Combinatorial Optimization Problems and Examples

Massimiliano Cattafi

Department of Engineering
University of Ferrara

The background

- Cooperation of:
  - Artificial Intelligence (in particular CLP)
  - Operational Research
  - Other fields (e.g. Civil Engineering)
- Applying knowledge in order to:
  - Better solve concrete problems
  - Compare approaches
  - Give feedback to new research (e.g. QCSPs)

Combinatorial Optimization

- Find an integer assignment to some variables such that:
  - Certain constraints are satisfied
  - The value of a certain function is minimized (or maximized)

Example: the knapsack problem

- n items in a set I
- Each one has a value v_i and a weight w_i
- A knapsack with a bearable weight limit WL
- Find the set $I' \subseteq I$ which maximizes the total value of taken items without 'breaking' the knapsack

Modeling the knapsack problem in CLP

```
knapsack(WL, V, W, S) :-
    length(V, LV),
    length(S, LV),
    S ins 0..1,
    % value to be constrained
    scalar_product(W, S, TW),
    TW #=< WL,% constraint
    % function to be maximized
    scalar_product(V, S, TV),
    once(labeling([max(TV)], S)).% maximization

?- knapsack(10, [1,1,2,3,4], [1,2,4,4,3], S).
S = [1, 1, 0, 1, 1].
```

Modeling the knapsack problem in CHR

Making domains and initialising values

```
item(V,W) <=> in(item(V,W),[0,1]).

knapsack(WL) <=>
    retractall(memlistitems(X)),
    retractall(memtotalv(X)),
    asserta(memlistitems([])), asserta(memtotalv(0)),
    wlimit(WL), ptotalw(0),
    ptotalv(0), listitems([]),
    labelk, labelmem.
```

Modeling the knapsack problem in CHR

Constraint on weight

```
wlimit(WL), ptotalw(TW) \ in(item(V,W),[0,1]) <=>
        TW2 is TW+W, TW2 > WL | in(item(V,W),[0]).
```

Modeling the knapsack problem in CHR

Updating weight, updating value, collecting solution and labeling

```
wlimit(WL) \ in(item(V,W),[1]), ptotalw(TW),
    ptotalv(TV), listitems(L) <=>
        TW2 is TW+W, TW2 =< WL |
            ptotalw(TW2), TV2 is TV+V, ptotalv(TV2),
            listitems([item(V,W) | L]).

labelk \ in(item(V,W), [0,1]) <=>
    member(P,[0,1]), in(item(V,W),[P]).
```

Modeling the knapsack problem in CHR

Compare solution with previous ones, take suitable choice

```
labelmem \ listitems(L), ptotalv(TV) <=>
    memtotalv(TV1), TV > TV1, memlistitems(L1) |
        retract(memtotalv(TV1)),
        asserta(memtotalv(TV)),
        retract(memlistitems(L1)),
        asserta(memlistitems(L)),
        print('best set '), print(L), nl,
        print('with value '), print(TV), nl, nl,
        fail.

labelmem \ ptotalv(TV) <=>
    memtotalv(TV1), TV =< TV1 | fail.
```

Modeling the knapsack problem in CHR

Example goal

```
?- item(1,1), item(1,2), item(2,4),
    item(3,4), item(4,3), knapsack(10).

[...]
best set
[item(1, 1), item(1, 2), item(3, 4), item(4, 3)]
with value 9
```

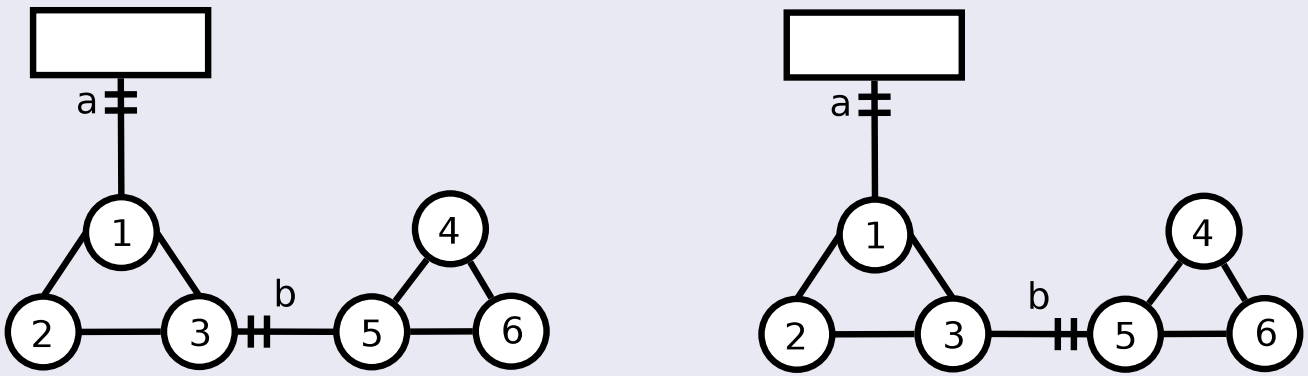
Time comparison

	10				20			
	CLP		CHR		CLP		CHR	
	a	b	a	b	a	b	a	b
5	0.026	0.063	0.013	0.011	0.029	0.048	0.012	0.014
10	0.143	0.222	0.1	0.199	0.177	0.225	0.261	0.271

Average execution time in seconds
Rows: number of items available for choice
Columns: knapsack limit
Subcolumns a and b: order by which items were given in input

A real world example: valve placement

- A water supply system, made up by a source and some client nodes connected by pipes
- Place valves (available in a limited amount) at pipes'extremes so that when a pipe breaks it is possible to isolate it with minimum collateral damage for the rest of the network
- Pipe (3,5) breaks. Configuration on the left: valve b is closed and nodes 4, 5, 6 are disconnected. Configuration on the right: it's necessary to close valve a and disconnect the entire network

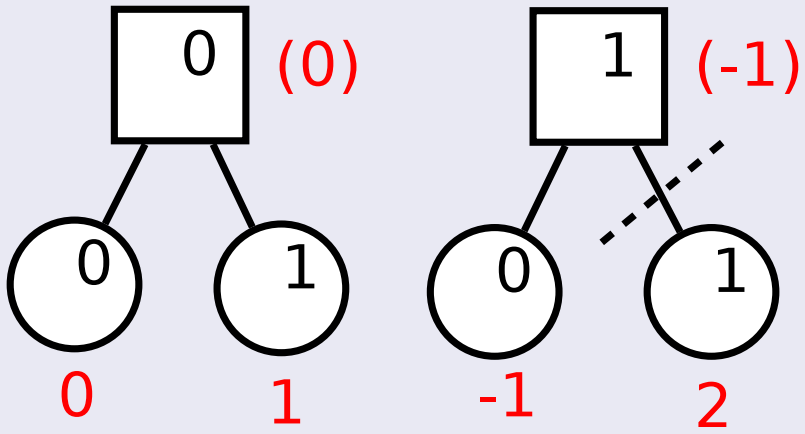


A real world example: valve placement

- Some authors [O. Giustolisi, D. Savic et al] addressed it with genetic algorithms
- Seems reasonable to try alternative strategies
  - Techniques for ‘games’
  - Quantified Constraint Satisfaction Problems

The valve placement game

- We have to minimize a maximum (the worst possible damage)
- Minmax algorithm for adversarial search (games)
  - Player ‘min’ does the first move choosing a placement for valves
  - Player ‘max’ chooses which pipe to break (trying to maximize colleteral damage)
  - Player ‘min’ closes the necessary valves to isolate the pipe (and if they had been placed in the best way, minimizes the damage)
- Alpha-beta cuts on minmax (squares are player ‘max’)

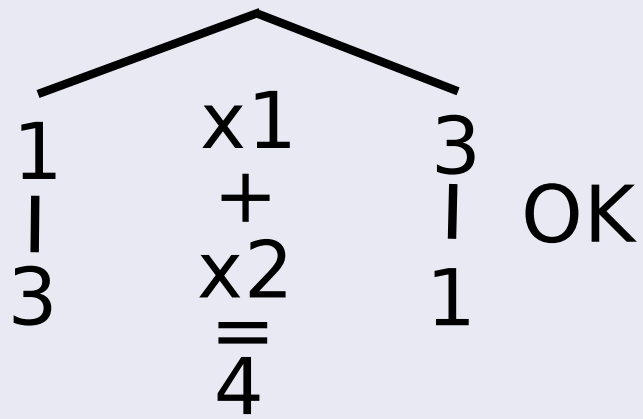


The valve placement QCSP

- $\exists PV_1, ...PV_{nv}\ \forall G \in P_1...P_{np}\ \exists OV_1, ...OV_{nv}\ G \in UD$
- There exists a valve placement such that *for each* possible broken pipe, there exists a closed valves configuration such that the broken pipe belongs to the disconnected network
- In normal CSPs only existential quantifiers are available, in QCSPs also universal ones [see works by M. Benedetti, L. Bordeaux, I. Gent, K. Stergiou et al]

QCSPs

- Useful to achieve robustness and reason under uncertainty e.g. in
  - Scheduling
  - Configuration problems
  - *Games*
- Unlike for CSPs, solution is not an assignment but a tree/strategy
- E.g. $\forall x_1 \in \{1,3\}\ \exists x_2 \in \{1,3\}\ (x_1 + x_2 = 4)$:

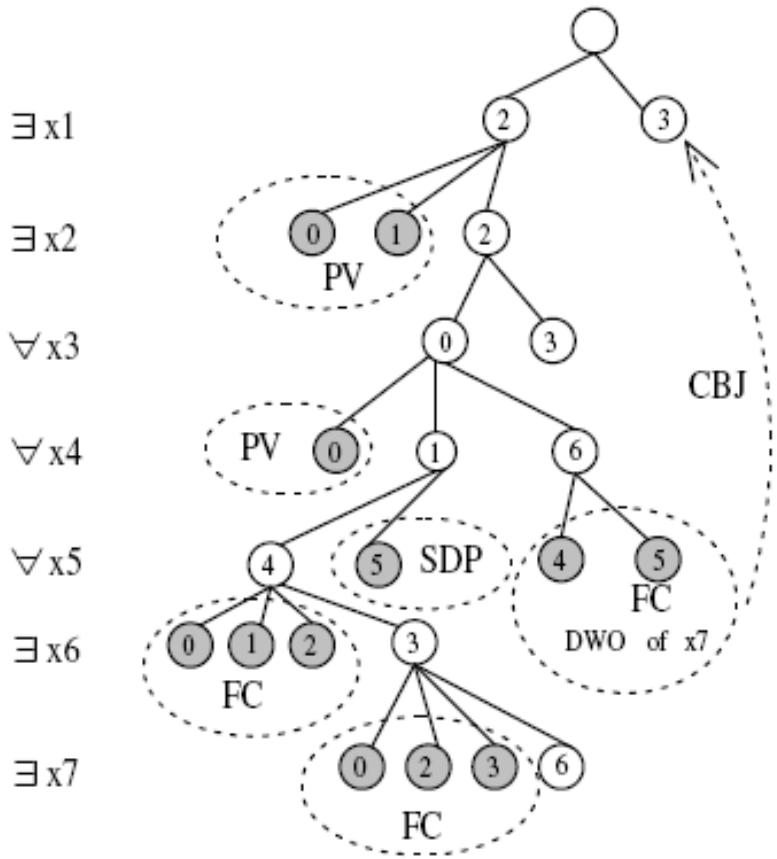


QCSP solving techniques

- Quantified Arc Consistency (taking into account the four possible cases)
- FC1 (extension of forward checking with look ahead behaviour on universals)
- Pure Value rule (pruning of (committing to) pure values if the variable is universal (existential))
- Conflict based backjumping (keep track of the variables responsible for domain wipeout)
- Solution directed pruning (when it would be needed to expand universals, first check if successive existential don't conflict)

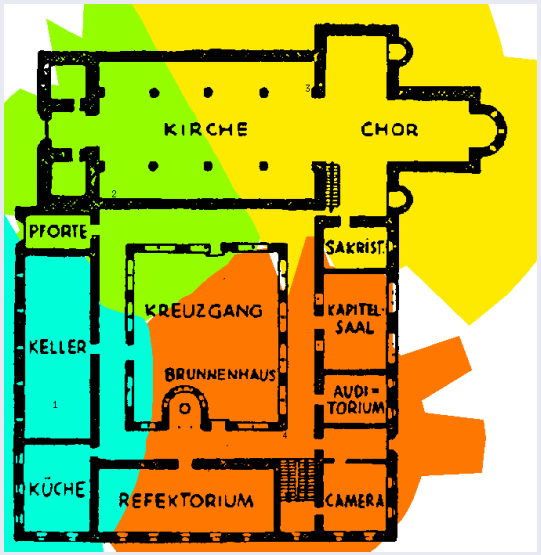
QCSP solving example

$$\exists x_1 \in \{2,3\}\ \exists x_2 \in \{0,1,2\}$$
$$\forall x_3 \in \{0,3\}\ \forall x_4 \in \{0,1,6\}\ \forall x_5 \in \{4,5\}$$
$$\exists x_6 \in \{0,1,2,3\}\ \exists x_7 \in \{0,2,3,6\}$$
$$(x_1 \neq x_6) \wedge (x_1 \neq x_7) \wedge (x_2 \neq x_6) \wedge$$
$$(x_3 \neq x_6) \wedge (x_3 < x_7) \wedge (x_4 \neq x_6) \wedge$$
$$(x_4 \neq x_7) \wedge (x_5 \neq x_6) \wedge (x_5 < x_7)$$



CHR and (real world) optimization

- CHR proved to be an effective tool for related problems
- A notable application: optimal placement of local telecommunication transmitter stations [P. Brisset, J.R. Molwitz and T. Fruehwirth]



- More to come (dynamic programming, games, ...)?