Tutorial: Constraint-Based Local Search

Pierre Flener

ASTRA Research Group on CP
Department of Information Technology
Uppsala University
Sweden

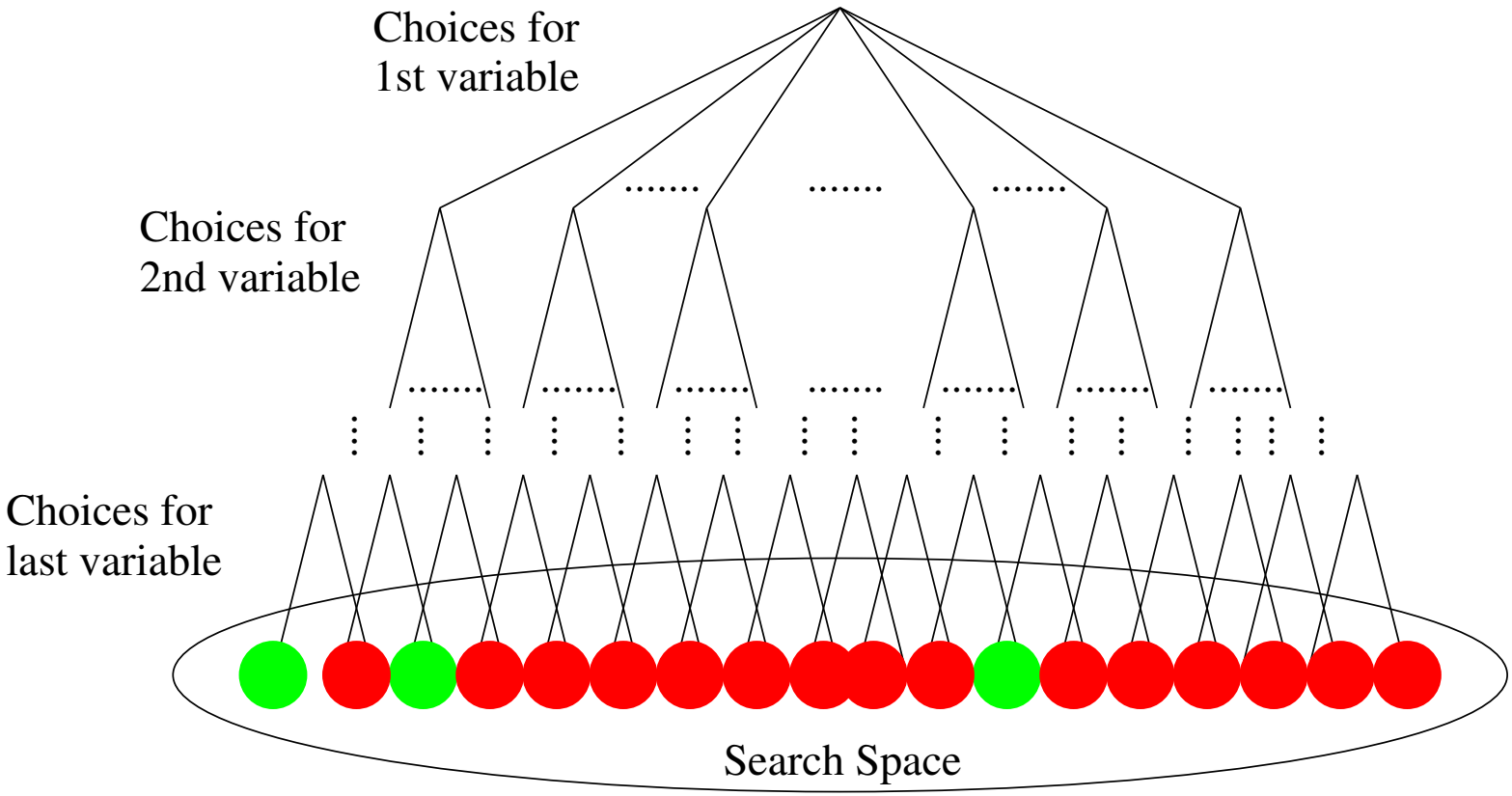
CP meets CAV
25 June 2012

UPPSALA UNIVERSITET

Outline

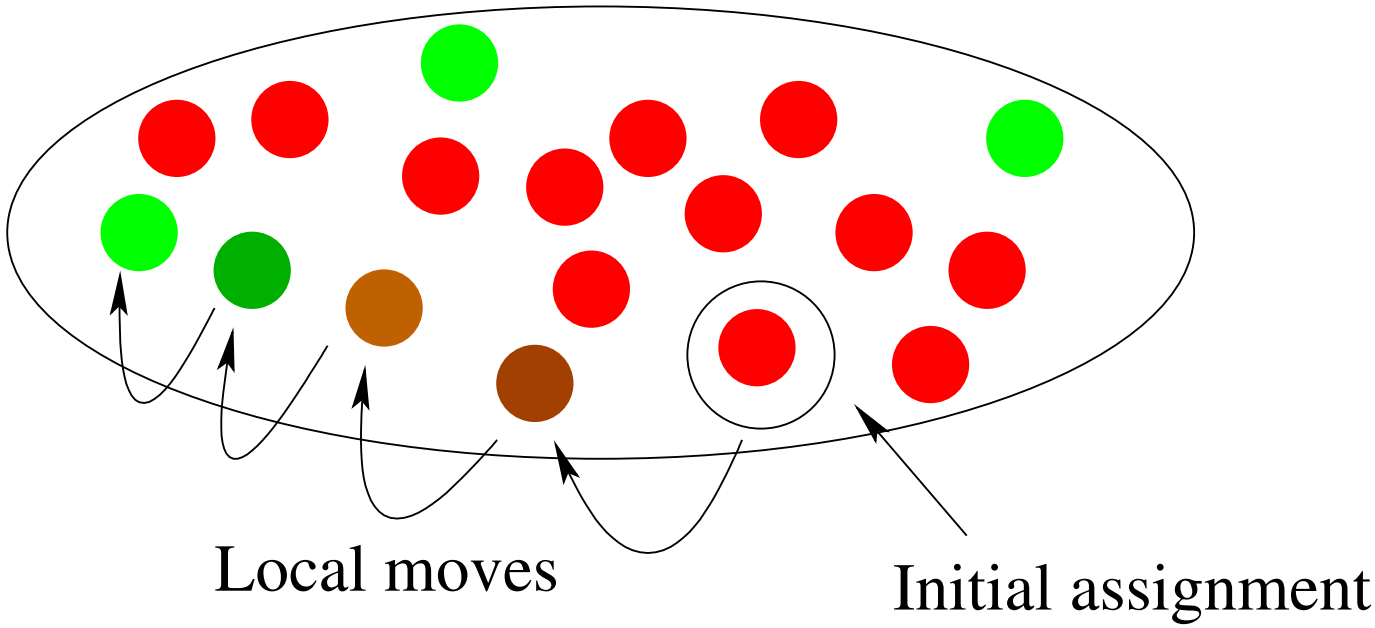
So Far: Inference + Systematic Search

- Values are found 1-by-1 for the decision variables.
- Stop when solution or unsatisfiability proof is obtained.
- Search space from a systematic search viewpoint:



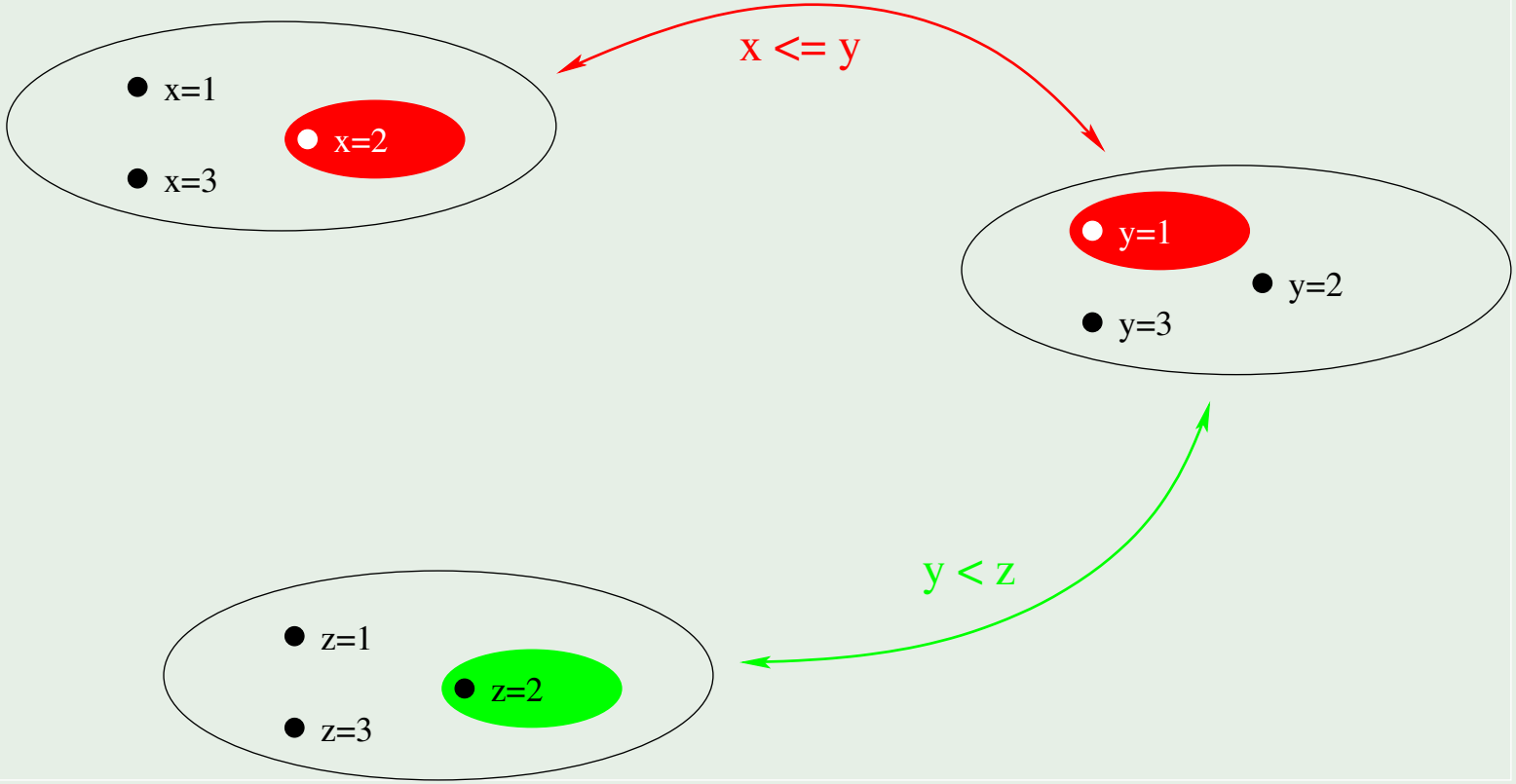
Now: Inference + Local Search

- Values are given to all the variables at the same time.
- Search proceeds by moves, which make small updates to complete assignments, upon probing the impacts of many candidate moves, called the neighbourhood.
- Stop when a good enough assignment has been found or when an allocated resource (running time, or a number of iterations) has been exhausted.



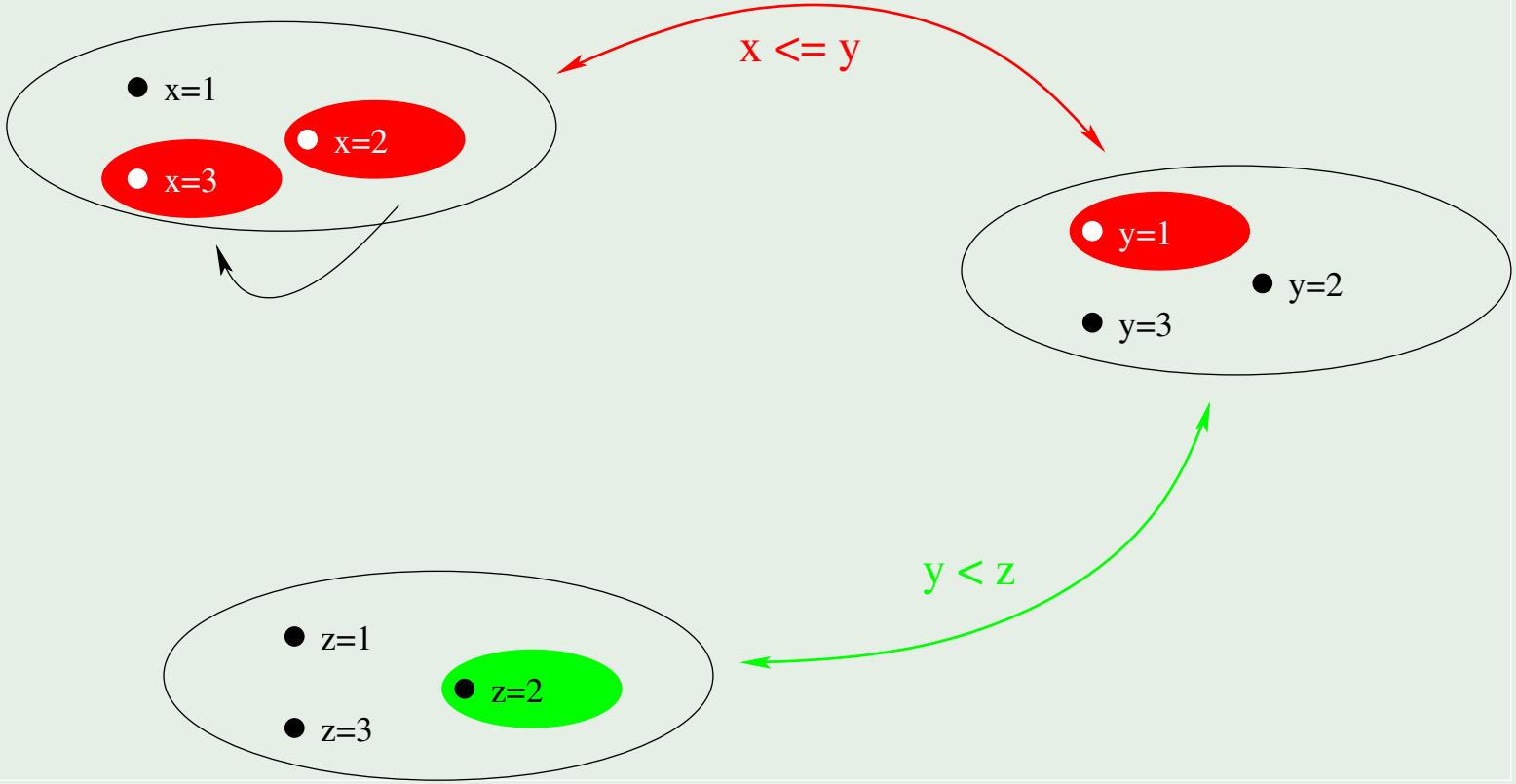
Example ($x, y, z \in \{1, 2, 3\} \wedge x \leq y \wedge y < z$)

Unsatisfying assignment (the constraint $x \leq y$ is violated; the decision variables x and y are violating wrt $x \leq y$):



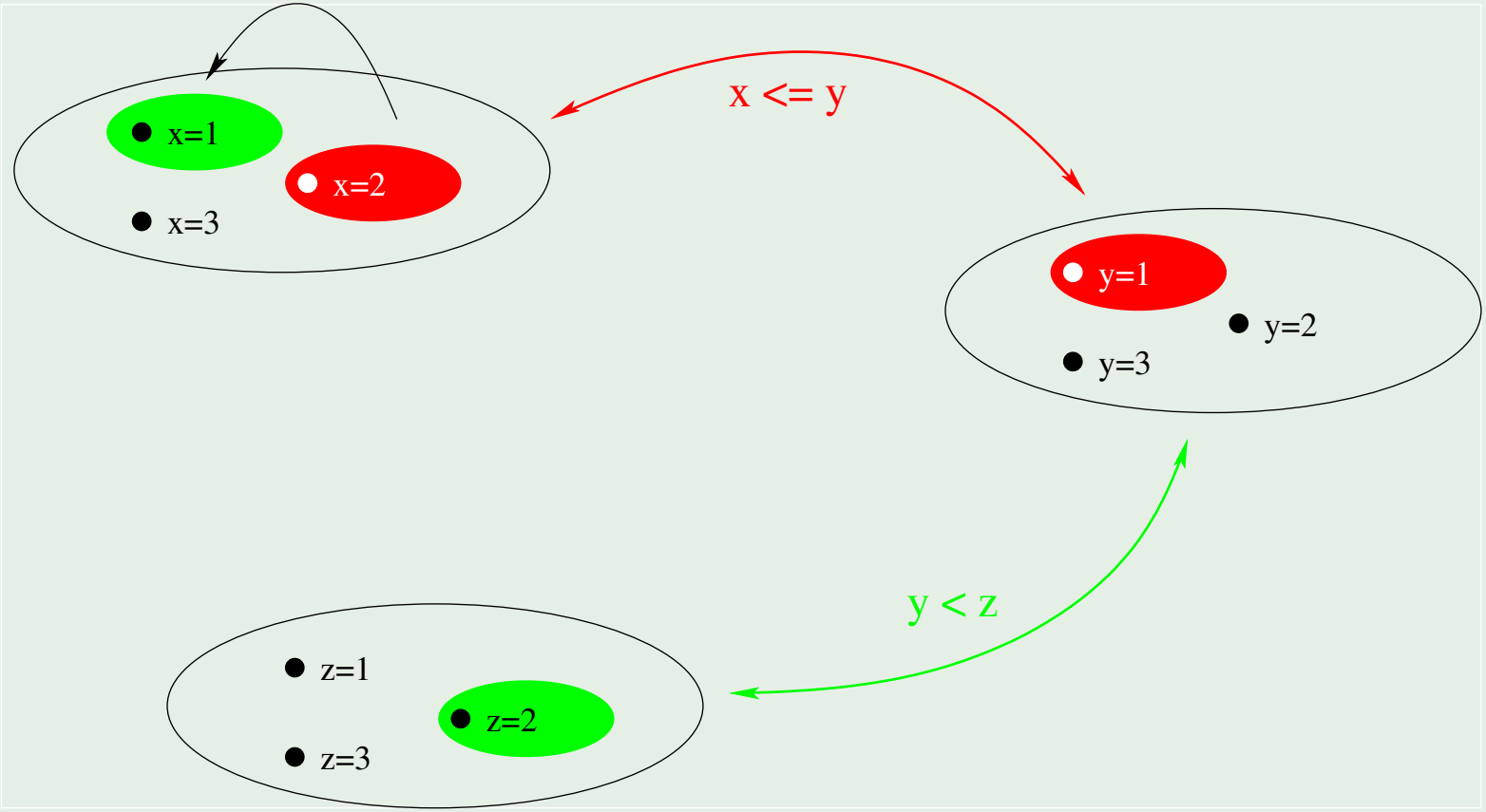
Example ($x, y, z \in \{1, 2, 3\} \wedge x \leq y \wedge y < z$)

Candidate local move $x := 3$, reaching another unsatisfying assignment (the constraint $x \leq y$ is still violated; the decision variables x and y are still violating wrt $x \leq y$):



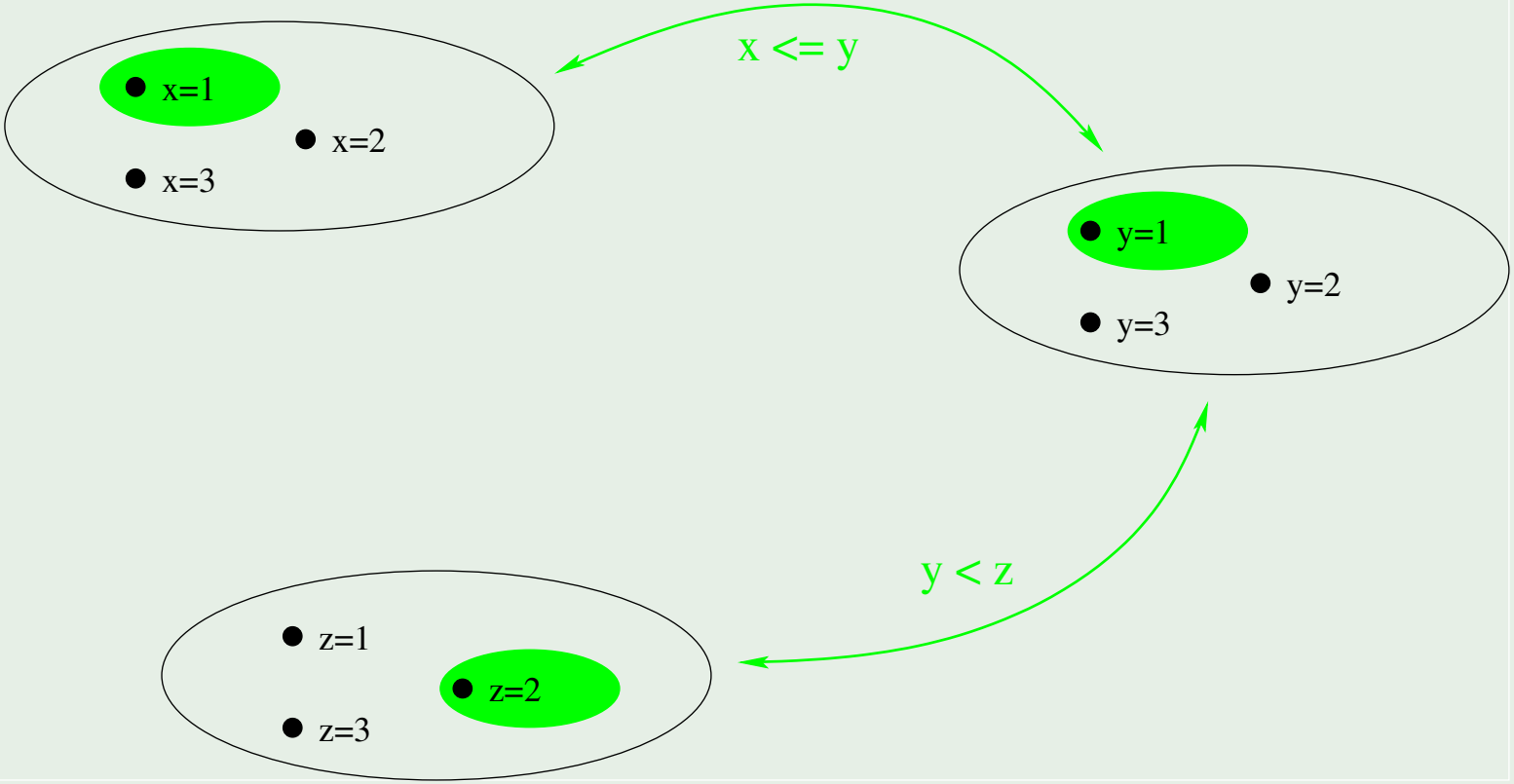
Example ($x, y, z \in \{1, 2, 3\} \wedge x \leq y \wedge y < z$)

Another candidate local move $x := 1$, reaching a satisfying assignment (there are no more violated constraints or violating variables):



Example ($x, y, z \in \{1, 2, 3\} \wedge x \leq y \wedge y < z$)

Another candidate local move $x := 1$, reaching a satisfying assignment (there are no more violated constraints or violating variables):



UPPSALA UNIVERSITET

Systematic search:

+ Will find an (optimal) solution, if one exists.
+ Will give a proof of unsatisfiability, otherwise.
− May take a long time to complete.
− Sometimes does not scale well to large instances.
− May need a lot of tweaking: branching heuristics, ...

Local search:

+ May find an (optimal) solution, if one exists.
− Can never give a proof of unsatisfiability, otherwise.
− Can never guarantee that the found solution is optimal.
+ Often scales well to large instances.
− May need a lot of tweaking: heuristics, parameters, ...

Local search trades completeness and quality for speed!

UPPSALA UNIVERSITET

Local Search: Sample Heuristics

Example

Systematic (partial) exploration of the neighbourhood:

- First improving neighbour: Make the first move that improves on the current assignment.
- Steepest/Gradient descent: Make a random best move.
- Min-conflict: Make a random best move that modifies a violating variable.
- ...

Random walk:

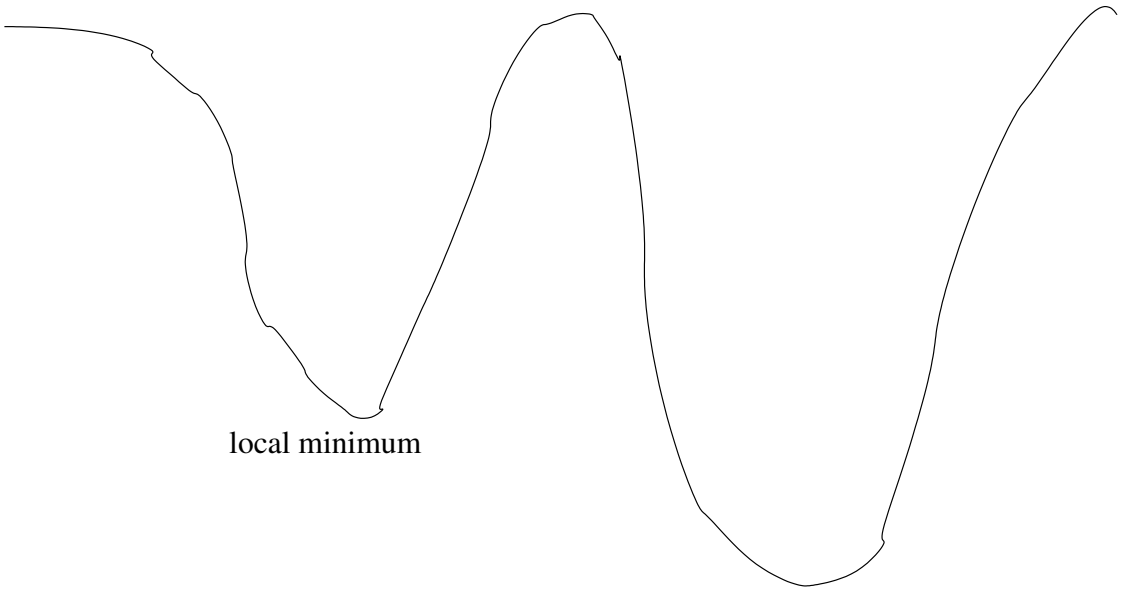
- Random improvement: Select a random move and make it if it improves on the current assignment.
- ...

UPPSALA UNIVERSITET

Local Search: Sample Meta-Heuristics

Meta-heuristics collect information on the moves made and are used for escaping local minima (of the weighted sum of the objective function and the total amount of violation of the constraints) and guiding the search towards global optima:

- Random restarts
- Tabu search
- Simulated annealing
- . . .



UPPSALA UNIVERSITET

Evaluation of Local Search

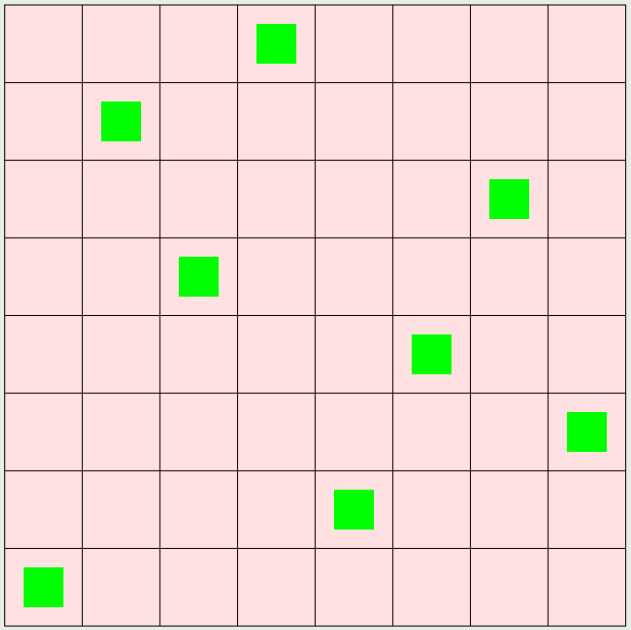
- It is hard to reuse (parts of) a local search algorithm of one problem for other problems.
- We want reusable software components!

In constraint-based local search (CBLS):

- A problem is modelled as a conjunction of constraints, which declaratively encapsulate inference algorithms specific to common combinatorial substructures and are thus reusable.
- A master search algorithm operates on the model, guided by user-indicated/designed (meta-)heuristics.

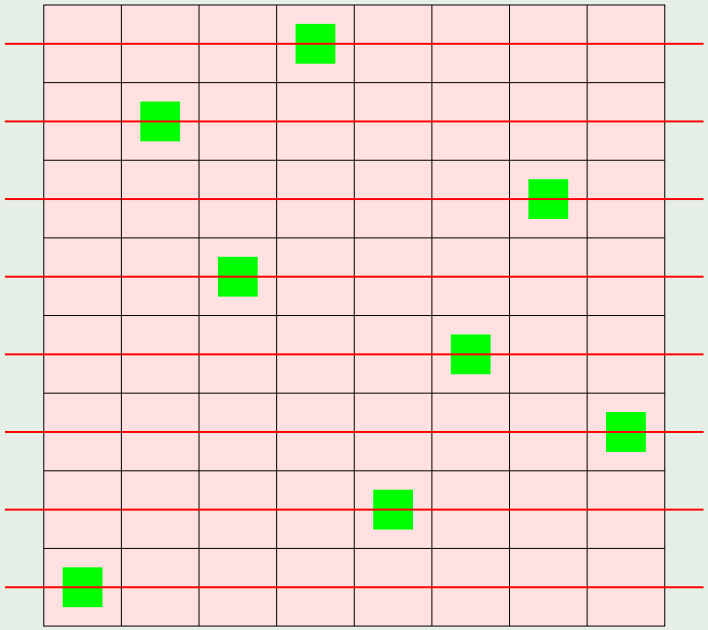
CBLS by itself makes no contributions to the design of local search (meta-)heuristics, but it facilitates their formulation and improves their reusability.

Example (8 Queens)



Place 8 queens on the chess board such that no two queens attack each other:

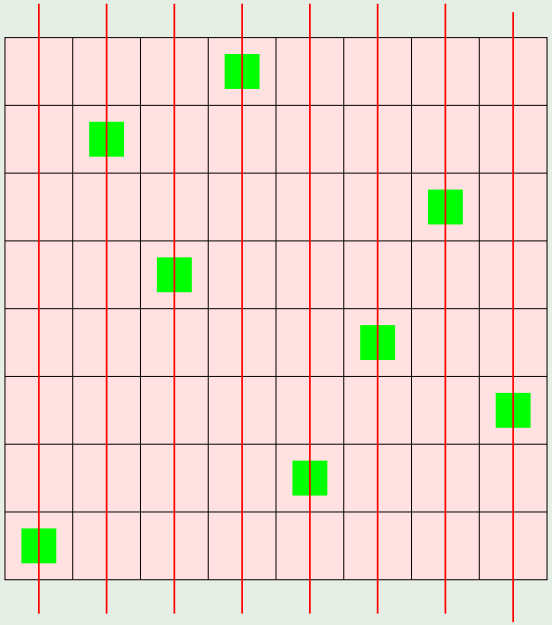
Example (8 Queens)



Place 8 queens on the chess board such that no two queens attack each other:

1. No two queens are on the same row.

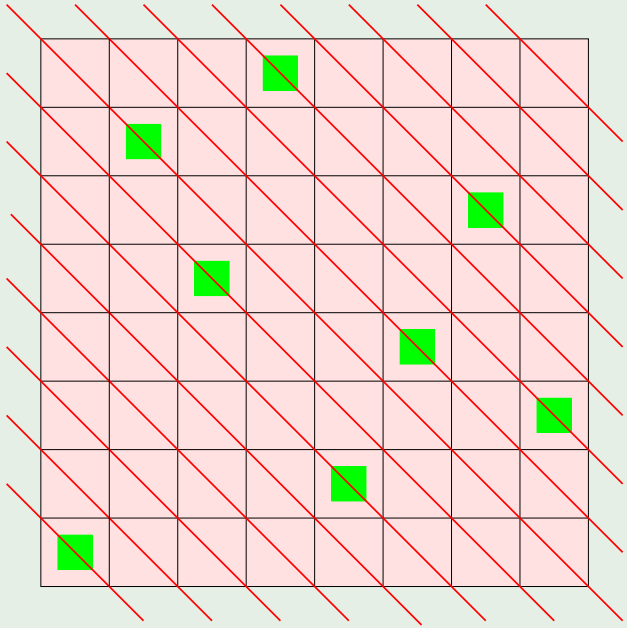
Example (8 Queens)



Place 8 queens on the chess board such that no two queens attack each other:

1. No two queens are on the same row.
2. No two queens are on the same column.

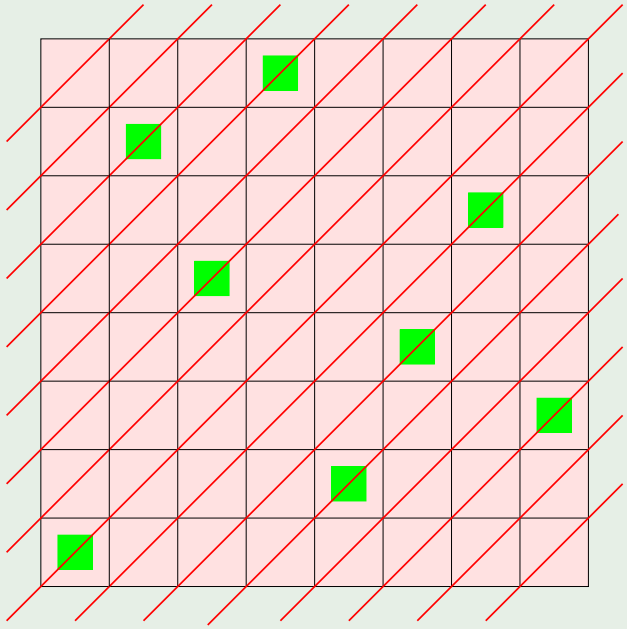
Example (8 Queens)



Place 8 queens on the chess board such that no two queens attack each other:

1. No two queens are on the same row.
2. No two queens are on the same column.
3. No two queens are on the same down diagonal.

Example (8 Queens)



Place 8 queens on the chess board such that no two queens attack each other:

1. No two queens are on the same row.
2. No two queens are on the same column.
3. No two queens are on the same down diagonal.
4. No two queens are on the same up diagonal.

Example (8 Queens: Model)

Let the row of the queen on column i be represented by a decision variable Q_i with values in $\{1, \ldots, 8\}$:

1. No two queens are on the same row:
2. No two queens are on the same column:
3. No two queens are on the same down diagonal:
4. No two queens are on the same up diagonal:

Example (8 Queens: Model)

Let the row of the queen on column i be represented by a decision variable Q_i with values in $\{1, \ldots, 8\}$:

1. No two queens are on the same row:
   ☞ $\forall i < j \in \{1, \ldots, 8\} : Q_i \neq Q_j$,
   that is $\textsc{AllDiff}(\{Q_1, \ldots, Q_8\})$
2. No two queens are on the same column:
3. No two queens are on the same down diagonal:
4. No two queens are on the same up diagonal:

Example (8 Queens: Model)

Let the row of the queen on column i be represented by a decision variable Q_i with values in $\{1, \ldots, 8\}$:

1. No two queens are on the same row:
   ☞ $\forall i < j \in \{1, \ldots, 8\} : Q_i \neq Q_j$,
   that is $\text{ALLDIFF}(\{Q_1, \ldots, Q_8\})$
2. No two queens are on the same column:
   ☞ Guaranteed by the choice of the decision variables.
3. No two queens are on the same down diagonal:
4. No two queens are on the same up diagonal:

Example (8 Queens: Model)

Let the row of the queen on column i be represented by a decision variable Q_i with values in $\{1, \ldots, 8\}$:

1. No two queens are on the same row:
   ☞ $\forall i < j \in \{1, \ldots, 8\} : Q_i \neq Q_j$,
   that is $\text{ALLDIFF}(\{Q_1, \ldots, Q_8\})$
2. No two queens are on the same column:
   ☞ Guaranteed by the choice of the decision variables.
3. No two queens are on the same down diagonal:
   ☞ $\forall i < j \in \{1, \ldots, 8\} : Q_i - i \neq Q_j - j$,
   that is $\text{ALLDIFF}(\{Q_1 - 1, \ldots, Q_8 - 8\})$
4. No two queens are on the same up diagonal:

Example (8 Queens: Model)

Let the row of the queen on column i be represented by a decision variable Q_i with values in $\{1, \ldots, 8\}$:

1. No two queens are on the same row:
   ☞ $\forall i < j \in \{1, \ldots, 8\} : Q_i \neq Q_j$,
   that is $\text{ALLDIFF}(\{Q_1, \ldots, Q_8\})$
2. No two queens are on the same column:
   ☞ Guaranteed by the choice of the decision variables.
3. No two queens are on the same down diagonal:
   ☞ $\forall i < j \in \{1, \ldots, 8\} : Q_i - i \neq Q_j - j$,
   that is $\text{ALLDIFF}(\{Q_1 - 1, \ldots, Q_8 - 8\})$
4. No two queens are on the same up diagonal:
   ☞ $\forall i < j \in \{1, \ldots, 8\} : Q_i + i \neq Q_j + j$,
   that is $\text{ALLDIFF}(\{Q_1 + 1, \ldots, Q_8 + 8\})$

Constraints in Local Search

Every constraint is equipped with:

- A constraint violation function, which gives a measure of how much the constraint is violated under the current assignment: the violation is 0 if and only if the constraint is satisfied, and positive otherwise.
- A variable violation function, which gives a measure of how much a suitable change of a decision variable may decrease the constraint violation.
- ... (to be continued)

At the constraint system level:

- The system constraint violation of a constraint system $\{c_1, \ldots, c_n\}$ is the sum of the violations of the c_i.
- The system variable violation of a variable is the sum of its variable violations in all the system constraints.

Violations

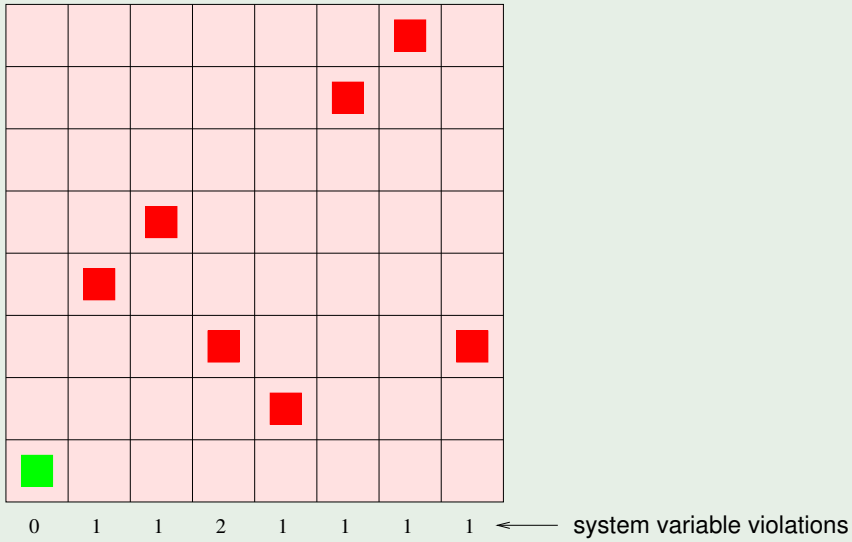
Example ($x \neq y$)

- When $x = 4$ and $y = 4$:
  - The constraint violation is 1: the constraint is violated.
  - The variable violations of x and y are both 1.
- When $x = 4$ and $y = 5$:
  - The constraint violation is 0: the constraint is satisfied.
  - The variable violations of x and y are both 0.

Example (AllDifferent($\{x_1, x_2, x_3, x_4\}$))

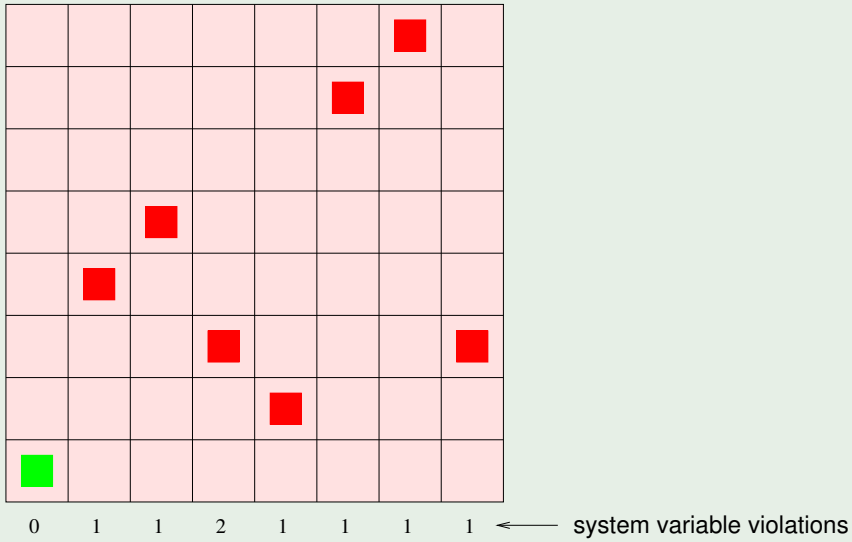
- When $x_1 = 5$, $x_2 = 5$, $x_3 = 5$, and $x_4 = 6$:
  - The constraint violation is 2, since at least two variables must be changed to reach a satisfying assignment.
  - The variable violations of x_1, x_2, and x_3 are 1.
  - The variable violation of x_4 is 0.

Example (8 Queens: Violations)



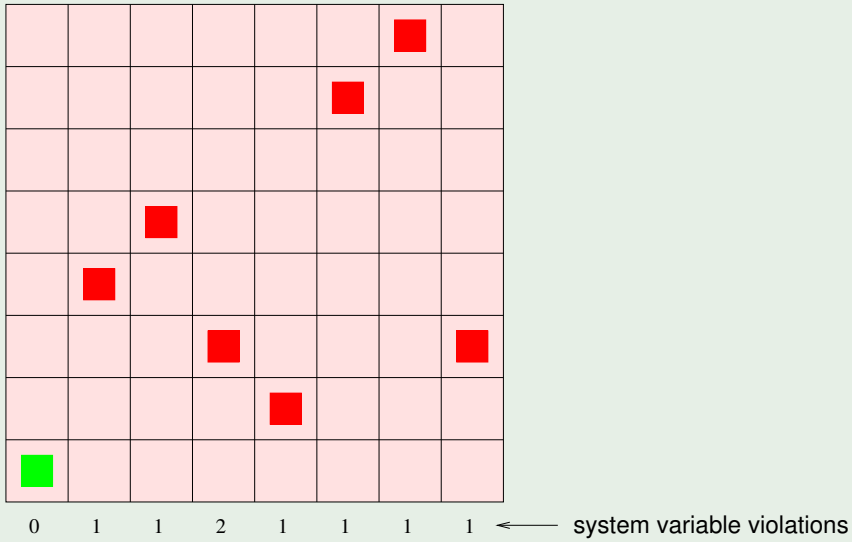
1. AllDiff($\{Q_1, \ldots, Q_8\}$)
2. AllDiff($\{Q_1 - 1, \ldots, Q_8 - 8\}$)
3. AllDiff($\{Q_1 + 1, \ldots, Q_8 + 8\}$)

Example (8 Queens: Violations)



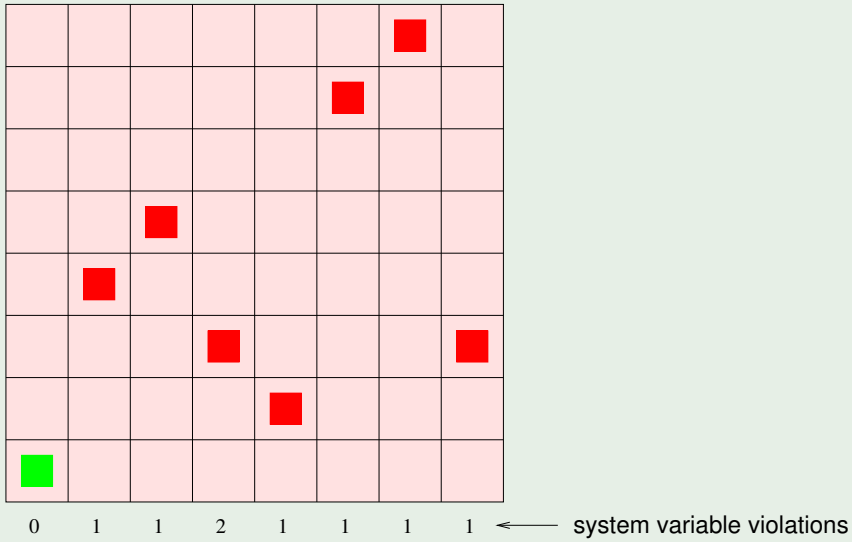
1. AllDiff($\{Q_1, \ldots, Q_8\}$)
   ☞ Violation of AllDiff($\{8, 5, 4, 6, 7, 2, 1, 6\}$) is 1.
2. AllDiff($\{Q_1 - 1, \ldots, Q_8 - 8\}$)
3. AllDiff($\{Q_1 + 1, \ldots, Q_8 + 8\}$)

Example (8 Queens: Violations)



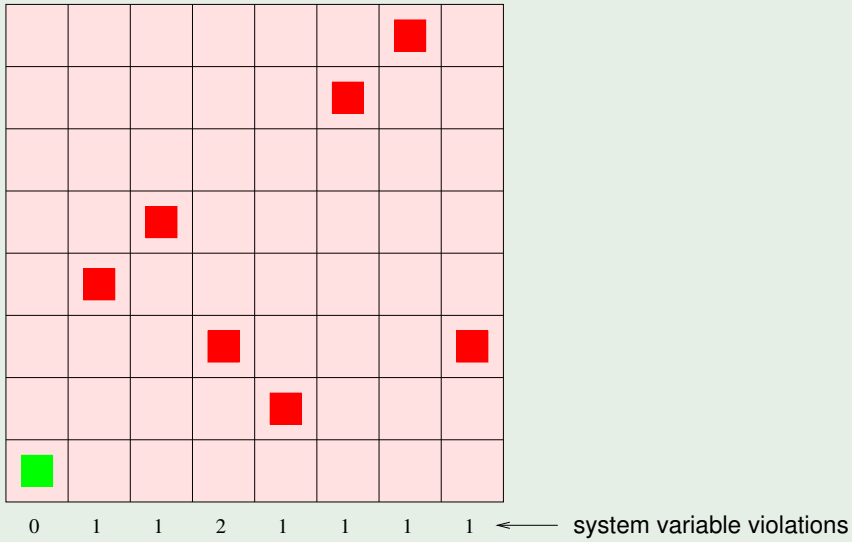
1. AllDiff($\{Q_1, \ldots, Q_8\}$)
   ☞ Violation of AllDiff($\{8, 5, 4, 6, 7, 2, 1, 6\}$) is 1.
2. AllDiff($\{Q_1 - 1, \ldots, Q_8 - 8\}$)
   ☞ Violation of AllDiff($\{7, 3, 1, 2, 2, -4, -6, -2\}$) is 1.
3. AllDiff($\{Q_1 + 1, \ldots, Q_8 + 8\}$)

Example (8 Queens: Violations)



1. AllDiff($\{Q_1, \ldots, Q_8\}$)
   ☞ Violation of AllDiff($\{8, 5, 4, 6, 7, 2, 1, 6\}$) is 1.
2. AllDiff($\{Q_1 - 1, \ldots, Q_8 - 8\}$)
   ☞ Violation of AllDiff($\{7, 3, 1, 2, 2, -4, -6, -2\}$) is 1.
3. AllDiff($\{Q_1 + 1, \ldots, Q_8 + 8\}$)
   ☞ Violation of AllDiff($\{9, 7, 7, 10, 12, 8, 8, 14\}$) is 2.

Example (8 Queens: Violations)



1. AllDiff($\{Q_1, \ldots, Q_8\}$)
   ☞ Violation of AllDiff($\{8, 5, 4, 6, 7, 2, 1, 6\}$) is 1.
2. AllDiff($\{Q_1 - 1, \ldots, Q_8 - 8\}$)
   ☞ Violation of AllDiff($\{7, 3, 1, 2, 2, -4, -6, -2\}$) is 1.
3. AllDiff($\{Q_1 + 1, \ldots, Q_8 + 8\}$)
   ☞ Violation of AllDiff($\{9, 7, 7, 10, 12, 8, 8, 14\}$) is 2.

The system constraint violation is $1 + 1 + 2 = 4$.

Constraints in Local Search (continued)

Every constraint is also equipped with:

- An assignment delta function, which gives the increase in constraint violation upon a probed $x := v$ assignment move for decision variable x and domain value v.
- A swap delta function, which gives the increase in constraint violation upon a probed $x :=: y$ swap move between two decision variables x and y.

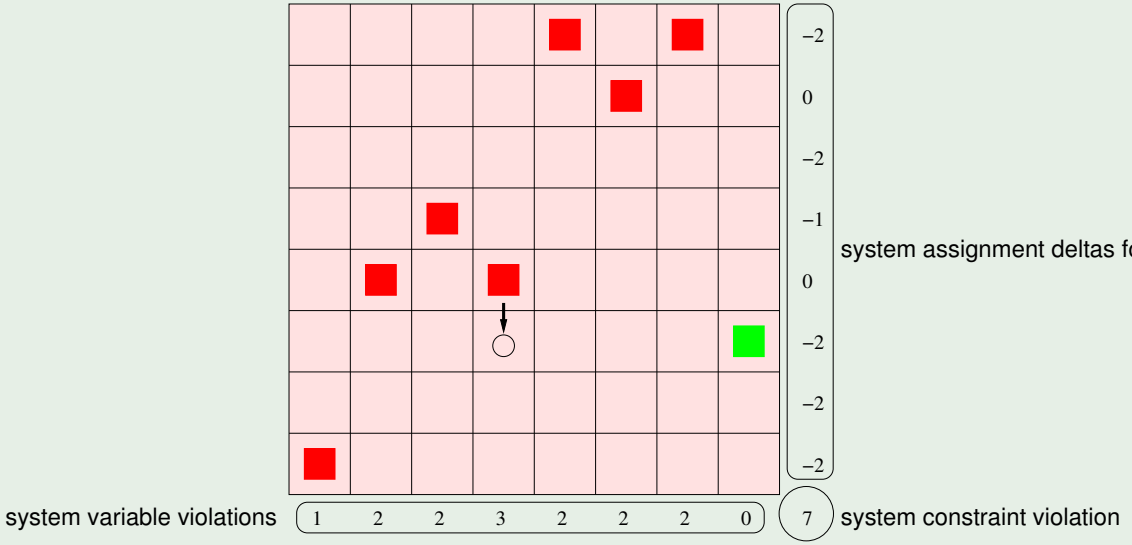
The more negative a delta the better!

At the constraint system level:

- The system assignment delta of $x := v$ in a system $\{c_1, \ldots, c_n\}$ is the sum of assignment deltas of all c_i.
- The system swap delta of $x :=: y$ in a system $\{c_1, \ldots, c_n\}$ is the sum of the swap deltas of all c_i.

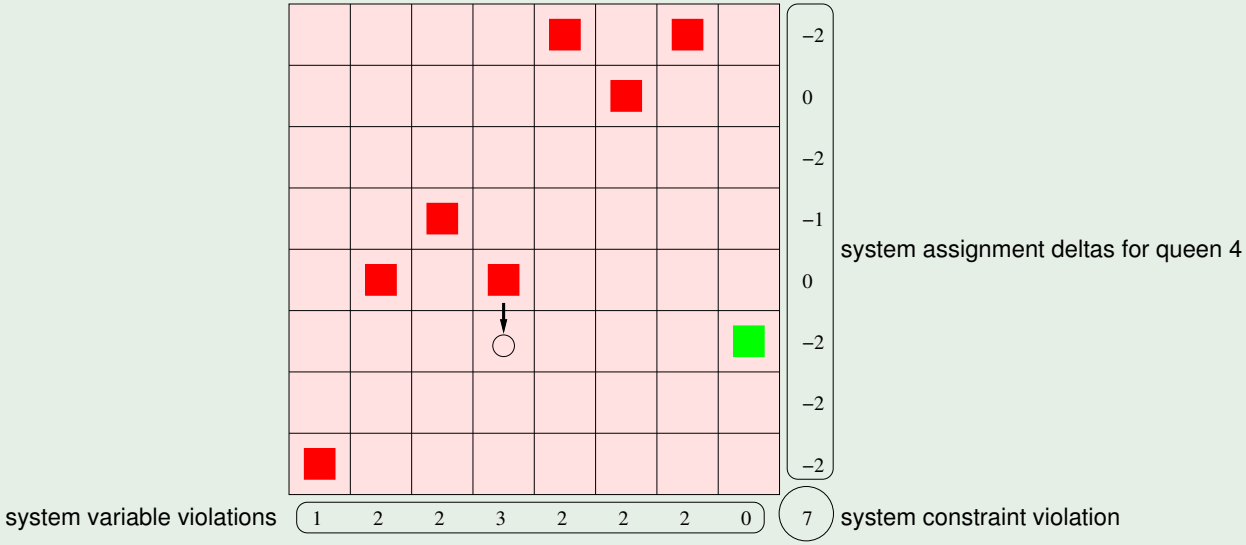
Other kinds of moves can be added.

Example (8 Queens: Differentiation)



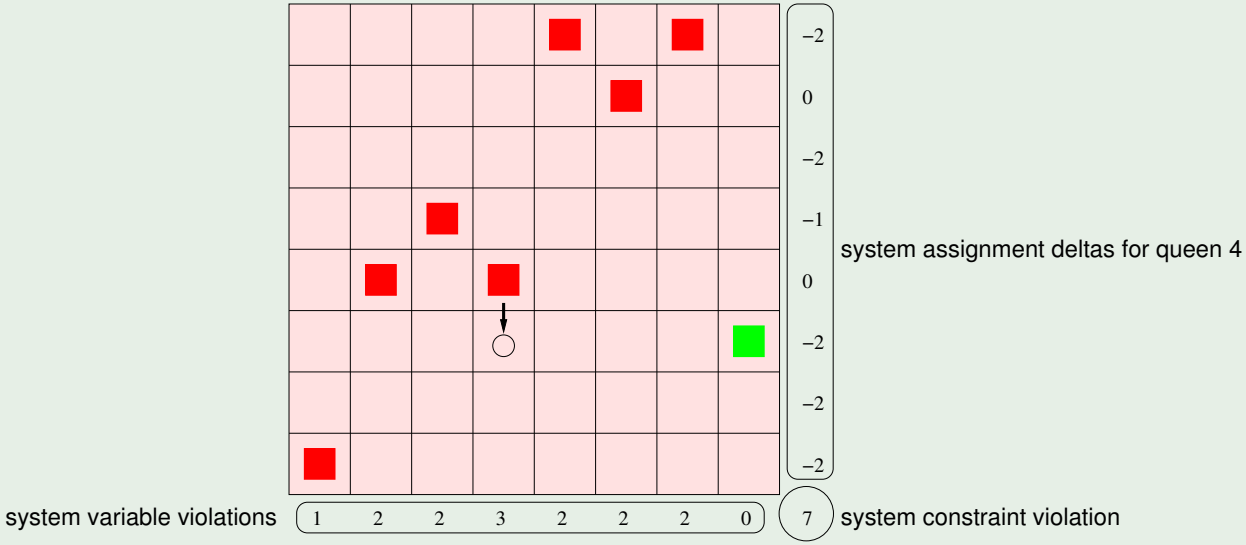
1. $\text{ALLDIFF}(\{Q_1, \ldots, Q_4, \ldots, Q_8\})$
2. $\text{ALLDIFF}(\{Q_1 - 1, \ldots, Q_4 - 4, \ldots, Q_8 - 8\})$
3. $\text{ALLDIFF}(\{Q_1 + 1, \ldots, Q_4 + 4, \ldots, Q_8 + 8\})$

Example (8 Queens: Differentiation)



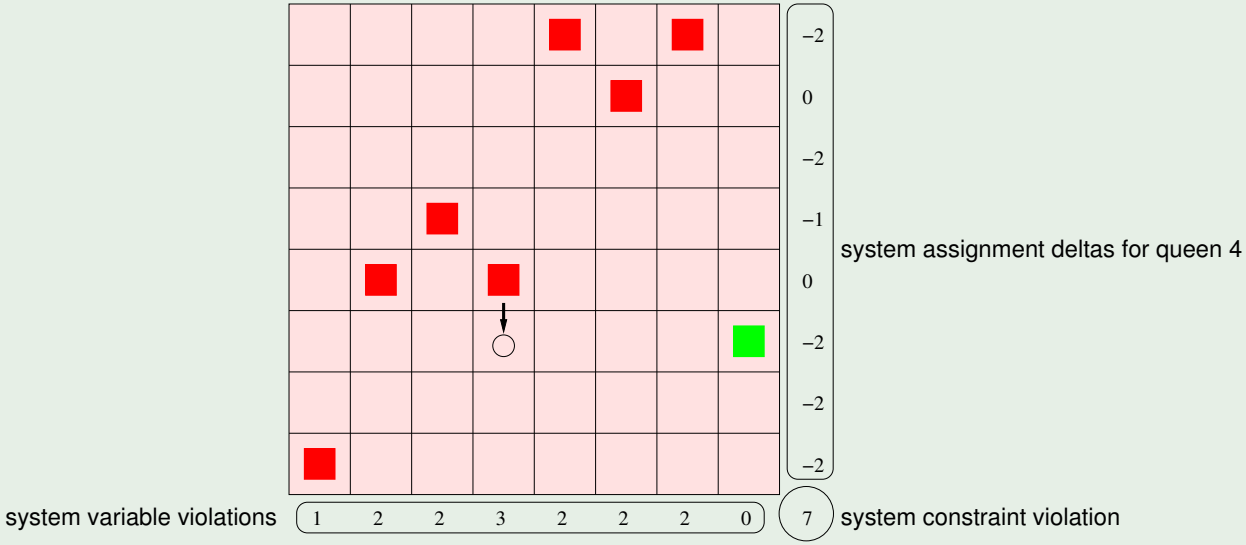
1. $\text{ALLDIFF}(\{Q_1, \ldots, Q_4, \ldots, Q_8\})$
   ☞ Delta of $Q_4 := 6$ in $\text{ALLDIFF}(\{8, 5, 4, 5, 1, 2, 1, 6\})$ is ± 0.
2. $\text{ALLDIFF}(\{Q_1 - 1, \ldots, Q_4 - 4, \ldots, Q_8 - 8\})$
3. $\text{ALLDIFF}(\{Q_1 + 1, \ldots, Q_4 + 4, \ldots, Q_8 + 8\})$

Example (8 Queens: Differentiation)



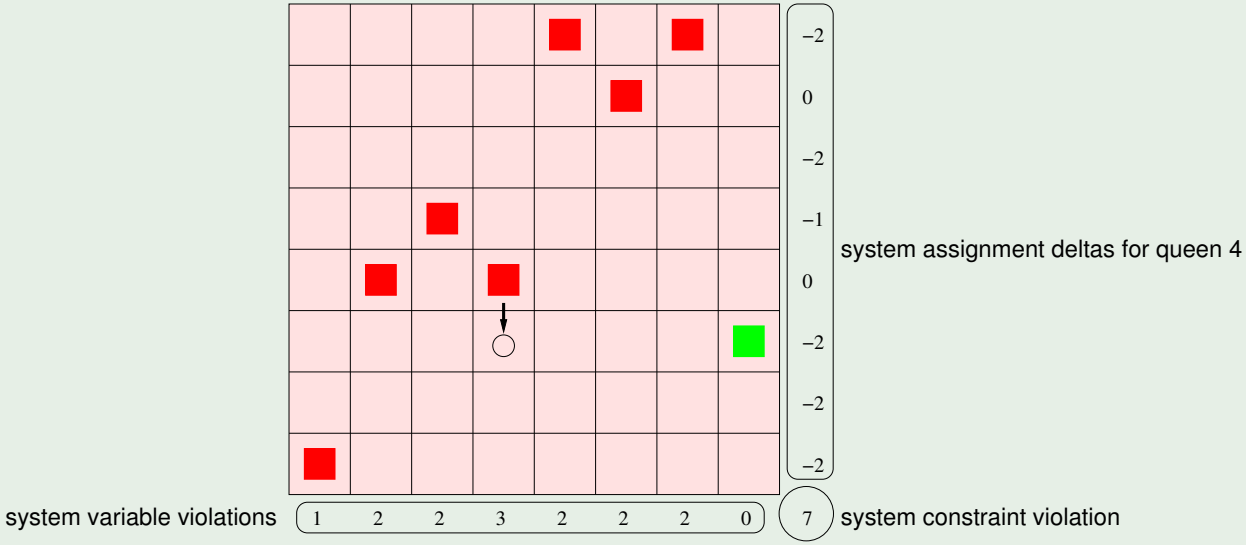
1. $\text{ALLDIFF}(\{Q_1, \ldots, Q_4, \ldots, Q_8\})$
   - ☞ Delta of $Q_4 := 6$ in $\text{ALLDIFF}(\{8, 5, 4, 5, 1, 2, 1, 6\})$ is ± 0.
2. $\text{ALLDIFF}(\{Q_1 - 1, \ldots, Q_4 - 4, \ldots, Q_8 - 8\})$
   - ☞ Delta of $Q_4 := 6$ in $\text{ALLDIFF}(\{7, 3, 1, 1, -4, -4, -6, -2\})$ is -1.
3. $\text{ALLDIFF}(\{Q_1 + 1, \ldots, Q_4 + 4, \ldots, Q_8 + 8\})$

Example (8 Queens: Differentiation)



1. $\textsc{AllDiff}(\{Q_1, \ldots, Q_4, \ldots, Q_8\})$
   - ☞ Delta of $Q_4 := 6$ in $\textsc{AllDiff}(\{8, 5, 4, 5, 1, 2, 1, 6\})$ is ± 0.
2. $\textsc{AllDiff}(\{Q_1 - 1, \ldots, Q_4 - 4, \ldots, Q_8 - 8\})$
   - ☞ Delta of $Q_4 := 6$ in $\textsc{AllDiff}(\{7, 3, 1, 1, -4, -4, -6, -2\})$ is -1.
3. $\textsc{AllDiff}(\{Q_1 + 1, \ldots, Q_4 + 4, \ldots, Q_8 + 8\})$
   - ☞ Delta of $Q_4 := 6$ in $\textsc{AllDiff}(\{9, 7, 7, 9, 6, 8, 8, 14\})$ is -1.

Example (8 Queens: Differentiation)



1. $\textsc{AllDiff}(\{Q_1, \ldots, Q_4, \ldots, Q_8\})$
   - ☞ Delta of $Q_4 := 6$ in $\textsc{AllDiff}(\{8, 5, 4, 5, 1, 2, 1, 6\})$ is ± 0.
2. $\textsc{AllDiff}(\{Q_1 - 1, \ldots, Q_4 - 4, \ldots, Q_8 - 8\})$
   - ☞ Delta of $Q_4 := 6$ in $\textsc{AllDiff}(\{7, 3, 1, 1, -4, -4, -6, -2\})$ is -1.
3. $\textsc{AllDiff}(\{Q_1 + 1, \ldots, Q_4 + 4, \ldots, Q_8 + 8\})$
   - ☞ Delta of $Q_4 := 6$ in $\textsc{AllDiff}(\{9, 7, 7, 9, 6, 8, 8, 14\})$ is -1.

The system assignment delta of $Q_4 := 6$ is $0 + (-1) + (-1) = -2$.

UPPSALA UNIVERSITET

Constraints in Local Search (end)

- The functions equipping a constraint can be used to guide the local search:
  - The constraint violation function helps to select a promising constraint for selecting variable(s) to change in a move.
  - The variable violation function helps to select promising variable(s) to change in a move.
  - The delta functions help to make a move in the good direction for a constraint or variable.
- The violation functions are the counterpart of the subsumption checking of systematic search.
- The delta functions are the counterpart of the propagators of systematic search.
- These functions must be implemented for highest time/space efficiency, as they are queried in the exploration of the neighbourhood at each iteration.

The COMET System

COMET is a language and a tool for the modelling and solving of constraint problems.

COMET has a CBLS back-end, as well as CP (systematic search with propagation) and MIP (mixed integer linear programming) back-ends:

- High-level software components (constraints) for representing constraint models of problems.
- High-level constructs for specifying search algorithms.
- An open architecture allowing user-defined extensions.

COMET (marketed by http://dynadec.com/) is free of charge for academic purposes.

UPPSALA UNIVERSITET

Modelling in COMET

Example (8 Queens: COMET Model)

```
import cotls;
Solver<LS> m();
int n = 8;
range Size = 1..n;
UniformDistribution distr(Size);
var{int} Q[Size](m,Size) := distr.get();
ConstraintSystem<LS> S(m);
S.post(alldifferent(Q));
S.post(alldifferent(all(i in Size) Q[i]-i));
S.post(alldifferent(all(i in Size) Q[i]+i));
m.close();
```

Define an array of 8 decision variables and initialise each variable with a random value in the domain $\{1, \ldots, 8\}$.

Constraint-Based Local Search in COMET

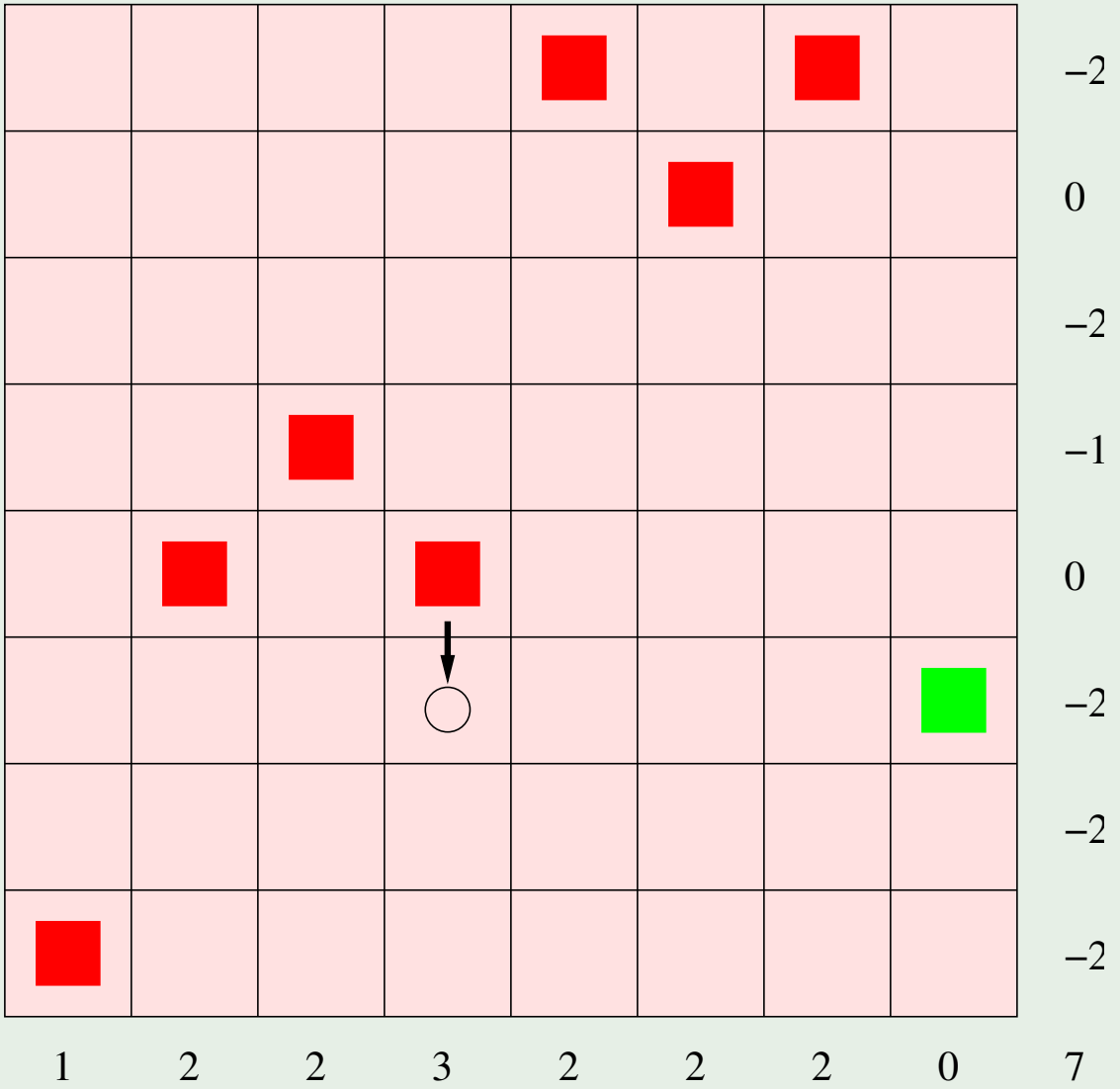
Example (8 Queens: COMET CBLS)

```
int iter = 0;
while (S.violations() > 0 && iter < 50 * n) {
  selectMax(i in 1..n)(S.violations(Q[i]))
    selectMin(r in 1..n)(S.getAssignDelta(Q[i],r))
      Q[i] := r;
  iter++;
}
```

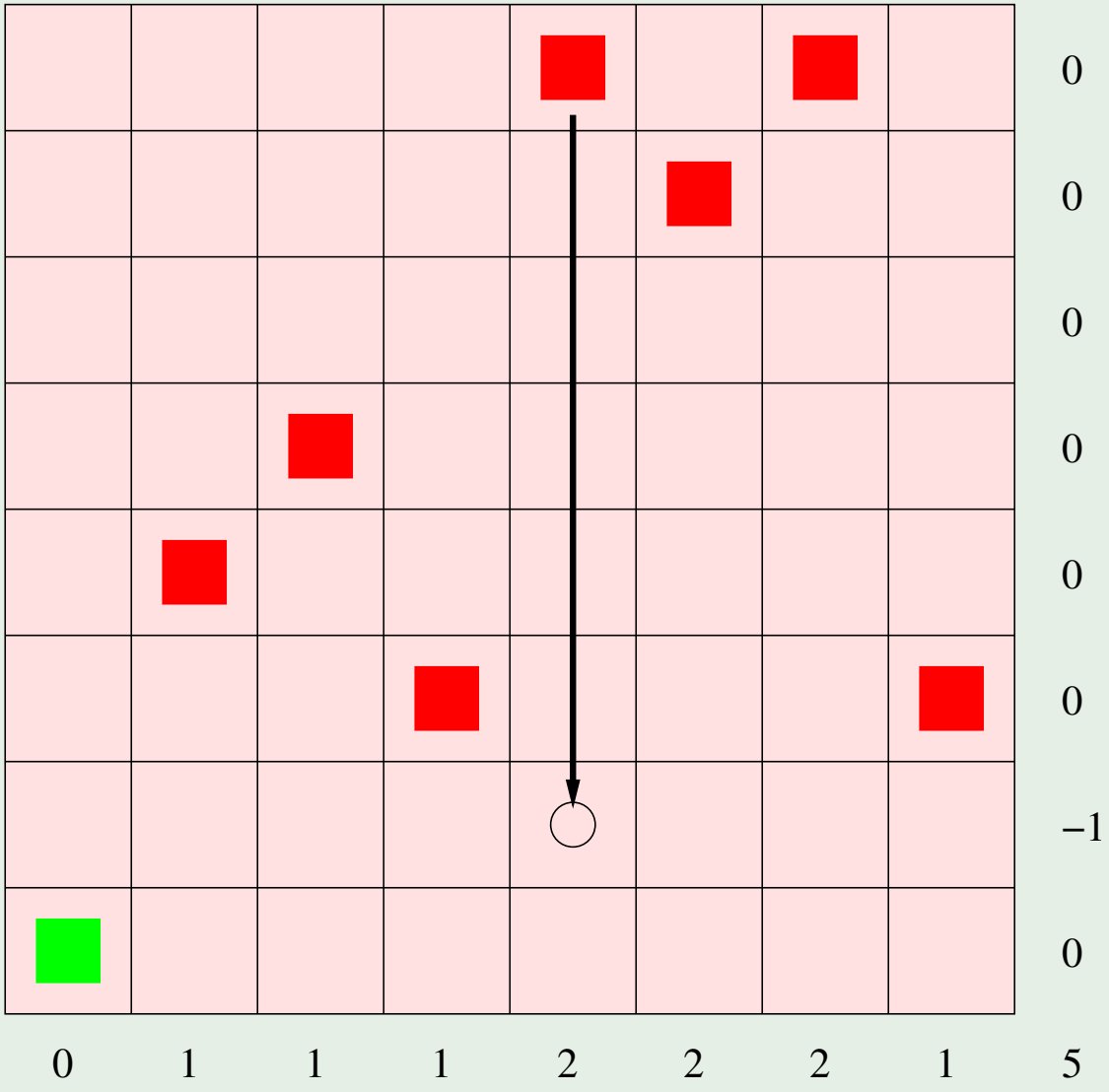
In words:

while there are a violated constraint and iterations left **do**
select a variable `Q[i]` with the maximum system violation
select a value `r` with the min system assignment delta for `Q[i]`
assign value `r` to decision variable `Q[i]`
increment the iteration counter

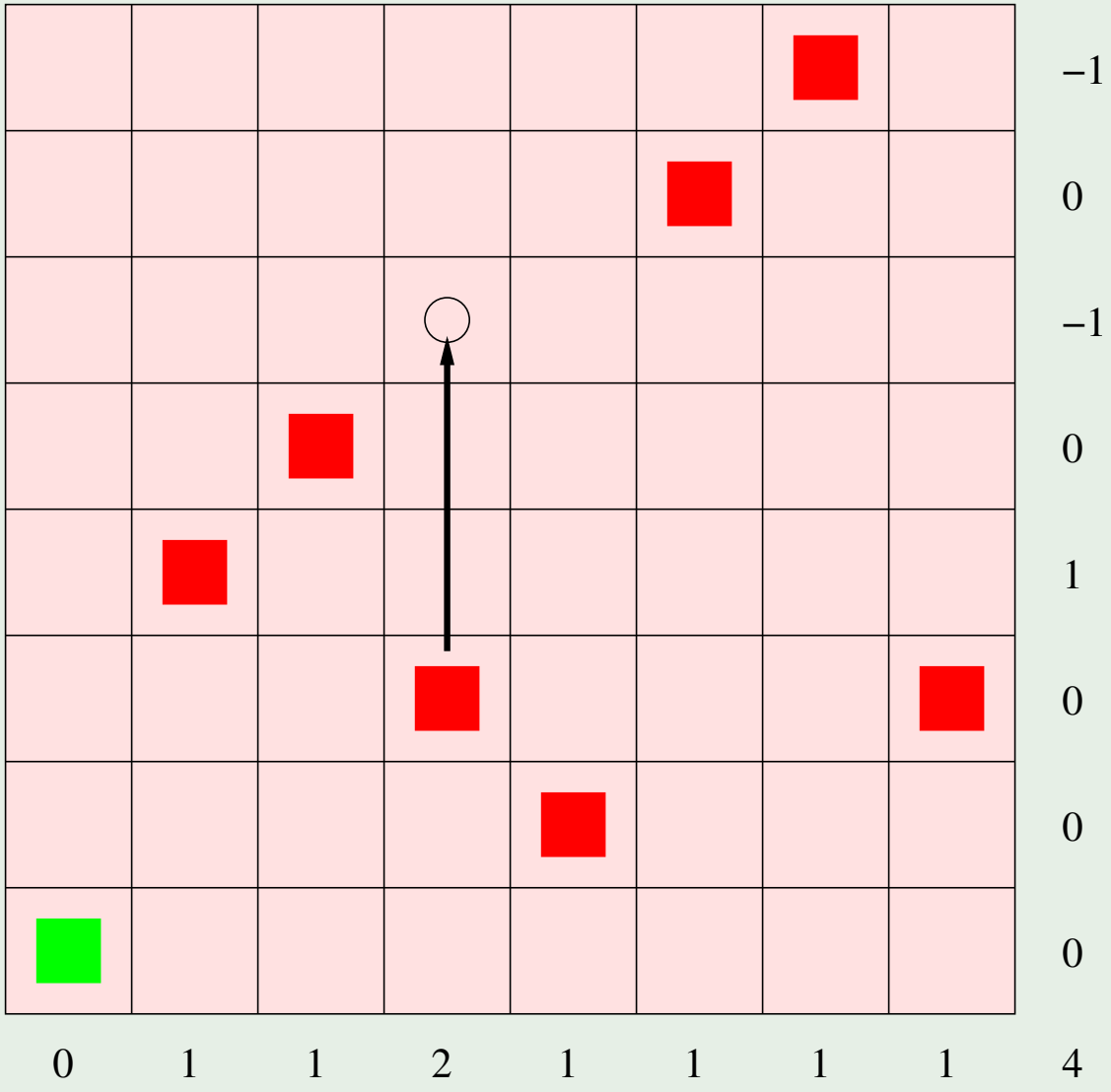
Example (8 Queens: Sample Run)



Example (8 Queens: Sample Run)



Example (8 Queens: Sample Run)

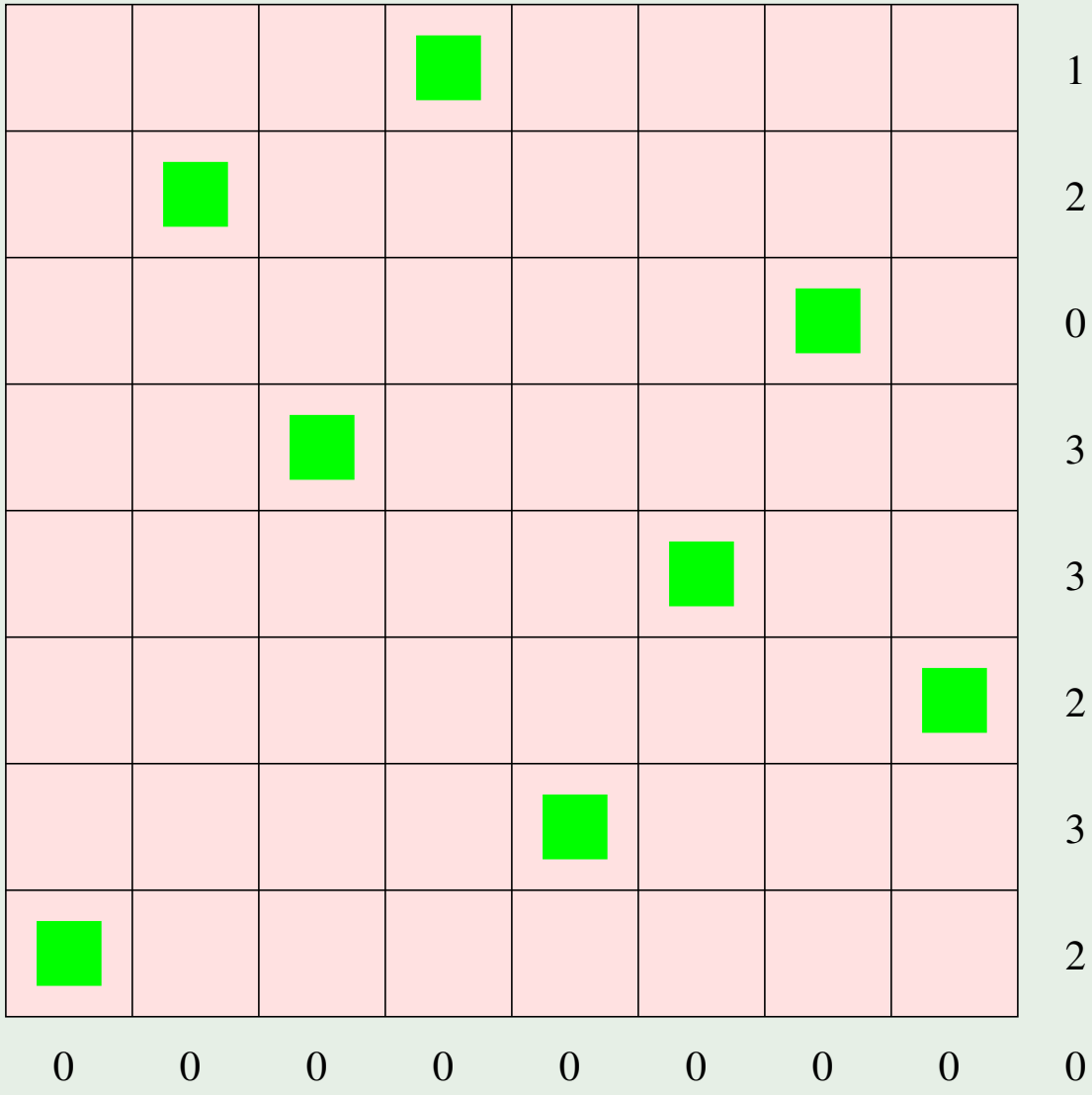


UPPSALA UNIVERSITET

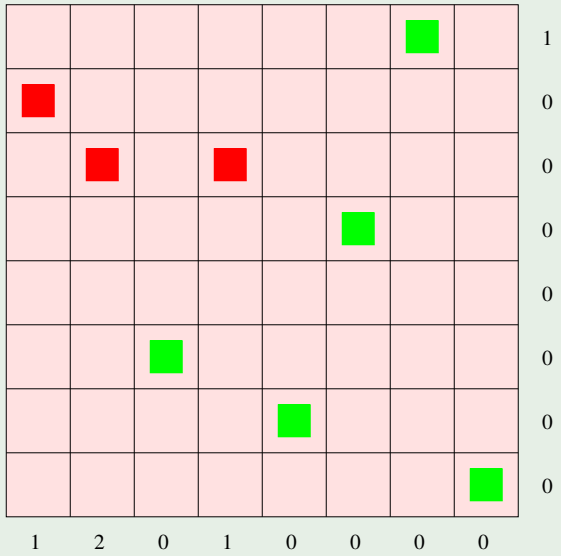
Example (8 Queens: Sample Run)

. . . and so on, until . . .

Example (8 Queens: Sample Run)



Example (8 Queens: Local Minimum)



- Queen 2 is selected, as the only most violating queen.
- Queen 2 is placed on one of rows 2 to 8, as the system violation will increase by 1 if she is placed on row 1.
- Queen 2 remains the only most violating queen!
- Queen 2 is selected over and over again.

Reference

Some of the material in this presentation is inspired from:

Pascal Van Hentenryck and Laurent Michel.
Constraint-Based Local Search.
The MIT Press, 2005.
ISBN: 0-262-22077-6