (3.33)
Average	582.3	220.9 (2.64)	192.2 (3.03)

- Figures are elapsed time in seconds (and speedup ratio) to find a solution for the given goal of two JSSPs.
- “Single” runs 3 solvers (SA, Taboo, IBB) on a single machine.
- “SGE” and “Globus” runs 12 solvers on 4 machines.

Design Goals of g-HECS system

- *Efficiency*
  Fast enough for practical use
- *Scalability*
  Faster with more machines
- *Reliability*
  Never hang-up
- *Portability*
  Globus, SGE; Cream, clp(fd), ILOG, SAT solvers, Mathematica
- *Usability*
  Web service, Spreadsheet front-end

Architecture Design of g-HECS system

- g-HECS consists of the following software components organizing a solver/space network:
  - *Slave solver*: solves a problem by alone.
  - *Master solver*: solves a problem by using several other sub-solvers.
  - *Tuple space*: is used for client-to-solver and solver-to-solver communication.
- Problems are described in a *CSP description language* (possibly in XML)..
- User can post the problem to a tuple space through a Web service. interface.