What I Have Learned From All These Solver Competitions

Neng-Fa Zhou

CUNY Brooklyn College & Graduate Center
zhou@sci.brooklyn.cuny.edu

In this talk, I would like to share my experiences gained from participating in four CSP solver competitions and the second ASP solver competition. In particular, I'll talk about how various programming techniques can make huge differences in solving some of the benchmark problems used in the competitions. These techniques include global constraints, table constraints, and problem-specific propagators and labeling strategies for selecting variables and values. I'll present these techniques with experimental results from B-Prolog and other CLP(FD) systems.

What I Have Learned From All These Solver Competitions

Neng-Fa Zhou
CUNY Brooklyn College and Graduate Center

Outline

- Preparations
  - CSP and CLP(FD)
  - Global constraints
  - Table constraints
  - Action rules
- Programming techniques
  - Using global constraints
  - Using table constraints
  - Using specialized propagators
  - Using problem-specific labeling strategies
- Conclusion

Constraint Satisfaction Problems

- CSP
  - *A set of variables $V=\{V_1,\ldots,V_n\}$*
  - *Each variable has a domain V_i :: D_i*
  - *A set of constraints*
- *Example*
    - A:{0,1}, B:{0,1}, C:{0,1}
    - C = A and B
- Solution to CSP
  - *An assignment of values to the variables that satisfies all the constraints*

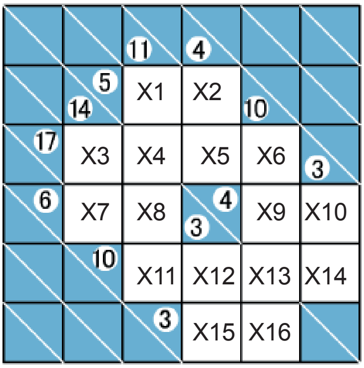
CLP(FD)

- CLP(FD) language
  - An extension of Prolog that provides built-ins for describing and solving CSPs
- CLP(FD) systems
  - B-Prolog, CHIP, ECLiPSe, GNU-Prolog, IF/Prolog, Prolog-IV, SICStus, SWI-Prolog, YAP, …

CLP(FD) - B-Prolog

- Domain constraints
    - X in D
    - X notin D
- Unification and arithmetic constraints
    - Exp R Exp
        - R is one of the following: #=, #\=, #>, #>=, #<, #=
        - Exp may contain +, -, *, /, //, mod, sum, min, max
- Boolean constraints
    - Exp R Exp
        - R is one of the following: #/\, #\/, #=>, #<=>, #\
- Global constraints
- Labeling built-ins

Example



A Kakuro puzzle

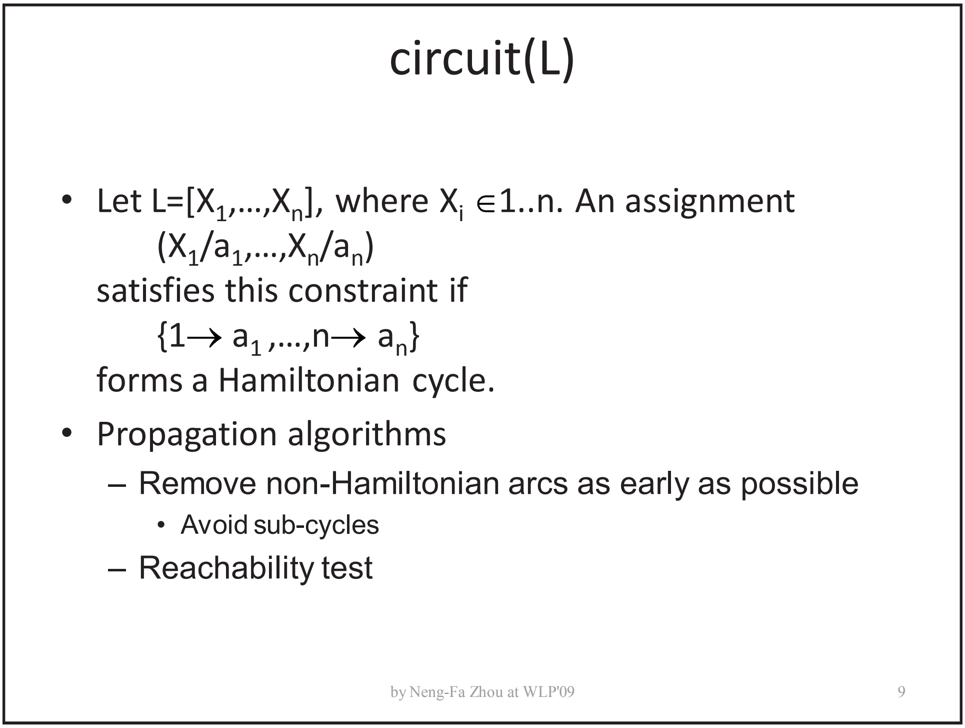
```
go:-
    Vars=[X1,X2,…,X16],
     Vars :: 1..9,
     word([X1,X2],5),
     word([X3,X4,X5,X6],17),
     …
     word([X10,X14],3),
     labeling(Vars),
     writeln(Vars).
word(L,Sum):-
     sum(L) #= Sum,
     all_different(L).
```

Global Constraints

- all_different(L)
- all_distinct(L)
- circuit(L)
- cumulative(Starts,Durations,Resources,Limit)

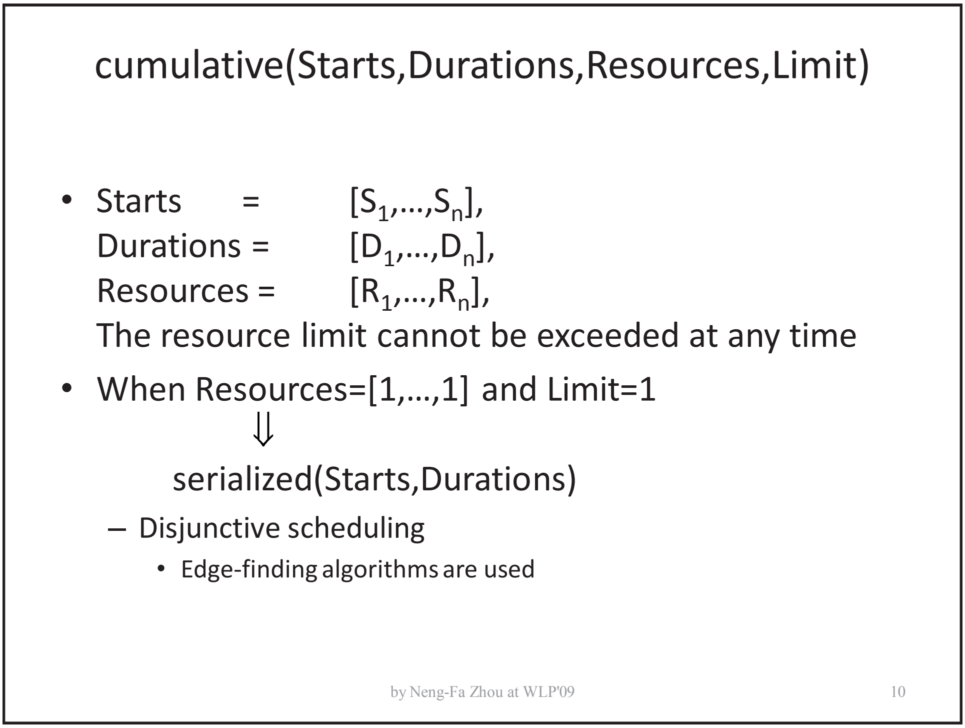
all_different(L) and all_distinct(L)

- all_different(L)
  - Let L=[X_1,…,X_n].
    For each i,j$\in$1..n (i<j) $X_i \neq X_j$.
- all_distinct(L)
  - Maintains some sort of hyper-arc consistency
    - Hall-set finding (B-Prolog and ECLiPSe)
    - Maximal-matching (SICStus)



circuit(L)

- Let L=[X_1,…,X_n], where $X_i \in 1..n$. An assignment
  (X_1/a_1,…,X_n/a_n)
  satisfies this constraint if
  {1→ a_1 ,…,n→ a_n}
  forms a Hamiltonian cycle.
- Propagation algorithms
  - Remove non-Hamiltonian arcs as early as possible
    - Avoid sub-cycles
  - Reachability test



cumulative(Starts,Durations,Resources,Limit)

- Starts = [S_1,…,S_n],
  Durations = [D_1,…,D_n],
  Resources = [R_1,…,R_n],
  The resource limit cannot be exceeded at any time
- When Resources=[1,…,1] and Limit=1
  $\Downarrow$
  serialized(Starts,Durations)
  - Disjunctive scheduling
    - Edge-finding algorithms are used

Table Constraints

- Positive constraints

```
(X,Y,Z) in [(0,1,1),
            (1,0,1),
            (1,1,0)]
```

- Negative constraints

```
(X,Y,Z) notin [(0,1,1),
               (1,0,1),
               (1,1,0)]
```

Action Rules

Agent, Condition, {EventSet} => Action

- Events
  - Instantiation: `ins(X)`
  - Domain
    - `bound(X)`,`dom(X)`,`dom(X,E)`,`dom_any(X)`, and `dom_any(X,E)`
  - Time : `time(X)`
  - GUI
    - `actionPerformed(X)`,`mouseClicked(X,E)`…
  - General
    - `event(X,O)`

Applications of Action Rules

- Lazy evaluation

```
freeze(X,G), var(X), {ins(X)} => true.
freeze(X,G) => call(G).
```

- Constraint propagators

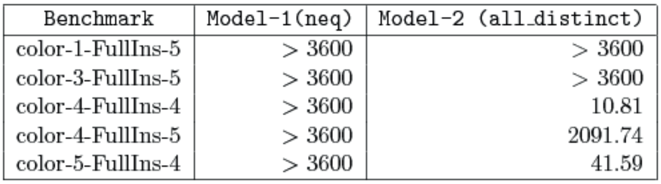
```
'X in C-Y_ac'(X,Y,C),var(X),var(Y),
      {dom(Y,Ey)}
      =>
      Ex is C-Ey,
      exclude(X,Ex).
'X in C-Y_ac'(X,Y,C) => true.
```

Using Global Constraints
all_distinct(L) (1)

- Graph coloring
  - Model-1 (neq)
    - For each two neighbors i and j, $C_i \neq C_j$
  - Model-2 (all_distinct)
    - For each complete subgraph $\{i_1,i_2,\ldots,i_k\}$, all_distinct($[Ci_1, Ci_2,\ldots, Ci_k]$)
    - post_neqs(Neqs) in B-Prolog

Using Global Constraints
all_distinct(L) (2)

- Benchmarking results (seconds)

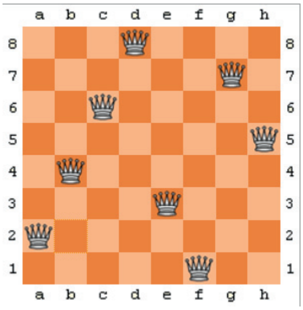


Benchmark	Model-1(neq)	Model-2 (all_distinct)
color-1-FullIns-5	> 3600	> 3600
color-3-FullIns-5	> 3600	> 3600
color-4-FullIns-4	> 3600	10.81
color-4-FullIns-5	> 3600	2091.74
color-5-FullIns-4	> 3600	41.59

Source of benchmarks:
Agostino Dovier, Andrea Formisano, and Enrico Pontelli,
A comparison of CLP(FD) and ASP solutions to NP-complete problems,
ICLP’05.

Using Global Constraints
all_distinct(L) (3)

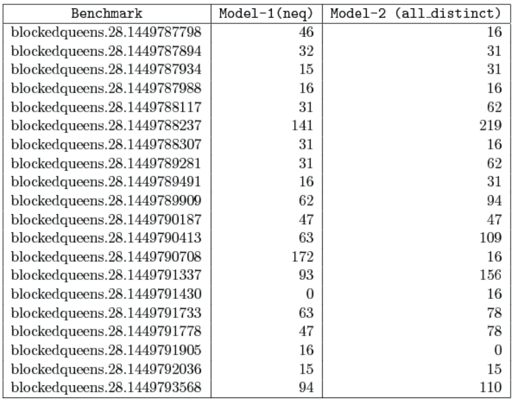
- N-Queens problem



- Model-1 (neq)
  - For i,j∈1..n (i < j)
    $Q_i \neq Q_i$
    $Q_i-Q_j \neq (j-i)$
    $Q_j-Q_i \neq (j-i)$
- Model-2 (all_distinct)
  - all_distinct([Q_1,…,Q_n]),
    all_distinct([Q_1,Q_2-1,…,Q_n-n]),
    all_distinct([Q_1,Q_2+1,…,Q_n+n])

Using Global Constraints
all_distinct(L) (4)

- Benchmarking results (ms)

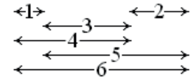


Benchmark	Model-1(neq)	Model-2 (all_distinct)
blockedqueens.28.1449787798	46	16
blockedqueens.28.1449787894	32	31
blockedqueens.28.1449787934	15	31
blockedqueens.28.1449787988	16	16
blockedqueens.28.1449788117	31	62
blockedqueens.28.1449788237	141	219
blockedqueens.28.1449788307	31	16
blockedqueens.28.1449789281	31	62
blockedqueens.28.1449789491	16	31
blockedqueens.28.1449789909	62	94
blockedqueens.28.1449790187	47	47
blockedqueens.28.1449790413	63	109
blockedqueens.28.1449790708	172	16
blockedqueens.28.1449791337	93	156
blockedqueens.28.1449791430	0	16
blockedqueens.28.1449791733	63	78
blockedqueens.28.1449791778	47	78
blockedqueens.28.1449791905	16	0
blockedqueens.28.1449792036	15	15
blockedqueens.28.1449793568	94	110

Source of benchmarks:
2nd ASP Competition

Using Global Constraints
all_distinct(L) (5)

- Some times all_different(L) is faster than all_distinct(L)
- An example: Golomb ruler



```
all_different([M2-M1,M3-M1,M4-M1,
               M3-M2,M4-M2,
               M4-M3])
```

Using Global Constraints
all_distinct(L) (6)

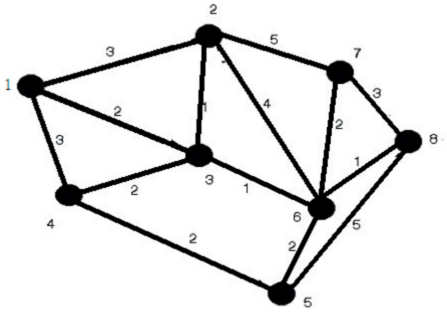
- Benchmarking results (seconds)

Benchmark	all_different	all_distinct
golomb-8-positions-100-16	0.03	0.05
golomb-8-positions-50-16	0.03	0.06
golomb-10-positions-100-36	2.20	5.09
golomb-10-positions-125-36	2.20	5.11
golomb-10-positions-75-36	2.19	5.11
golomb-11-positions-100-35	47.72	125.39
golomb-11-positions-125-35	47.22	125.87
golomb-11-positions-75-35	45.45	121.12
golomb-12-positions-100-48	476.53	> 600
golomb-12-positions-125-48	478.11	> 600
golomb-12-positions-150-48	479.34	> 600

Source of benchmarks: 2nd ASP Competition

Using Global Constraints
circuit(L)

- The Traveling Salesperson Problem



```
tsp(Vars):-
      Vars=[V1,V2,…,V8],
      V1 :: [2,3,4],
      V2 :: [1,3,6,7],
      …
      V8 :: [5,6,7],
      circuit(Vars).
```

Using Global Constraints serialized(Starts,Durations) (1)

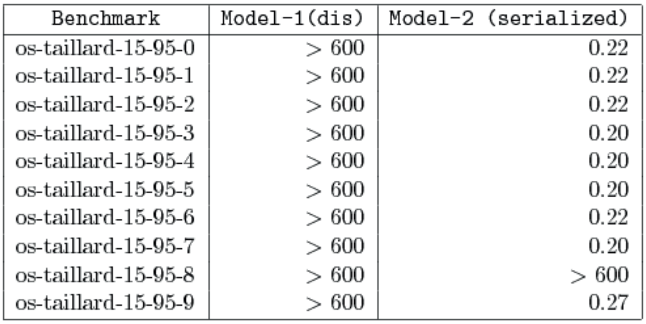
- Scheduling
  - Model-1: use disjunctive constraints

```
VV5+97#=<VV17#\/VV17+52#=<VV5,
VV5+97#=<VV26#\/VV26+59#=<VV5,
VV5+97#=<VV32#\/VV32+41#=<VV5,
VV5+97#=<VV49#\/VV49+63#=<VV5,
                           …..
```

  - Model-2: use global constraints
    - `post_disjunctive_tasks(Disjs)` in B-Prolog
      `Disjs=[disj_tasks(S1,D1,S2,D2),…]`
      - converts disjunctive constraints into serialized

Using Global Constraints serialized(Starts,Durations) (2)

- Benchmarking results



Benchmark	Model-1(dis)	Model-2 (serialized)
os-taillard-15-95-0	> 600	0.22
os-taillard-15-95-1	> 600	0.22
os-taillard-15-95-2	> 600	0.22
os-taillard-15-95-3	> 600	0.20
os-taillard-15-95-4	> 600	0.20
os-taillard-15-95-5	> 600	0.20
os-taillard-15-95-6	> 600	0.22
os-taillard-15-95-7	> 600	0.20
os-taillard-15-95-8	> 600	> 600
os-taillard-15-95-9	> 600	0.27

Source of benchmarks:
www.cril.univ-artois.fr/~lecoutre/research/benchmarks/benchmarks.html

Using Table Constraints (1)

- The Schur number problem
  - Partition n positive integers into m sets such that all of the sets are sum-free.
- Model-1 (sum-free triplet)

$$S_i = S_j \rightarrow S_{i+j} \neq S_i$$

- Mode-2 (use redundant constraints)

$$S_i{=}S_j \rightarrow S_{i+j}{\neq}S_i,\ S_i{=}S_{i+j} \rightarrow S_j{\neq}S_i,\ S_j{=}S_{i+j} \rightarrow S_i{\neq}S_j.$$

- Mode-3 (use table constraints)

$$(S_i,S_j,S_{i+j})\ \texttt{notin}\ [(1,1,1),(2,2,2),\ldots]$$

Using Table Constraints (2)
The Schur number problem

- Benchmarking results (seconds)

Benchmark	Model-1	Model-2 (redundant)	Model-3 (table)
15.1.schur.lp	> 600	> 600	441.72
15.10.schur.lp	> 600	> 600	432.56
15.14.schur.lp	> 600	> 600	> 600
15.16.schur.lp	> 600	> 600	> 600
15.19.schur.lp	> 600	> 600	> 600
15.20.schur.lp	> 600	> 600	328.96
15.3.schur.lp	> 600	> 600	252.20
15.4.schur.lp	> 600	> 600	> 600
15.5.schur.lp	> 600	> 600	393.90

Source of benchmarks: 2nd ASP Competition

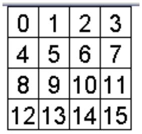
Using Table Constraints (3)
The Knights Problem

- Model-1: use disjunctive constraints

```
(abs(P1//N-P2//N)#=1 #/\ abs(P1 mod N-P2 mod N)#=2) #\/
(abs(P1//N-P2//N)#=2 #/\ abs(P1 mod N-P2 mod N)#=1)
```

- Model-2: use table constraints

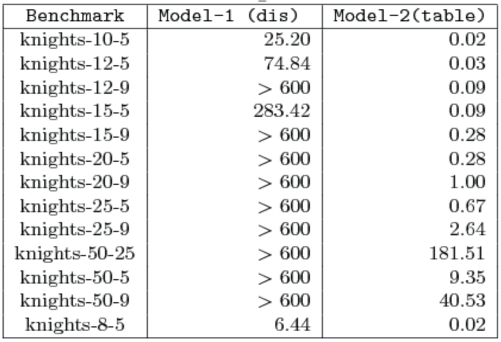
```
(P1,P2) in [(0,6),(0,9),(1,7),(1,8),(1,10),…]
```



0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Using Table Constraints (4)
The Knights Problem

- Benchmarking results (seconds)



Benchmark	Model-1 (dis)	Model-2(table)
knights-10-5	25.20	0.02
knights-12-5	74.84	0.03
knights-12-9	> 600	0.09
knights-15-5	283.42	0.09
knights-15-9	> 600	0.28
knights-20-5	> 600	0.28
knights-20-9	> 600	1.00
knights-25-5	> 600	0.67
knights-25-9	> 600	2.64
knights-50-25	> 600	181.51
knights-50-5	> 600	9.35
knights-50-9	> 600	40.53
knights-8-5	6.44	0.02

Source of benchmarks:
www.cril.univ-artois.fr/~lecoutre/research/benchmarks/benchmarks.html

Not Using Table Constraints (1)

- The Black Hole problem

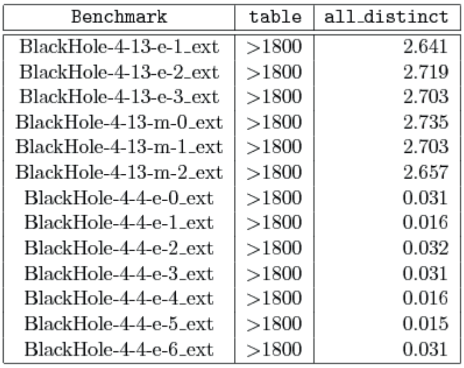
```
(X,Y) notin [(0,0),(1,1),(2,2),…]
```

- Transform table constraints

```
(X,Y) notin [(0,0),(1,1),(2,2),…],
                    ↓
                  X #\= Y
                    ↓
           all_distinct(…)
```

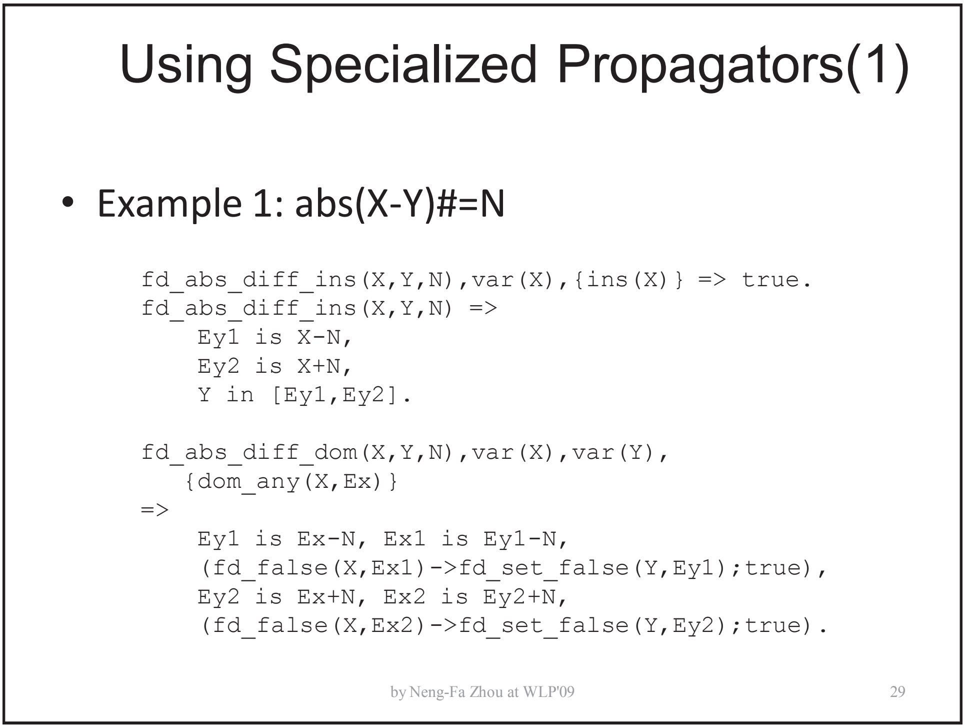
Not Using Table Constraints (2)
The Black Hole problem

- Benchmarking results (seconds)



Benchmark	table	all_distinct
BlackHole-4-13-e-1_ext	>1800	2.641
BlackHole-4-13-e-2_ext	>1800	2.719
BlackHole-4-13-e-3_ext	>1800	2.703
BlackHole-4-13-m-0_ext	>1800	2.735
BlackHole-4-13-m-1_ext	>1800	2.703
BlackHole-4-13-m-2_ext	>1800	2.657
BlackHole-4-4-e-0_ext	>1800	0.031
BlackHole-4-4-e-1_ext	>1800	0.016
BlackHole-4-4-e-2_ext	>1800	0.032
BlackHole-4-4-e-3_ext	>1800	0.031
BlackHole-4-4-e-4_ext	>1800	0.016
BlackHole-4-4-e-5_ext	>1800	0.015
BlackHole-4-4-e-6_ext	>1800	0.031

www.cril.univ-artois.fr/~lecoutre/research/benchmarks/benchmarks.html

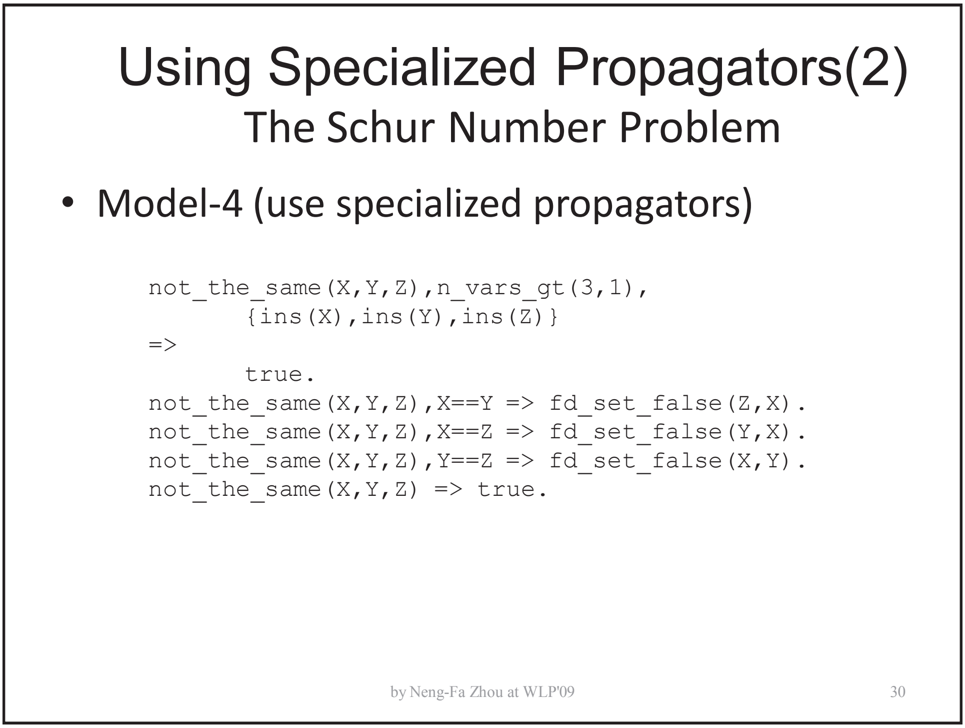


Using Specialized Propagators(1)

- Example 1: abs(X-Y)#=N

```
fd_abs_diff_ins(X,Y,N),var(X),{ins(X)} => true.
fd_abs_diff_ins(X,Y,N) =>
    Ey1 is X-N,
    Ey2 is X+N,
    Y in [Ey1,Ey2].

fd_abs_diff_dom(X,Y,N),var(X),var(Y),
   {dom_any(X,Ex)}
=>
    Ey1 is Ex-N, Ex1 is Ey1-N,
    (fd_false(X,Ex1)->fd_set_false(Y,Ey1);true),
    Ey2 is Ex+N, Ex2 is Ey2+N,
    (fd_false(X,Ex2)->fd_set_false(Y,Ey2);true).
```



Using Specialized Propagators(2)

The Schur Number Problem

- Model-4 (use specialized propagators)

```
not_the_same(X,Y,Z),n_vars_gt(3,1),
       {ins(X),ins(Y),ins(Z)}
=>
       true.
not_the_same(X,Y,Z),X==Y => fd_set_false(Z,X).
not_the_same(X,Y,Z),X==Z => fd_set_false(Y,X).
not_the_same(X,Y,Z),Y==Z => fd_set_false(X,Y).
not_the_same(X,Y,Z) => true.
```

Using Specialized Propagators(3)

- Benchmarking results (seconds)

Benchmark	Model-3 (table)	Model-4 (specialized)
15.1.schur.lp	441.72	155.23
15.10.schur.lp	432.56	256.78
15.14.schur.lp	> 600	435.89
15.16.schur.lp	> 600	348.46
15.19.schur.lp	> 600	486.20
15.20.schur.lp	328.96	128.00
15.3.schur.lp	252.20	106.32
15.4.schur.lp	> 600	> 600
15.5.schur.lp	393.90	138.04

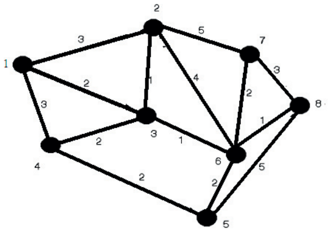
Source of benchmarks: 2nd ASP Competition

Using Problem-Specific Labeling Strategies

- Variable selection
  - queens: labeling([ff],Vars)
  - golomb: labeling([],Vars)
- Value selection
  - tsp: select an edge with the lowest weight

Value Selection

- Select an edge with the lowest weight



```
tsp(Vars):-
        Vars=[V1,V2,…,V8],
        V1 :: [2,3,4],
       put_attr_no_hook(V1,nbs,[3,2,4]),
       V2 :: [1,3,6,7],
       put_attr_no_hook(V2,nbs,[3,1,6,7]),
        …
        V8 :: [5,6,7],
       pub_attr_no_book(V8,nbs,[6,7,5]),
        circuit(Vars).

Use get_attr(V,nbs,Nbs),member(V,Nbs)
     rather than indomain(V) to label V.
```

Conclusion

- Techniques
  - Using global constraints
  - Using table constraints
  - Using specialized propagators
  - Using problem-specific labeling strategies
- More techniques and systems to explore
  - Integrating CLP(FD) with SAT and ASP solvers
- Thanks!
  - Organizers of the solver competitions
  - Program committee of WLP’09