OPPA European Social Fund
Prague & EU: We invest in your future.

Combinatorial Optimization

Zdeněk Hanzálek
hanzalek@fel.cvut.cz

CTU FEE Department of Control Engineering

May 3, 2013

European Social Fund Prague & EU: We invests in your future.

Constraint Programming

Zdeněk Hanzálek, Jan Kelbel
hanzalek@fel.cvut.cz

CTU FEE Department of control engineering

May 16, 2011

1. Ipspiration - Sudoku

2. Constraint Satisfaction Problem (CSP)
   - Search and Propagation
   - Arc consistency
     - AC-3 Algorithm
   - Global constraints

What is Constraint Programming?

- Sudoku is Constraint Programming

Motivation - Sudoku

			2		5			
	9					7	3	
		2			9		6	
2						4		9
				7				
6		9						1
	8		4			1		
	6	3					8	
			6		8			

Assign digits to blank fields such that:

digits distinct per rows, columns, blocks

Sudoku

			2		5			
	9					7	3	
		2			9		6	
2						4		9
				7				
6		9						1
	8		4			1		
	6	3					8	
			6		8			

Assign digits to blank fields such that:

digits distinct per **rows**, columns, blocks

Sudoku

			2		5			
	9					7	3	
		2			9		6	
2						4		9
				7				
6		9						1
	8		4			1		
	6	3					8	
			6		8			

Assign digits to blank fields such that:

digits distinct per rows, **columns**, blocks

Sudoku

			2		5			
	9					7	3	
		2			9		6	
2						4		9
				7				
6		9						1
	8		4			1		
	6	3					8	
			6		8			

Assign digits to blank fields such that:

digits distinct per rows, columns, **blocks**

Sudoku - propagation in the lower left block

	8	
	6	3

No blank field in the block can have value of 3,6,8

Sudoku - propagation in the lower left block

1,2,4,5,7,9	8	1,2,4,5,7,9
1,2,4,5,7,9	6	3
1,2,4,5,7,9	1,2,4,5,7,9	1,2,4,5,7,9

No blank field in the block can have value of 3,6,8

- propagate to all blank fields

Use the same propagation for rows and columns

Sudoku - propagation in one field

			2		5			
	9					7	3	
		2			9		6	
2						4		9
				7				
6		9						1
	8		4			1		
	6	3					8	
			6		8			

Prune digits from fields such that:

digits distinct per rows, columns, blocks

Sudoku - propagation in one field

			2		5			
	9					7	3	
		2			9		6	
2						4		9
				7				
6		9						1
	8		4			1		
	6	3					8	
			6		8			

1,3,5,6,7,8

Prune digits from fields such that:

digits distinct per **rows**, columns, blocks

Sudoku - propagation in one field

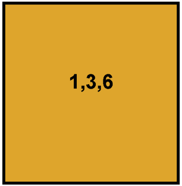
			2		5			
	9					7	3	
		2			9		6	
2						4		9
				7				
6		9						1
	8		4			1		
	6	3					8	
			6		8			

Prune digits from fields such that:

digits distinct per rows, **columns**, blocks

Sudoku - propagation in one field

			2		5			
	9					7	3	
		2			9		6	
2						4		9
				7				
6		9						1
	8		4			1		
	6	3					8	
			6		8			



Prune digits from fields such that:

digits distinct per rows, columns, **blocks**

Sudoku - iterated propagation

			2		5			
	9					7	3	
		2			9		6	
2						4		9
				7				
6		9						1
	8		4			1		
	6	3					8	
			6		8			

- **Iterate propagation** for rows, columns and blocks
  - When to stop?
  - What if more assignments exist?
  - What if no assignment exists?

Sudoku is constraint programming

			2		5			
	9					7	3	
		2			9		6	
2						4		9
				7				
6		9						1
	8		4			1		
	6	3					8	
			6		8			

Sudoku:

- **Variables** - fields
  - assign values - digits
  - maintain **domain** of variable - set of possible values
- **Constraints** - numbers in row, column and box must vary
  - relations among variables disable certain combinations of values

Constraint programming is **declarative programming**:

- **Model**: variables, domains, constraints
- **Solver**: propagation, searching

Constraint Satisfaction Problem - formally

Constraint Satisfaction Problem (CSP) is defined by triplet (X, D, C), where:

- $X = \{x_1, \dots, x_n\}$ is finite set of variables
- $D = \{D_1, \dots, D_n\}$ is finite set of domains of variables
- $C = \{C_1, \dots, C_t\}$ is finite set of constraints.

Domain $D_i = \{v_1, \dots, v_k\}$ is **finite** set of all possible values of x_i.

Constraint C_i is couple (S_i, R_i) where $S_i \subseteq X$ and R_i **is relation** relation over the set of variables S_i. For $S_i = \{x_{i_1}, \dots, x_{i_r}\}$ is $R_i \subseteq D_{i_1} \times \dots \times D_{i_r}$.

CSP is NP-complete problem.

Terminology - CSP, CSOP, Constraint Solving and CP

- **Solution** to (**CSP**) is complete **assignment of values** from domains to variables such that **all constraints are satisfied**
  - it is a decision problem.
- Constraint Satisfaction Optimization Problem (**CSOP**) is defined by $(X, D, C, f(X))$ where $f(X)$ is objective function. The search is not finished, when the first acceptable solution was found, but it is finished when the **optimal solution** was found (using branch&bound method for example).
- Constraint Solving is defined by (X, D, C) where D_i is defined on $\mathbb{R}$ (e.g. solution of the set of linear equations-inequalities).
- Constraint Programming, **CP** includes Constraint Satisfaction and Constraint Solving.

How it works - Search and Propagation

Example: $x \in \{3, 4, 5\}$, $y \in \{3, 4, 5\}$, $x \geq y$, $y > 3$

How it works - Search and Propagation

Example: $x \in \{3, 4, 5\}$, $y \in \{3, 4, 5\}$, $x \geq y$, $y > 3$

1. propagate $y > 3$: $x \in \{3, 4, 5\}$, $y \in \{4, 5\}$

How it works - Search and Propagation

Example: $x \in \{3,4,5\}$, $y \in \{3,4,5\}$, $x \geq y$, $y > 3$

1. propagate $y > 3$: $x \in \{3,4,5\}$, $y \in \{4,5\}$
2. propagate $x \geq y$: $x \in \{4,5\}$, $y \in \{4,5\}$

How it works - Search and Propagation

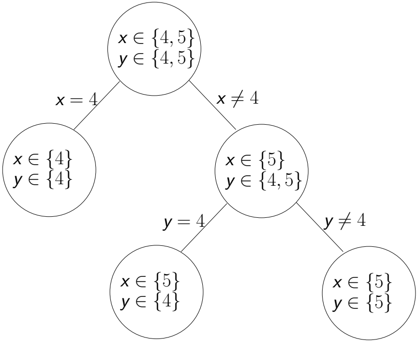
Example: $x \in \{3,4,5\}$, $y \in \{3,4,5\}$, $x \geq y$, $y > 3$

1. propagate $y > 3$: $x \in \{3,4,5\}$, $y \in \{4,5\}$
2. propagate $x \geq y$: $x \in \{4,5\}$, $y \in \{4,5\}$
3. propagation alone is not enough
   - product of the domains (incl. $x = 4$, $y = 5$) is a superset of solution
   - the search helps - we create subproblems

How it works - Search and Propagation

Example: $x \in \{3,4,5\}$, $y \in \{3,4,5\}$, $x \geq y$, $y > 3$

1. propagate $y > 3$: $x \in \{3,4,5\}$, $y \in \{4,5\}$
2. propagate $x \geq y$: $x \in \{4,5\}$, $y \in \{4,5\}$
3. propagation alone is not enough
   - product of the domains (incl. $x = 4$, $y = 5$) is a superset of solution
   - the search helps - we create subproblems
4. in subproblems we use propagation again



- The **search** can be driven by **various means** (order of the variables, division of domain/domains).
- By **propagation** of constraints we **filter domains** of variables.

Comparison with ILP

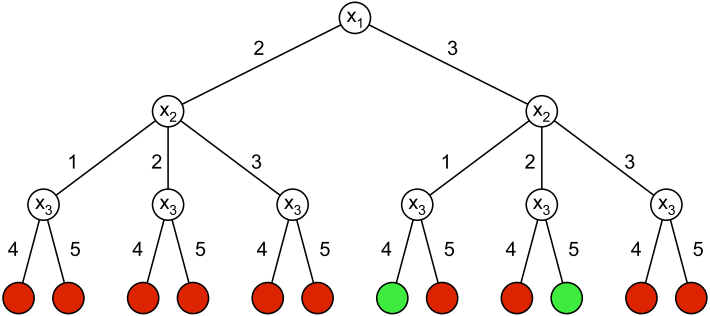
- In both cases we deal with declarative programming
- Performance differs from problem to problem
- CSP allows to formulate **complex constraints**
  (ILP uses inequalities only, CSP uses arbitrary relation - e.g. binary relation may be given by a set of compatible tuples)
  - CSP is more flexible, formulation is easier to understand
- it is difficult to represent continuous problems by CSP
  - finiteness of domains can be bypassed by using hybrid approaches - e.g. combination with LP
- CP is new technique, it is more open

Example: Search and Propagation

Complete search (for example Depth First Search):

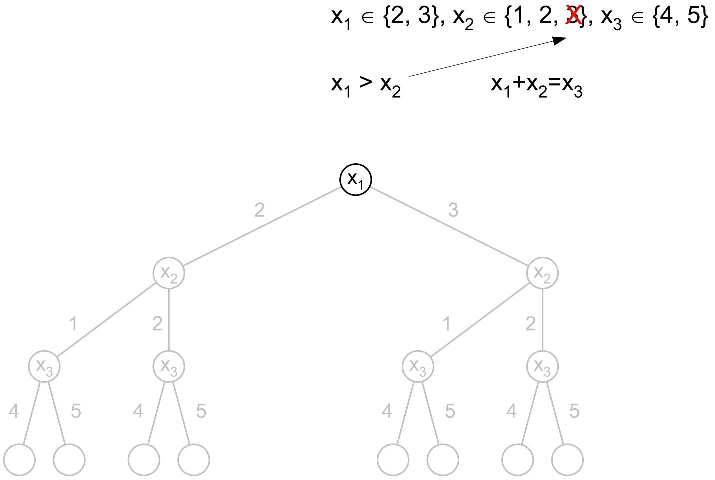
$x_1 \in \{2, 3\}, x_2 \in \{1, 2, 3\}, x_3 \in \{4, 5\}$

$x_1 > x_2 \qquad x_1+x_2=x_3$



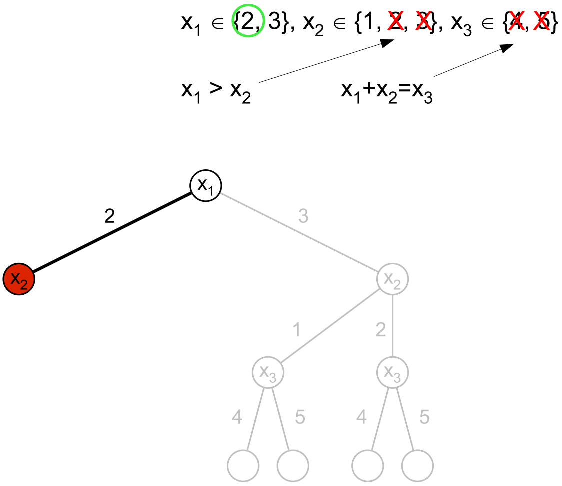
Example: Search and Propagation

Initial propagation of constraints:

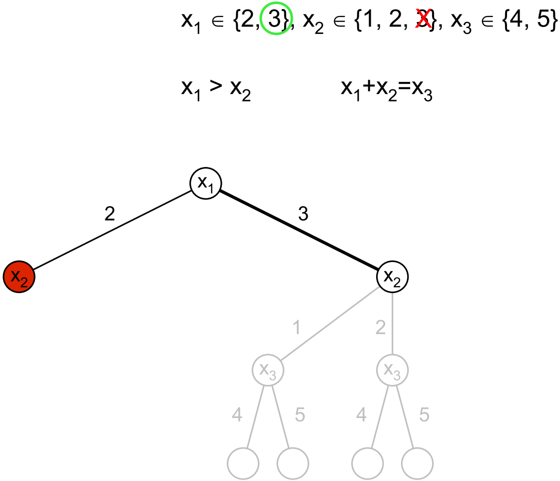


Example: Search and Propagation

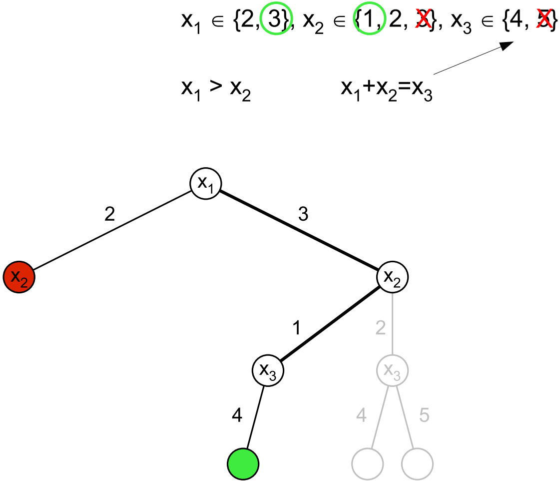
Choose $x_1 = 2$ and propagate constraints:



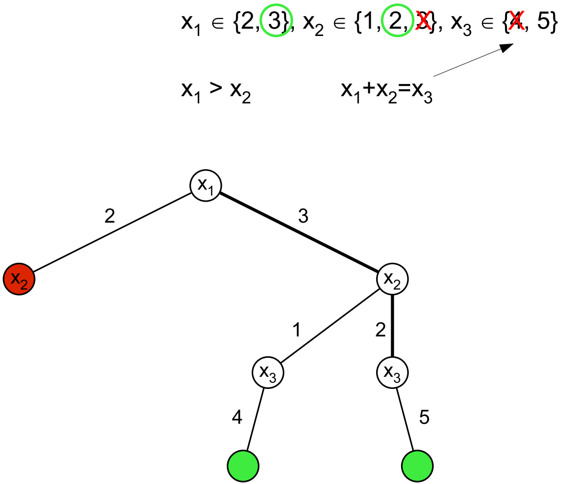
Choose $x_1 = 3$ and propagate constraints:



Choose $x_2 = 1$ and propagate constraints:



Choose $x_2 = 2$ and propagate constraints:



Arc consistency

We will continue to consider only **binary CSP**, where every constraint is binary relation

- general (n-ary) CSP can be converted to binary CSP
- binary CSP can be represented by **digraph** G
  - nodes are variables
  - if there is a constraint involving x_i, x_j, then the nodes x_i, x_j are interconnected by arcs (x_i, x_j) and (x_j, x_i)

Arc consistency is an essential method for propagation.

- Arc (x_i, x_j) is **Arc Consistent, AC** iff for each value $a \in D_i$ there exists value $b \in D_j$ such that assignment $x_i = a, x_j = b$ meets all binary constraints for variables x_i, x_j.
- A **CSP is arc consistent** if all arc are arc consistent.
- Note that AC is **oriented** - consistence of arc (x_i, x_j) does not guarantee consistence of arc (x_j, x_i).

There are other local consistencies (path consistency, k-consistency, singleton arc consistency,...). Some of them are stronger some are weaker.

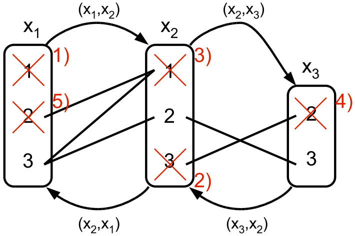
REVISE procedure

From domain D_i delete any value a, which is not consistent with domain D_j.

```
procedure REVISE
Input: Indexes i, j. Revised domain D_i. Domain D_j.
       Set of constraints C.
Output: Binary variable deleted indicating deletion of some value
        from D_i. Revised domain D_i.

deleted := 0;
for a ∈ D_i do
    if there is no b ∈ D_j ; x_i = a, x_j = b satisfies all constraints on x_i, x_j
    then
        D_i := D_i \ a;                    // delete a from D_i
        deleted := 1;
    end
end
```

Example: application of REVISE

CSP with variables $X = \{x_1, x_2, x_3\}$, constraints $x_1 > x_2,\ x_2 \neq x_3,\ x_2 + x_3 > 4$, and domains $D_1 = \{1, 2, 3\}, D_2 = \{1, 2, 3\}, D_3 = \{2, 3\}$.



revised arc	deleted	revised domain	(x_1, x_2)	(x_2, x_1)	(x_2, x_3)	(x_3, x_2)
(x_1, x_2)	$1^{1)}$	$D_1 = \{2, 3\}$	consist	nonconsist	nonconsist	consist
(x_2, x_1)	$3^{2)}$	$D_2 = \{1, 2\}$	consist	consist	nonconsist	nonconsist
(x_2, x_3)	$1^{3)}$	$D_2 = \{2\}$	nonconsist	consist	consist	nonconsist
(x_3, x_2)	$2^{4)}$	$D_3 = \{3\}$	nonconsist	consist	consist	consist

After revision, some the arcs are still **nonconsistent**

- the reason is that some of the domains have been reduced
- continue in revision till all arc are consistent (without consistence check - see AC-3)

revised arc	deleted	revised domain	(x_1, x_2)	(x_2, x_1)	(x_2, x_3)	(x_3, x_2)
(x_1, x_2)	$2^{5)}$	$D_1 = \{3\}$	consist	consist	consist	consist

Arc Consistency - AC-3 algorithm

Maintain a queue of arcs to be revised (the arc is added into queue only if it's consistency could have been affected by reduction of the domain).

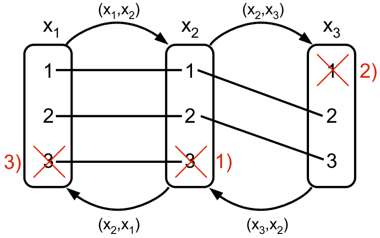
```
procedure AC-3
Input: X, D, C and graph G.
Output: Binary variable fail indicating no solution in this part of the
        state space. The set of revised domains D.

fail = 0; Q := E(G);                    // initialize Q by arcs of G
while Q ≠ ∅ do
    select and delete arc (x_k, x_m) from Q;
    (deleted, D_k)=REVISE(k, m, D_k, D_m, C);
    if deleted then
        if D_k = ∅ then fail = 1 and EXIT ;
        Q := Q ∪ {(x_i, x_k) such that (x_i, x_k) ∈ E(G) and i ≠ m};
    end
end
```

Note: revision of (x_k, x_m) does not change arc consistency of (x_m, x_k).

Example: iteration of AC-3

CSP with variables $X = \{x_1, x_2, x_3\}$, constraints $x_1 = x_2,\ x_2 + 1 = x_3$ and domains $D_1 = \{1,2,3\}, D_2 = \{1,2,3\}, D_3 = \{1,2,3\}$.

Initialization: $Q = \{(x_1,x_2),(x_2,x_1),(x_2,x_3),(x_3,x_2)\}$
revise (x_1,x_2)
$D_1 = \{1,2,3\}, D_2 = \{1,2,3\}, D_3 = \{1,2,3\}$
$Q = \{(x_2,x_1),(x_2,x_3),(x_3,x_2)\}$
revise (x_2,x_1)
$D_1 = \{1,2,3\}, D_2 = \{1,2,3\}, D_3 = \{1,2,3\}$
$Q = \{(x_2,x_3),(x_3,x_2)\}$
revise (x_2,x_3)
$D_1 = \{1,2,3\}, D_2 = \{1,2\}^{1)}, D_3 = \{1,2,3\}$
$Q = \{(x_3,x_2),(\mathbf{x_1},\mathbf{x_2})\}$
revise (x_3,x_2)
$D_1 = \{1,2,3\}, D_2 = \{1,2\}, D_3 = \{2,3\}^{2)}$
$Q = \{(x_1,x_2)\}$
revise (x_1,x_2)
$D_1 = \{1,2\}^{3)}, D_2 = \{1,2\}, D_3 = \{2,3\}$
$Q = \emptyset$



Global constraints

Global constraint

- capture **specific structure** of the problem
- use this structure to efficient propagation using **specialized propagation algorithm**

Example: On set $X = \{x_1, \ldots, x_n\}$ we apply constraint $x_i \neq x_j \; \forall i \neq j$

- This can be formulated by $(n^2 - n)/2$ disequalities.
- Second option is global constraint **alldifferent**, which uses a matching algorithm in bipartite graph, where one side represents variables and the other side represents values.

Other examples of global constraints:

- scheduling (edge-finder)
- graph algorithms (clique, cycle)
- finite state machine
- bin-packing

Tools for solving CSP

Proprietary:

- SICStus Prolog
- ILOG CP, CP Optimizer (C++)
- ILOG OPL Studio (OPL)
- Koalog (Java)

Open source:

- ECLiPSe (Prolog)
- Gecode (C++)
- Choco Solver (Java)
- Python constraints

References

Roman Barták.
Programování s omezujícími podmínkami.
http://kti.ms.mff.cuni.cz/~bartak/podminky/index.html, 2010.

Rina Dechter.
Constraint Processing.
Morgan Kaufmann, 2003.

Christian Schulte.
Constraint programming.
http://www.ict.kth.se/courses/ID2204/index.html, 2010.

OPP A PRA HA PRA GUE PRA GA PRA G EVROPSKÁ UNIE

OPPA European Social Fund
Prague & EU: We invest in your future.