AUTOMATED REASONING

Agostino Dovier

Università di Udine
CLPLAB

Udine, November 2016

Global Constraints

We have seen what is behind the propagation of the global constraint all_different.

In the Global Constraint Catalog
`http://sofdem.github.io/gccat/`
there is the list of (most) studied global constraints.
For each of them there are deep studies (similar to what done for all_different) and clever algorithms.
Sometimes obtaining GAC is NP hard and only approximated filtering algorithms are implemented.

Global Constraints of Minizinc `https://www.minizinc.org/downloads/doc-1.6/mzn-globals.html`

(Some) Global Constraints of MiniZinc

alldifferent	alldifferent_except_0
all_disjoint	all_equal
at_least (atleast)	at_most (atmost)
bin_packing	circuit
cumulative	element
global_cardinality	inverse
lex_greater	lex_less
maximum	minimum
sort	table

Sometimes you have “options” (e.g. :: domain in all_different, using the solver G12fd)

LARGE NEIGHBORHOOD SEARCH

LARGE NEIGHBORHOOD SEARCH

It is a hybrid search technique introduced by P. Shaw (Proc of CP 1998) and largely used by the language Comet (P. Van Hentenryck and L. Michel. Constraint-Based Local Search. The MIT Press, 2005)

Hybrid = Local Search + Constraint Programming

Its main use is for solving minimization problems (COP)

The language Comet (close in spirit and in syntax to Minizinc) is no longer available (due to an international copyright issue). However, there is a library for LNS in Minizinc.
`http://www.minizinc.org/minisearch/`

Local info. It was also used in CLP (eg DDFP 2010 — ICLP, best paper) and with an ad-hoc CP solver on GPU (eg CDP 2015 — JETAI) for dealing with the protein structure prediction problem.

LARGE NEIGHBORHOOD SEARCH

It is a hybrid search technique introduced by P. Shaw (Proc of CP 1998) and largely used by the language Comet (P. Van Hentenryck and L. Michel. Constraint-Based Local Search. The MIT Press, 2005)

Hybrid = Local Search + Constraint Programming

Its main use is for solving minimization problems (COP)

The language Comet (close in spirit and in syntax to Minizinc) is no longer available (due to an international copyright issue). However, there is a library for LNS in Minizinc.
`http://www.minizinc.org/minisearch/`

Local info. It was also used in CLP (eg DDFP 2010 — ICLP, best paper) and with an ad-hoc CP solver on GPU (eg CDP 2015 — JETAI) for dealing with the protein structure prediction problem.

LARGE NEIGHBORHOOD SEARCH

It is a hybrid search technique introduced by P. Shaw (Proc of CP 1998) and largely used by the language Comet (P. Van Hentenryck and L. Michel. Constraint-Based Local Search. The MIT Press, 2005)

Hybrid = Local Search + Constraint Programming

Its main use is for solving minimization problems (COP)

The language Comet (close in spirit and in syntax to Minizinc) is no longer available (due to an international copyright issue). However, there is a library for LNS in Minizinc.
`http://www.minizinc.org/minisearch/`

Local info. It was also used in CLP (eg DDFP 2010 — ICLP, best paper) and with an ad-hoc CP solver on GPU (eg CDP 2015 — JETAI) for dealing with the protein structure prediction problem.

LARGE NEIGHBORHOOD SEARCH

It is a hybrid search technique introduced by P. Shaw (Proc of CP 1998) and largely used by the language Comet (P. Van Hentenryck and L. Michel. Constraint-Based Local Search. The MIT Press, 2005)

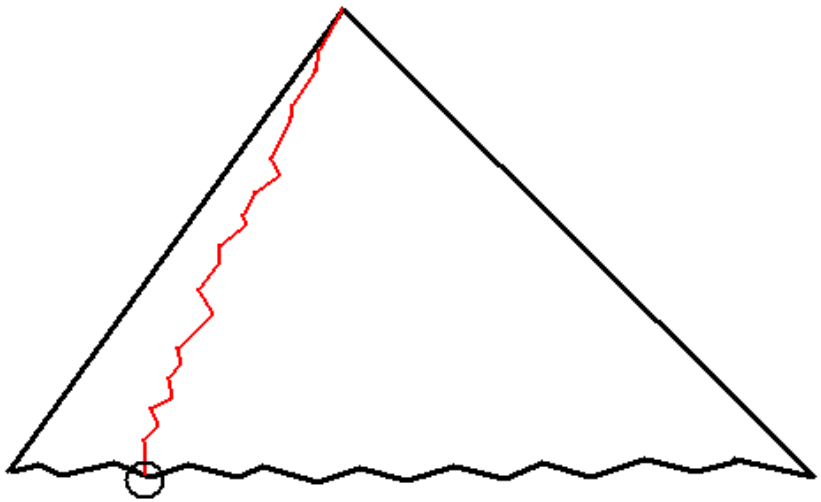
Hybrid = Local Search + Constraint Programming

Its main use is for solving minimization problems (COP)

The language Comet (close in spirit and in syntax to Minizinc) is no longer available (due to an international copyright issue). However, there is a library for LNS in Minizinc.
`http://www.minizinc.org/minisearch/`

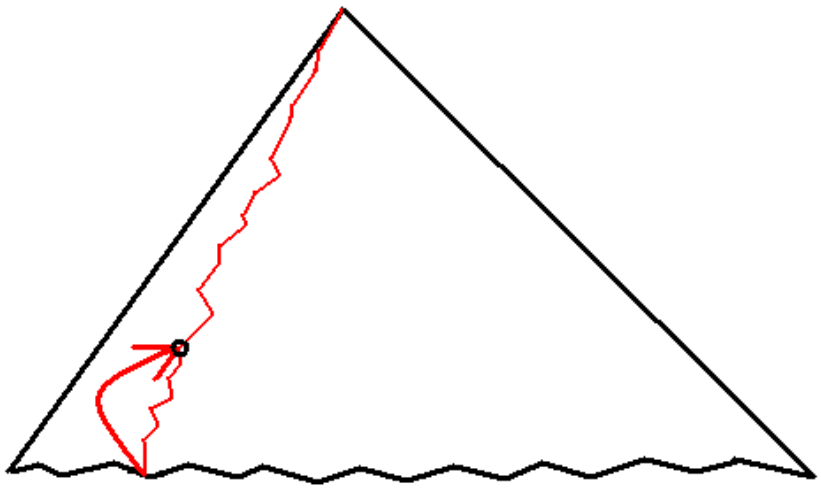
Local info. It was also used in CLP (eg DDFP 2010 — ICLP, best paper) and with an ad-hoc CP solver on GPU (eg CDP 2015 — JETAI) for dealing with the protein structure prediction problem.

CP PROP-LABELING TREE (SOLVING COPS)



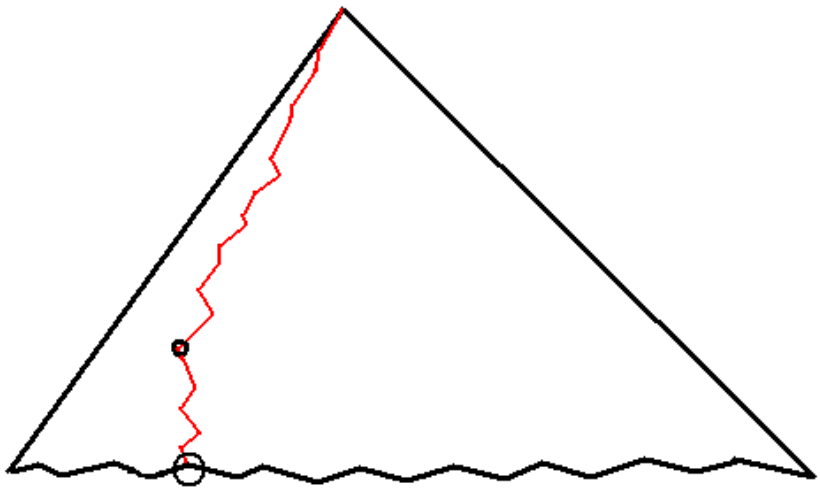
The complete visit of a huge tree might require too much time

CP PROP-LABELING TREE (SOLVING COPS)



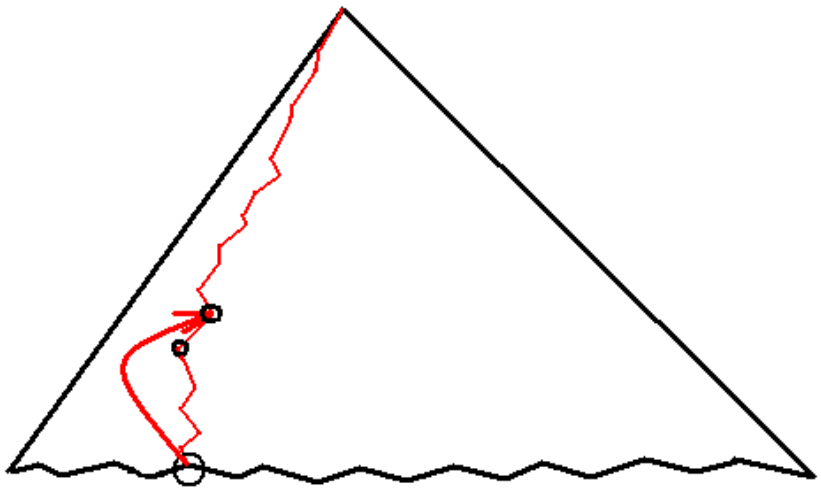
The complete visit of a huge tree might require too much time

CP PROP-LABELING TREE (SOLVING COPS)



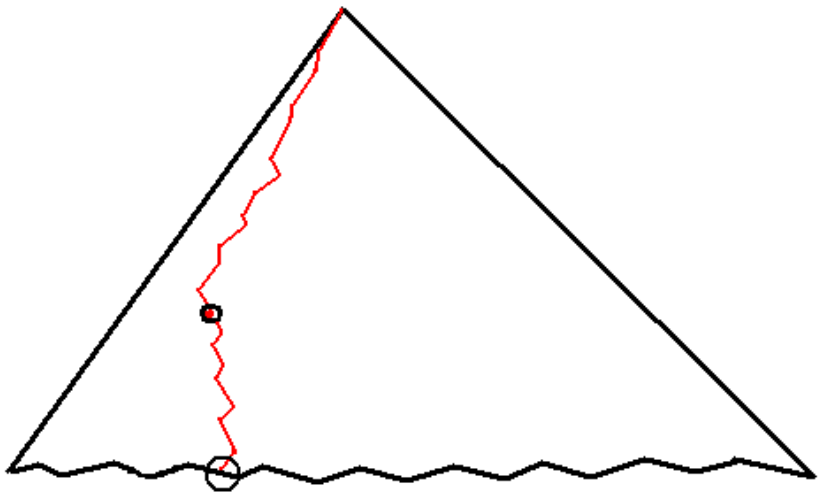
The complete visit of a huge tree might require too much time

CP PROP-LABELING TREE (SOLVING COPS)



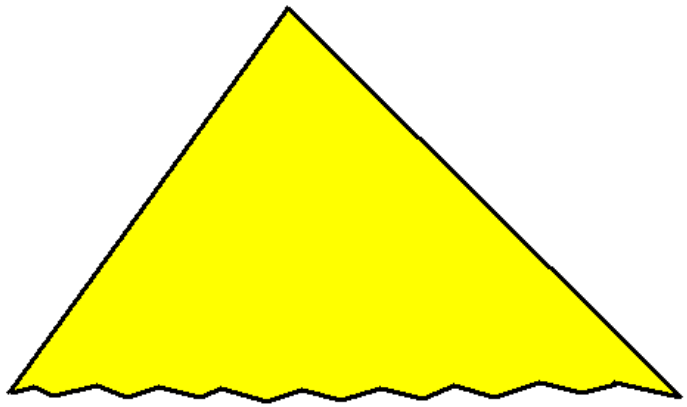
The complete visit of a huge tree might require too much time

CP PROP-LABELING TREE (SOLVING COPS)



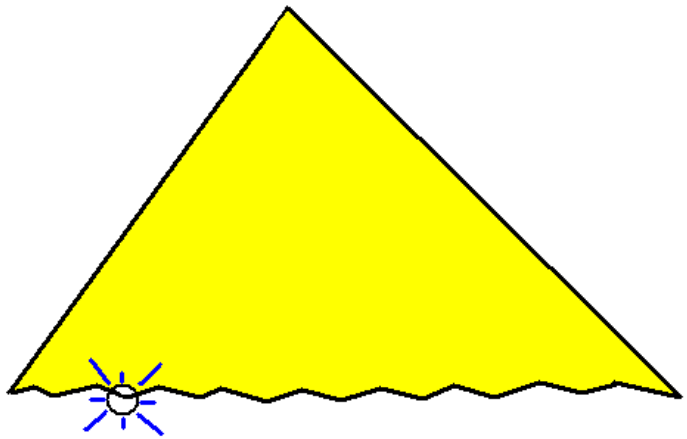
The complete visit of a huge tree might require too much time

LOCAL SEARCH (SOLVING COPS)



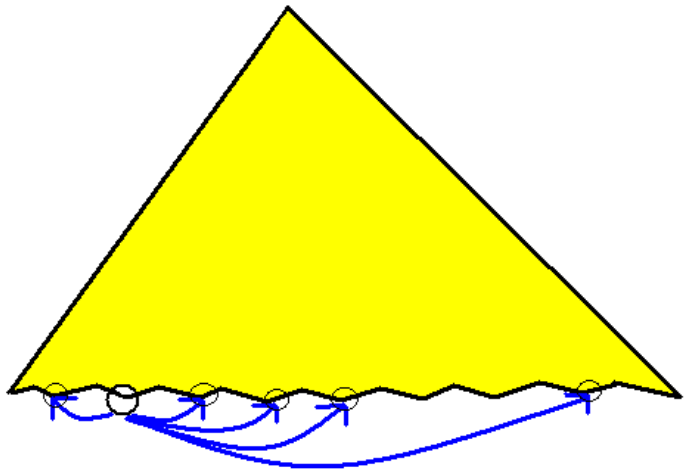
Given a possible solution, the algorithm perform moves on the frontier of the search tree. There are several techniques (meta heuristics) for deciding how to move. All of them are based on random choices that try to improve the solution.

LOCAL SEARCH (SOLVING COPS)



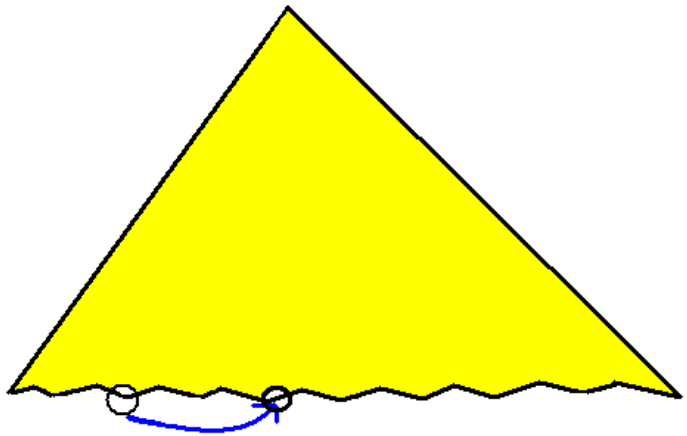
Given a possible solution, the algorithm perform moves on the frontier of the search tree. There are several techniques (meta heuristics) for deciding how to move. All of them are based on random choices that try to improve the solution.

LOCAL SEARCH (SOLVING COPS)



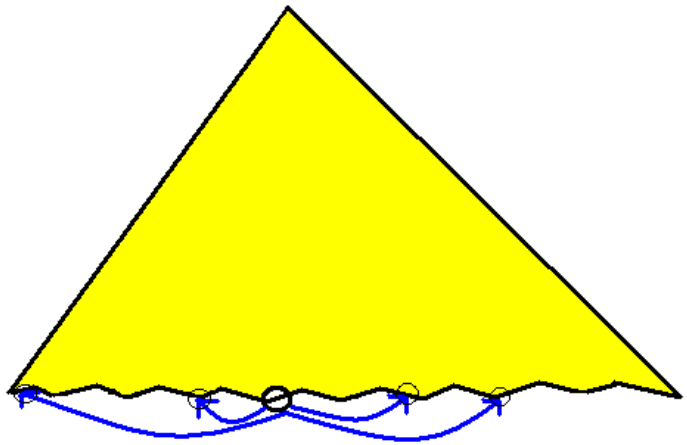
Given a possible solution, the algorithm perform moves on the frontier of the search tree. There are several techniques (meta heuristics) for deciding how to move. All of them are based on random choices that try to improve the solution.

LOCAL SEARCH (SOLVING COPS)



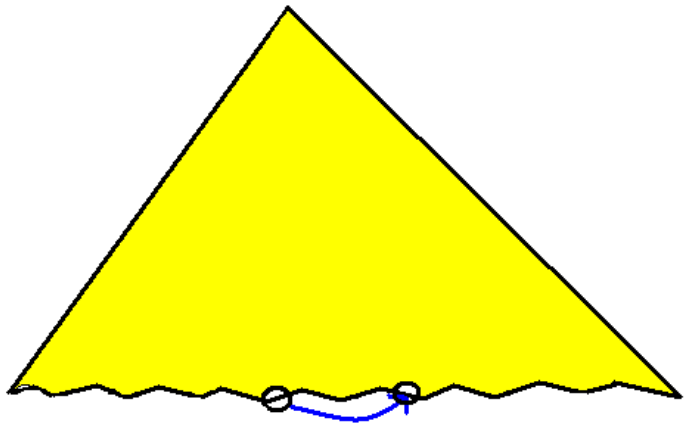
Given a possible solution, the algorithm perform moves on the frontier of the search tree. There are several techniques (meta heuristics) for deciding how to move. All of them are based on random choices that try to improve the solution.

LOCAL SEARCH (SOLVING COPS)



Given a possible solution, the algorithm perform moves on the frontier of the search tree. There are several techniques (meta heuristics) for deciding how to move. All of them are based on random choices that try to improve the solution.

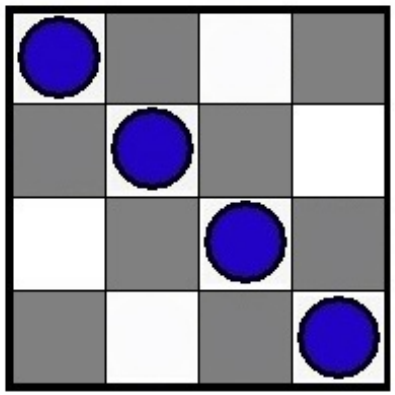
LOCAL SEARCH (SOLVING COPS)



Given a possible solution, the algorithm perform moves on the frontier of the search tree. There are several techniques (meta heuristics) for deciding how to move. All of them are based on random choices that try to improve the solution.

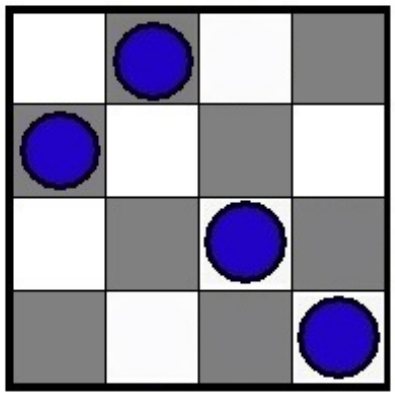
4 QUEENS, ONLY HOR+VER CONSTRAINTS

COST = NUMBER OF ATTACKS

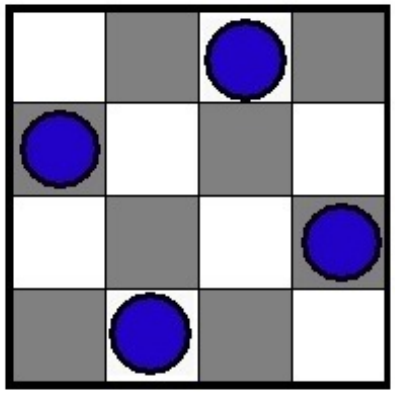


12 attacks

$\Rightarrow$ swap(1,2)

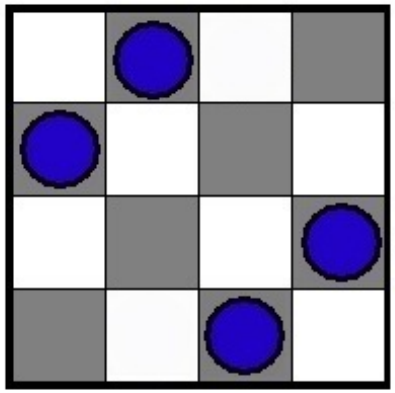


4 attacks $\Downarrow$ swap(3,4)



0 attacks

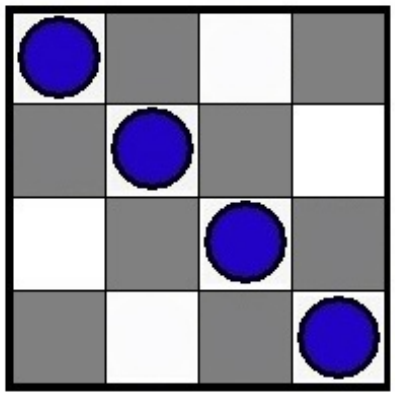
$\Leftarrow$ swap(2,3)



8 attacks

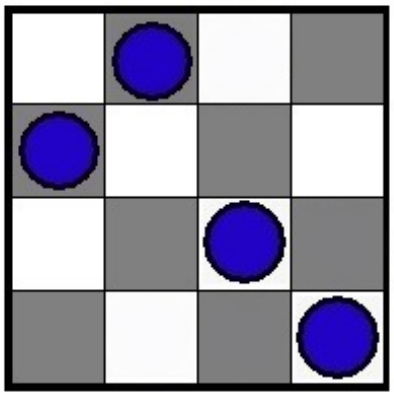
4 QUEENS, ONLY HOR+VER CONSTRAINTS

COST = NUMBER OF ATTACKS

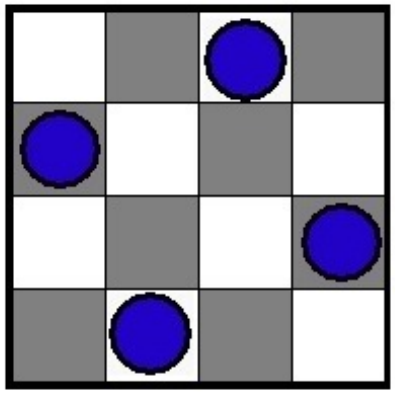


12 attacks

$\Rightarrow$ swap(1,2)

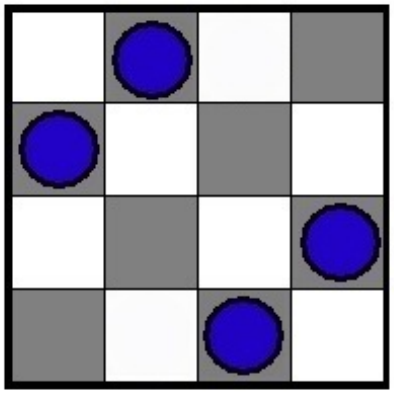


4 attacks $\Downarrow$ swap(3,4)



0 attacks

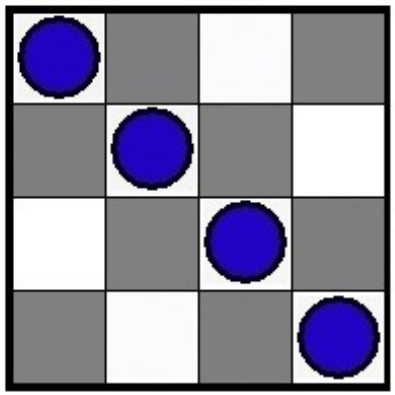
$\Leftarrow$ swap(2,3)



8 attacks

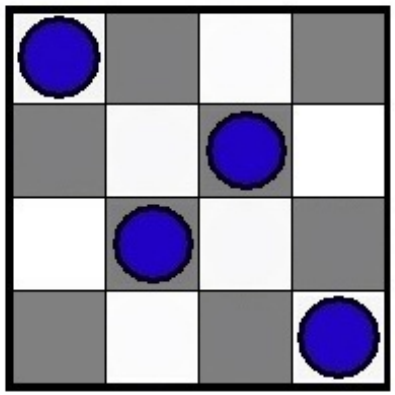
4 QUEENS, ONLY HOR+VER CONSTRAINTS

COST = NUMBER OF ATTACKS



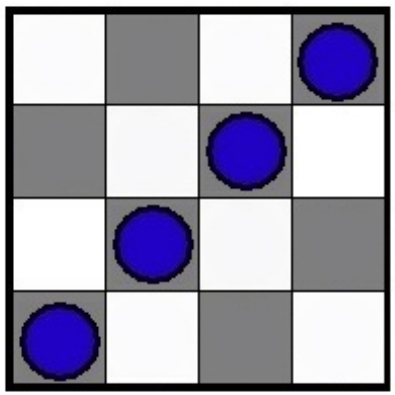
12 attacks

$\Rightarrow$ swap(2,3)



4 attacks $\Downarrow$ swap(1,4)

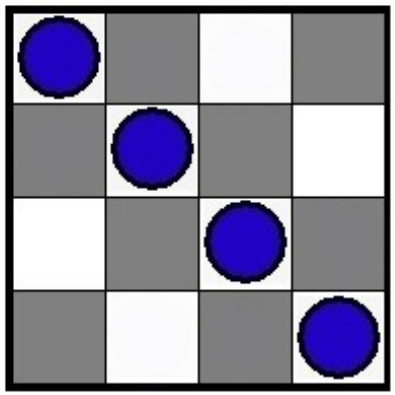
Not all
the choices
lead to
the minimum



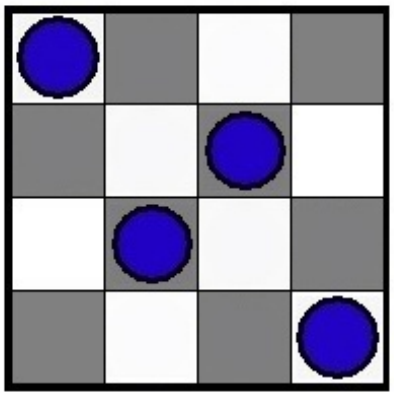
12 attacks

4 QUEENS, ONLY HOR+VER CONSTRAINTS

COST = NUMBER OF ATTACKS



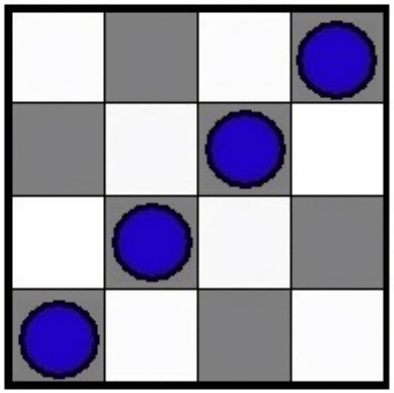
$\Rightarrow$ swap(2,3)



12 attacks

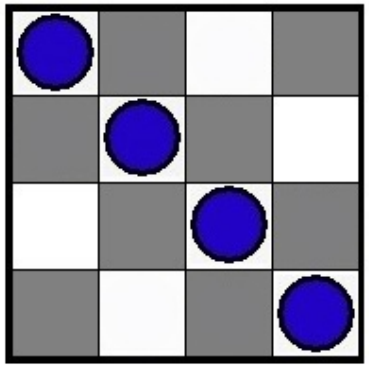
4 attacks $\Downarrow$ swap(1,4)

Not all
the choices
lead to
the minimum

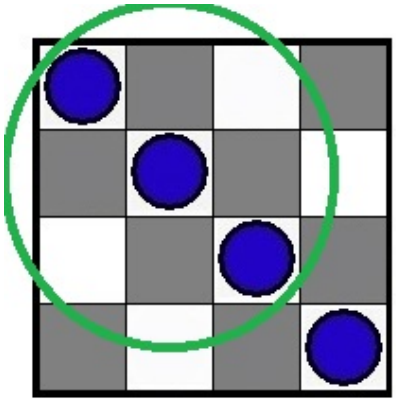


12 attacks

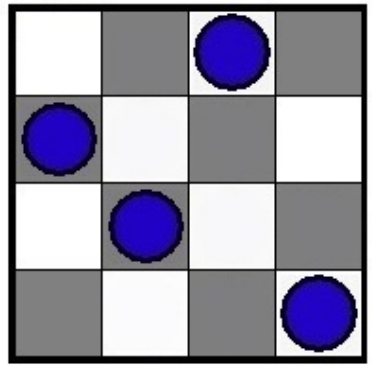
IDEA: ENLARGE THE NEIGHBORHOOD AND USE CP



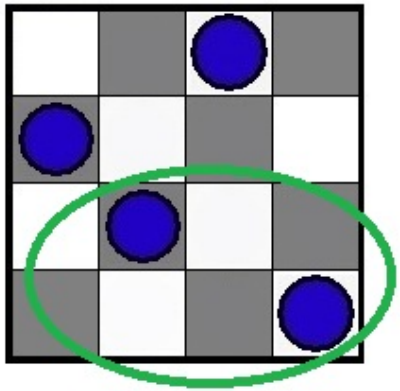
12 attacks



$\Rightarrow$
minimize
"locally"

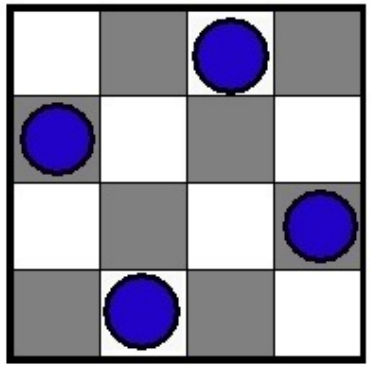


2 attacks



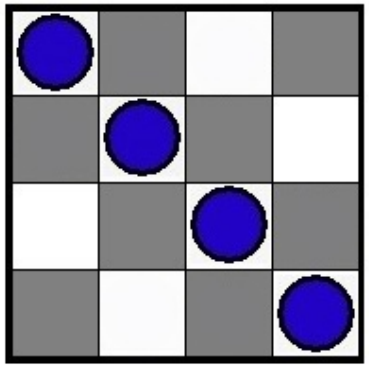
2 attacks

$\Rightarrow$
minimize
"locally"

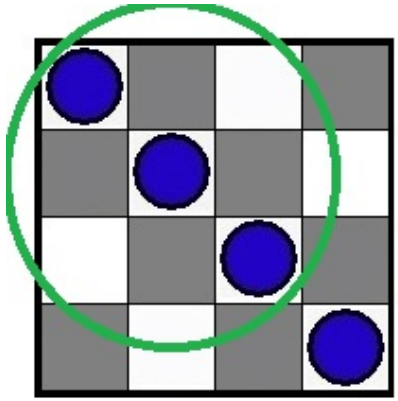


0 attacks

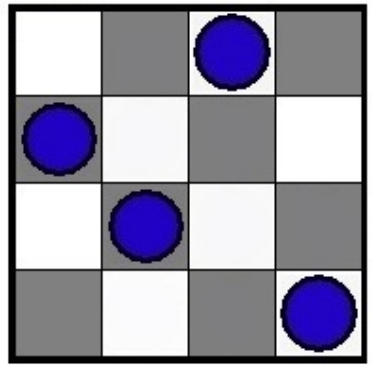
IDEA: ENLARGE THE NEIGHBORHOOD AND USE CP



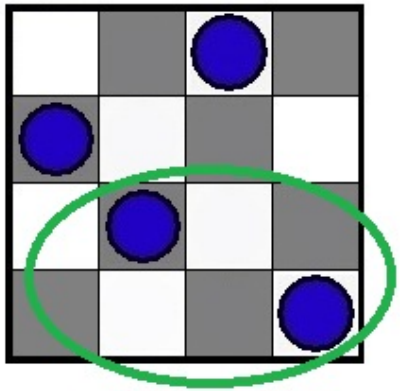
12 attacks



$\Rightarrow$
minimize
"locally"

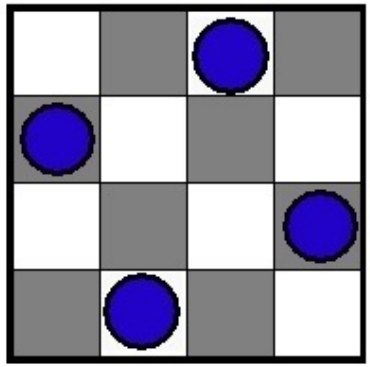


2 attacks



2 attacks

$\Rightarrow$
minimize
"locally"



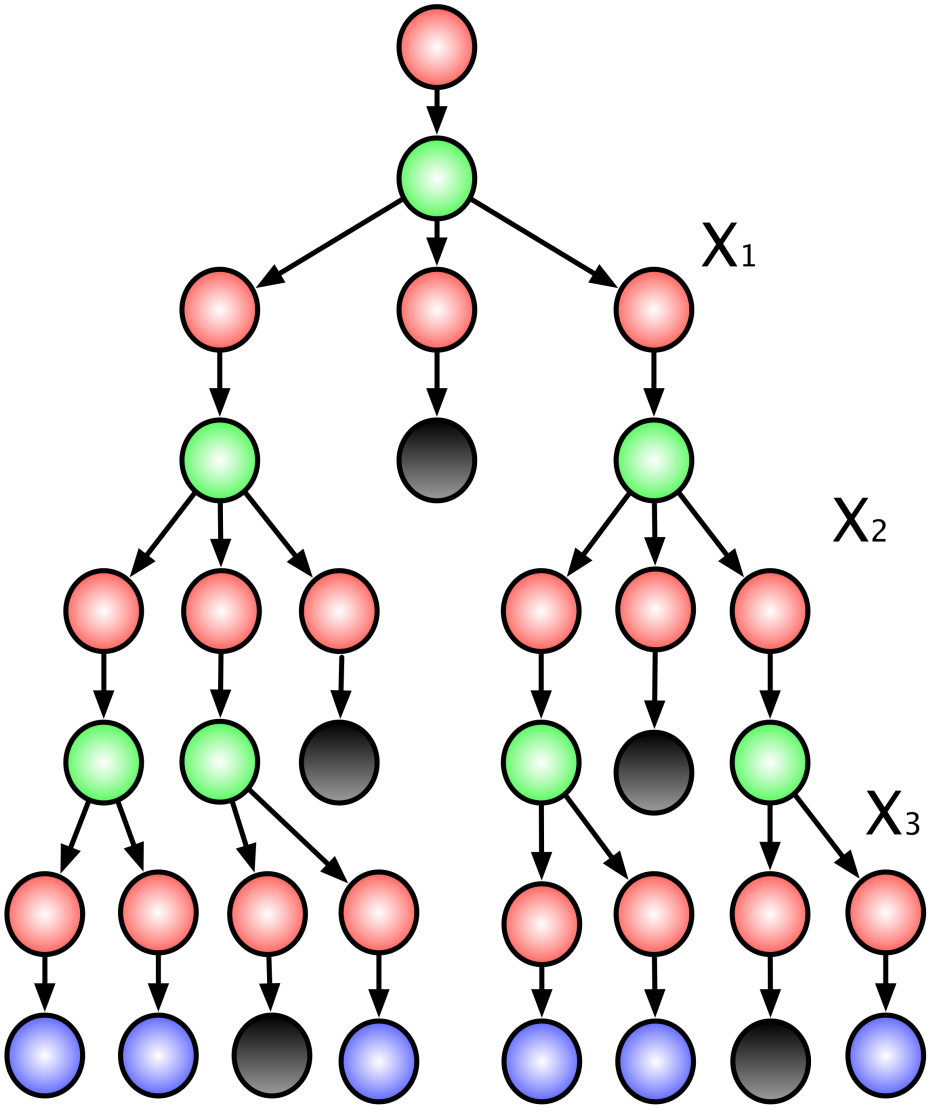
0 attacks

Local Search

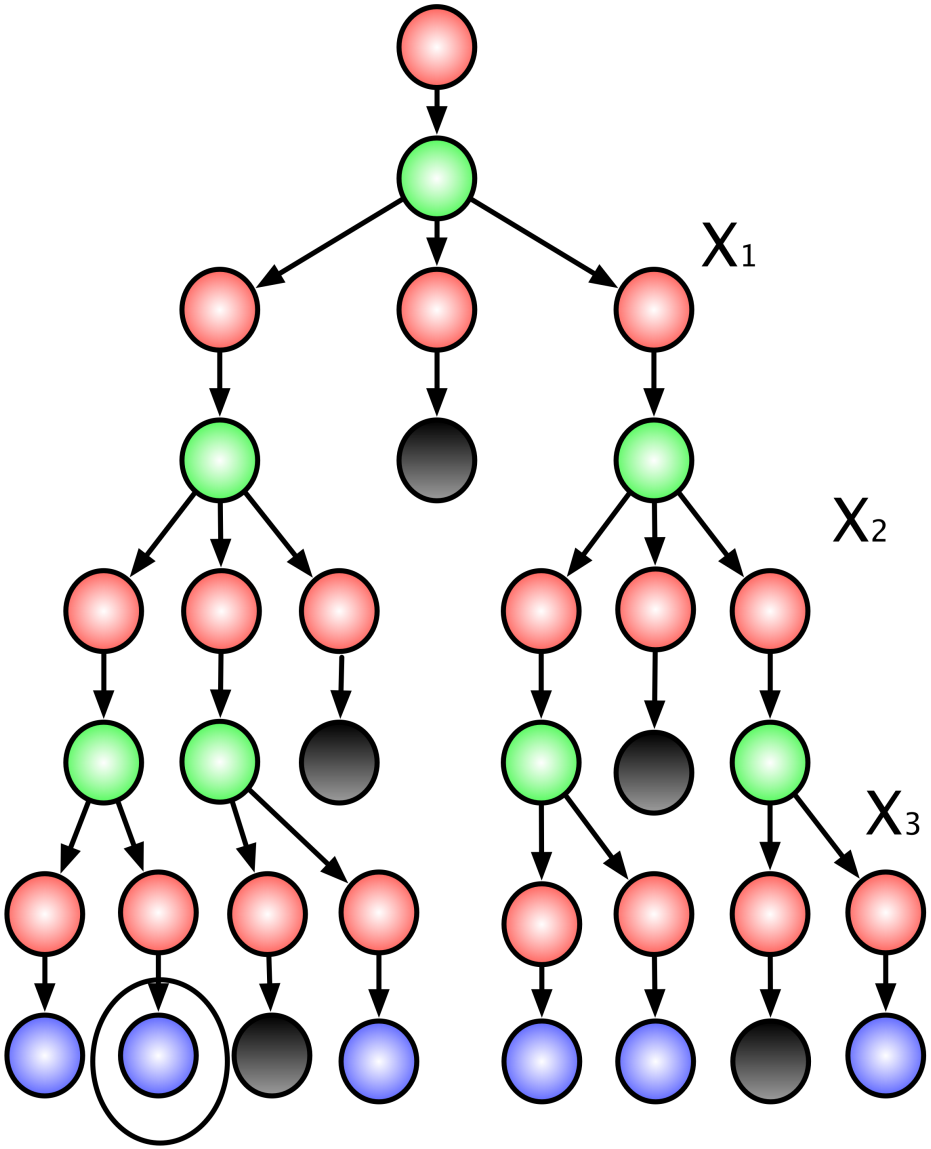
In practice

- An algorithm for finding the initial solution must be defined
- Moreover, a function N computing the Neighborhood of a solution σ that identifies a set $N(\sigma)$ of feasible solutions "easily" reachable from σ
- If possible the Neighborhood is characterized using a notion of *move* (e.g., change of the values of two variables)
- Moreover, one has to state:
  - How choosing a move
  - How exploring the Neighborhood
  - How avoding undesirable solutions/situations (e.g., loops)
  - How/When halting the computation (typically, after a fixed number of non-improving steps)
- One possible metaheuristics is Hill Climbing: choose $\sigma' \in N(\sigma)$ minimizing locally f, namely $f(\sigma') = \min\{f(\mu) : \mu \in N(\sigma)\}$

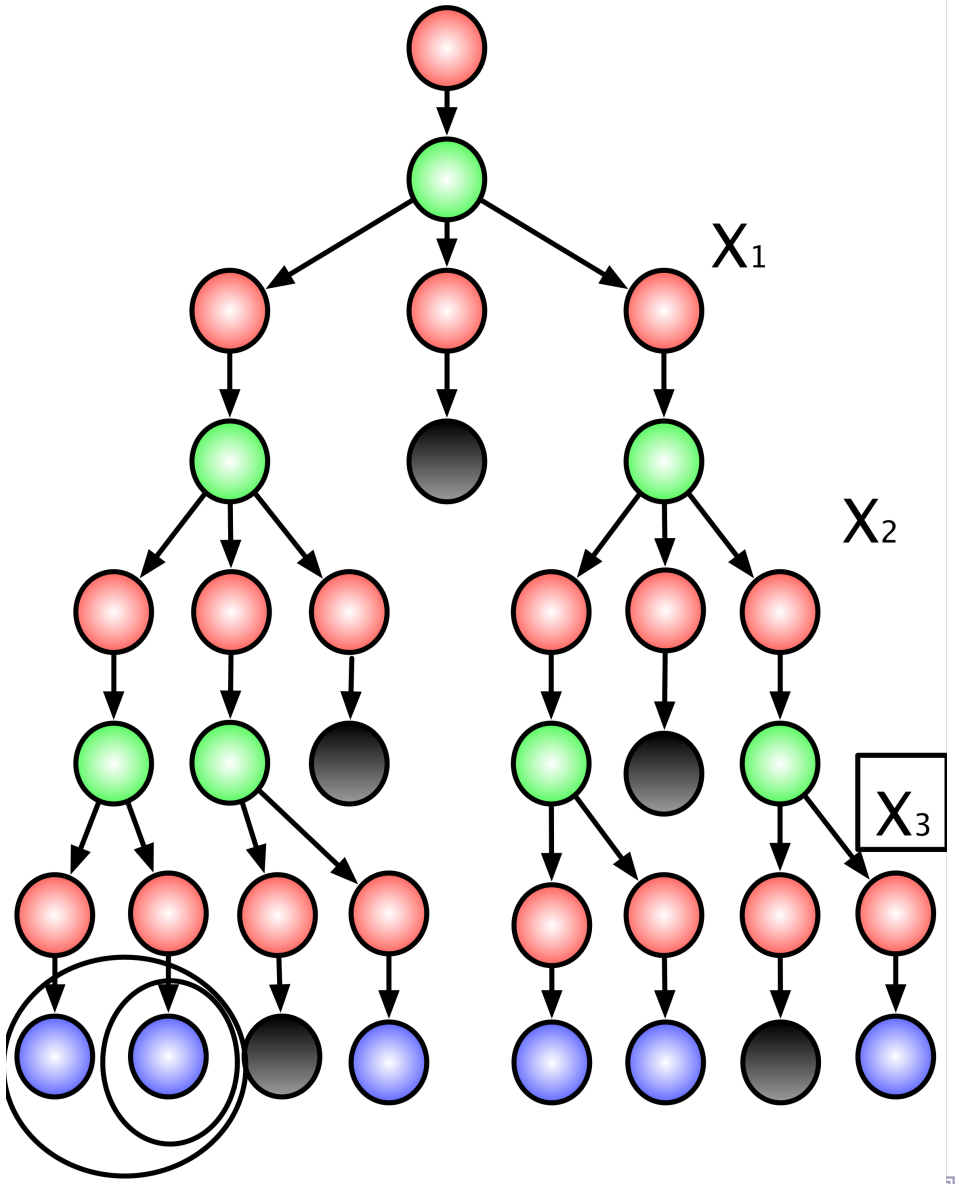
Small and Large Neighboorhood with CP



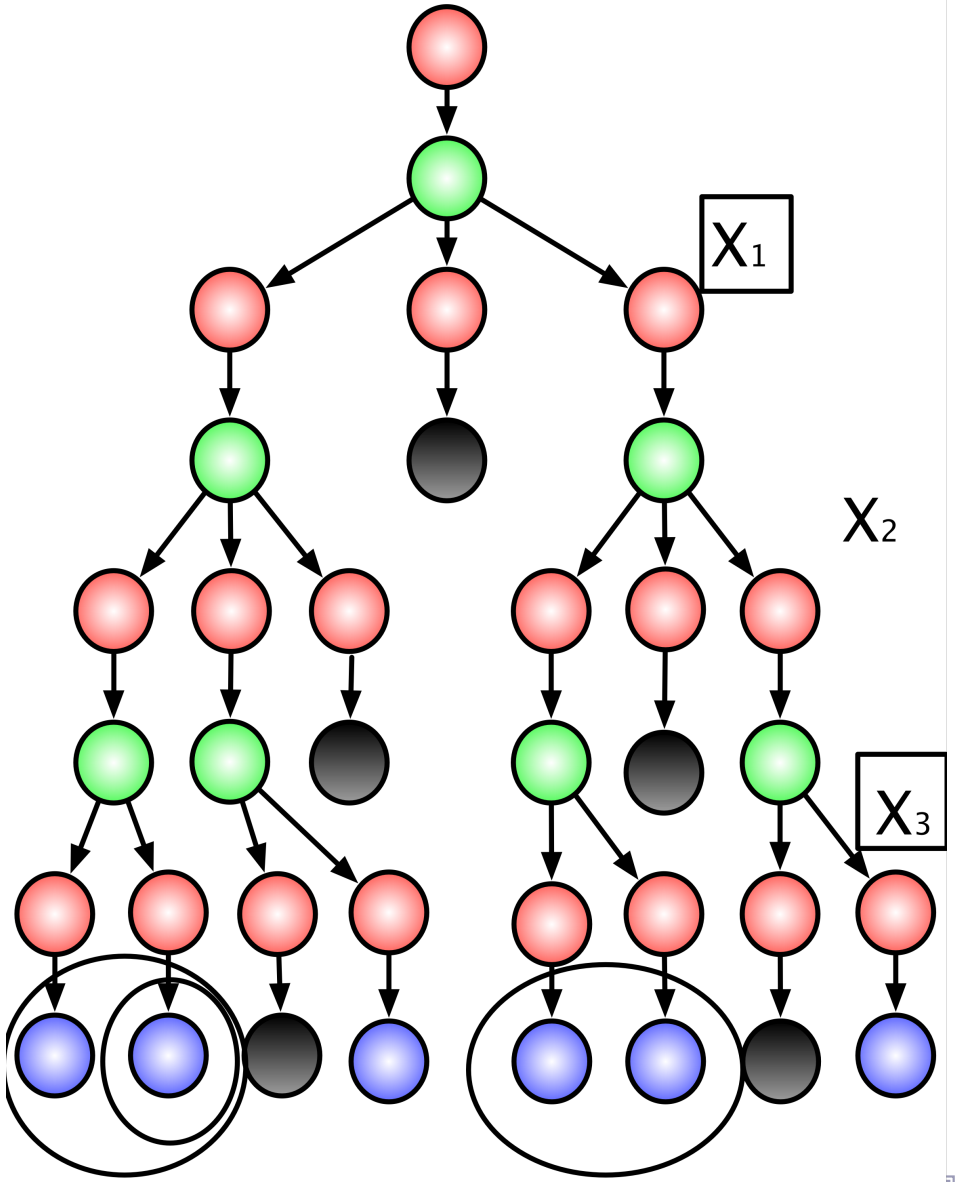
Small and Large Neighboorhood with CP



Small and Large Neighboorhood with CP



Small and Large Neighboorhood with CP



Large Neighborhood Search

- Given a solution $\vec{s}$ for the COP $\langle \vec{X}, \vec{D}, C, f \rangle$ we can "unassign" (randomly) some (a fixed or random percentage) of the variables, say $\mathcal{N} \subseteq \vec{X}$
- The set of values for $\mathcal{N}$ that are a solution of the COP constitutes a neighborhood of $\vec{s}$ (including $\vec{s}$)
- Given the COP, $\mathcal{N}$ identifies uniquely a neighborhood (that should be explored)
- We can run the CP solver on the neighborhood and finding the minimum.
- Or setting a timeout and finding the best solution in the timeout
- If it does not improve f, try a new neighborhood.
- Repeat until you encounter a sequence of k successive not improving stages OR until a global timeout is reached

Large Neighborhood Search

- Given a solution $\vec{s}$ for the COP $\langle \vec{X}, \vec{D}, C, f \rangle$ we can "unassign" (randomly) some (a fixed or random percentage) of the variables, say $\mathcal{N} \subseteq \vec{X}$
- The set of values for $\mathcal{N}$ that are a solution of the COP constitutes a neighborhood of $\vec{s}$ (including $\vec{s}$)
- Given the COP, $\mathcal{N}$ identifies uniquely a neighborhood (that should be explored)
- We can run the CP solver on the neighborhood and finding the minimum.
- Or setting a timeout and finding the best solution in the timeout
- If it does not improve f, try a new neighborhood.
- Repeat until you encounter a sequence of k successive not improving stages OR until a global timeout is reached

Large Neighborhood Search

- Given a solution $\vec{s}$ for the COP $\langle \vec{X}, \vec{D}, C, f \rangle$ we can "unassign" (randomly) some (a fixed or random percentage) of the variables, say $\mathcal{N} \subseteq \vec{X}$
- The set of values for $\mathcal{N}$ that are a solution of the COP constitutes a neighborhood of $\vec{s}$ (including $\vec{s}$)
- Given the COP, $\mathcal{N}$ identifies uniquely a neighborhood (that should be explored)
- We can run the CP solver on the neighborhood and finding the minimum.
- Or setting a timeout and finding the best solution in the timeout
- If it does not improve f, try a new neighborhood.
- Repeat until you encounter a sequence of k successive not improving stages OR until a global timeout is reached

THE MINIZINC LIBRARY

As told at the beginning, there is a library for LNS in Mininzinc
`http://www.minizinc.org/minisearch/`

```
x ... the list of decision variables
obj ... the variable storing the objective function

int: iterations = 100;
float: destructionRate = 0.3;
int: time_ms = 5*1000;
solve search lns_min(obj,x,iterations,destructionRate,time_ms);
```

I have a problem with it in my laptop. I asked the developer (Andrea Rendl). You can try in your systems. If it works, please contact me.

We use now CLPFD, but I'm sure we'll be able to use Rendl's library before the end of the course.