Local search meta-heuristics for combinatorial problems

Luca Di Gaspero

Dipartimento di Ingegneria Elettrica, Gestionale e Meccanica

Università degli Studi di Udine

l.digaspero@uniud.it

http://www.diegm.uniud.it/digaspero/

Outline of the talk

- CSPs, COPs & Local Search basics
- Basic Local Search techniques
- Composite Local Search techniques
- Case studies
- Future trends

Constraint Satisfaction Problems

- Given:
  - A set of variables $X = \{x_1, \ldots, x_n\}$;
  - For each variable x_i a corresponding domain $D_i = \{d_{i1}, \ldots, d_{in}\}$;
  - A set of constraints $C = \{c_1, \ldots, c_m\}$, $c_i \subseteq D_{i1} \times \ldots \times D_{ik}$;
- A CSP is the problem of finding an assignment $x_i := d_{ij}$ such that all constrains are satisfied.

The *n*-Queens problem

- Given a $n \times n$ chessboard and a set of n queens $Q = \{q_1, \ldots, q_n\}$, place the queens on the board in such a way that no pair of them attack each other.

Variables r_i
Domains $D_{r_i} = \{1, 2, \ldots, n\}$,
Constraints:
$\forall\, q_i, q_j \in Q,\ q_i \neq q_j \wedge r_i \neq r_j \wedge$
$r_i \neq r_j + j - i \wedge r_i \neq r_j + i - j$

5
4
3
2
1
1 2 3 4 5

Constrained Optimization Problems

- In the same settings as for CSP:
  - Given a cost function $f : D_1 \times \ldots \times D_n \rightarrow$;
  - A solution for a COP is an assignment that minimizes the cost function
- Common features of these problems:
  - Combinatorial problems (possibly $|D_1| \cdot \ldots \cdot |D_n|$ solutions);
  - In general, computationally intractable (NP-complete)

The Graph-Coloring Problem

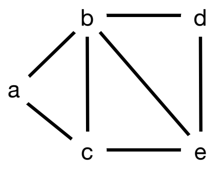
- Given a graph $G = (V, E)$, and a set of color values, find the minimum number of colors to be assigned to each vertex of the graph so that adjacent vertices are assigned different colors

  Variables c_v, $v \in V$

  Domains $D_v =$,

  Constraints: $\forall\, (u,v) \in E\ c_u \neq c_v$

  Objective function: $f(c) = |\{c_v : v \in V\}|$



Types of constraints

- Mainly two categories:
  - Hard constraints they *must* be satisfied in a feasible solution of the problem
  - Soft constraints: they *might* be not satisfied in a solution, their violation do not lead to an infeasible solution
    - However, the solutions that contain violations of the soft constraints should be penalized within the cost function

Solution techniques

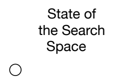
- Constructive search methods:
  - Exhaustive (backtracking-based): Forward checking, Backjumping, Branch & bound, ...
  - Incomplete (backtracking-free): Greedy construction, Heuristic repair, ...
- Selective search methods:
  - Single solution (Local Search): Hill-Climbing, Simulated Annealing, Tabu Search, ...
  - Population based (Evolutionary Algorithms): Genetic/Memetic Algorithms, Ant Colony
- Others: Integer Programming, ...

Main features

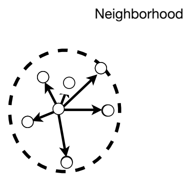
- Constructive techniques
  - More natural: better understanding of the automatic search
  - Reasonably fast for easy cases
  - Better control on the critical steps
- Selective techniques
  - Proved to be effective in many real applications
  - Provide approximate solutions
  - Revise previous solutions

An overview of Local Search

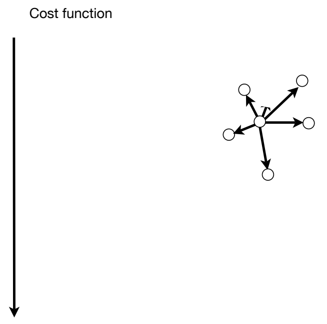
An overview of Local Search



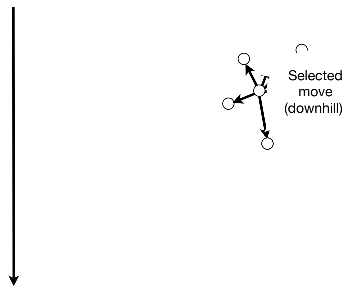
An overview of Local Search



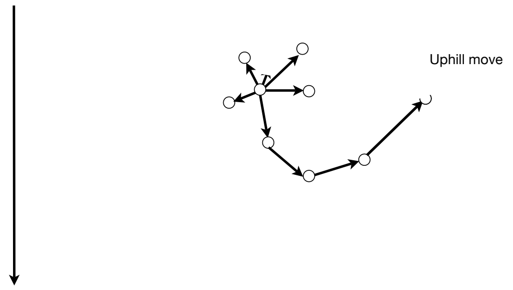
An overview of Local Search



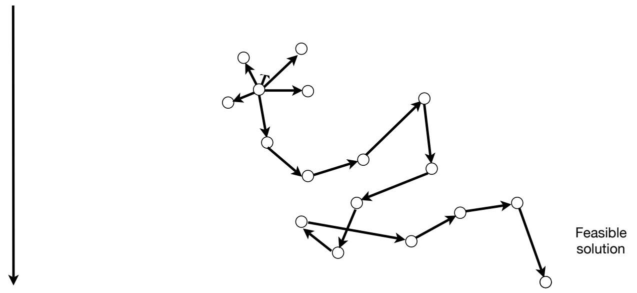
An overview of Local Search



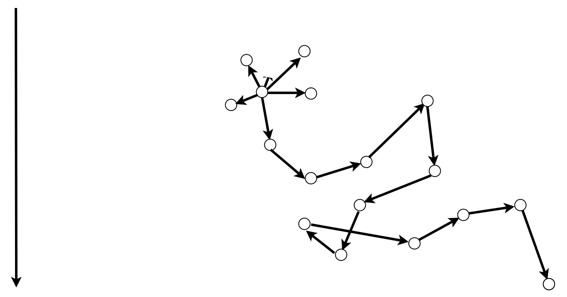
An overview of Local Search



An overview of Local Search



An overview of Local Search



From CSP/COPs to Local Search problems

- Search Space *S*: each element of it represents a possible solution of the problem. It should contain at least one feasible (or optimal) solution.
- Neighborhood Relation *N(s)*: how to move from a solution to a “close” one.
- Cost Function *F(s)*: assess the quality of each solution. Embeds distance from feasibility and, possibly, drives the search toward feasible regions.

n-Queens Local Search problem (1)

5
4
3
2
1
1 2 3 4 5

Search space: S_a: $D_1 \times D_2 \times D_3 \times D_4 \times D_5$, the assignments row_i of the row value to the *i*-th column queen q_i

$s = [1, 2, 5, 3, 2]$

n-Queens Local Search problem (1)

5
4
3
2
1
1 2 3 4 5

Cost function: account for the number of violated constraints

$F([1, 2, 5, 3, 2]) =$

n-Queens Local Search problem (1)

5
4
3
2
1

1 2 3 4 5

Cost function: account for the number of violated constraints

$F([1, 2, 5, 3, 2]) = 1$

n-Queens Local Search problem (1)

5
4
3
2
1

1 2 3 4 5

Cost function: account for the number of violated constraints

$F([1, 2, 5, 3, 2]) = 1 + 1 + 1 = 3$

n-Queens Local Search problem (1)

5
4
3
2
1

1 2 3 4 5

Neighborhood relation N_R: $S_a \rightarrow \mathcal{P}(S_a)$, assign a different row value to one queen (replace)

$s = [1, 2, 5, 3, 2]$

n-Queens Local Search problem (1)

5
4
3
2
1

1 2 3 4 5

Neighborhood relation N_R: $S_a \rightarrow \mathcal{P}(S_a)$, assign a different row value to one queen (replace)

$s = [1, 2, 5, 3, 2]$

n-Queens Local Search problem (1)

```
5
4
3
2
1
   1  2  3  4  5
```

Neighborhood relation N_R: $S_a \rightarrow \mathcal{P}(S_a)$, assign a different row value to one queen (replace)

$s = [1, 2, 5, 3, 2] \rightarrow s = [1, 2, 5, 3, 4]$

n-Queens Local Search problem (2)

```
5
4
3
2
1
   1  2  3  4  5
```

Search space: S_p: the permutations σ: $\{1, 2, 3, 4, 5\} \rightarrow \{1, 2, 3, 4, 5\}$ of row values assigned to each queen q_i

$s = [1, 2, 5, 3, 4]$

n-Queens Local Search problem (2)

```
5
4
3
2
1
   1  2  3  4  5
```

Search space: S_p: the permutations σ: $\{1, 2, 3, 4, 5\} \rightarrow \{1, 2, 3, 4, 5\}$ of row values assigned to each queen q_i

$s = [1, 2, 5, 3, 4]$

Note: $S_p \subset S_a$, constraint $\forall\ q_i, q_j \in Q \wedge r_i \neq r_j$ always satisfied

n-Queens Local Search problem (1)

```
5
4
3
2
1
   1  2  3  4  5
```

Cost function: as previously, account for the number of violated constraints

$F([1, 2, 5, 3, 2]) =$

n-Queens Local Search problem (1)

5
4
3
2
1

1 2 3 4 5

Cost function: as previously, account for the number of violated constraints

$F([1, 2, 5, 3, 2]) = 1$

n-Queens Local Search problem (1)

5
4
3
2
1

1 2 3 4 5

Cost function: as previously, account for the number of violated constraints

$F([1, 2, 5, 3, 2]) = 1 + 1 = 2$

n-Queens Local Search problem (2)

5
4
3
2
1

1 2 3 4 5

Neighborhood relation $N_S: S_p \rightarrow \mathcal{P}(S_p)$, swap the row values of two queens

$s = [1, 2, 5, 3, 4]$

Local search meta-heuristics for combinatorial problems

n-Queens Local Search problem (2)

5
4
3
2
1

1 2 3 4 5

Neighborhood relation $N_S: S_p \rightarrow \mathcal{P}(S_p)$, swap the row values of two queens

$s = [1, 2, 5, 3, 4]$

17

n-Queens Local Search problem (2)

5
4
3
2
1
1 2 3 4 5

Neighborhood relation N_S: $S_p \rightarrow \mathcal{P}(S_p)$, swap the row values of two queens

$s = [1, 2, 5, 3, 4] \rightarrow s = [1, 4, 5, 3, 2]$

Local Search: main features

- Technique independent (Local Search Model):
  - Search Space
  - Neighborhood Relation
  - Cost Function
  - Initial solution generation
- Technique specific (Meta-Heuristic):
  - Move selection & acceptance
  - Neighborhood exploration strategy
  - Prohibition mechanism
  - Stop criterion

Local Search abstract algorithm

```
procedure LocalSearch($S$,$N$,$F$)
begin
  $s_0$ := InitialSolution();
  $i$ := $0$;
  while (¬StopSearch($s_i$,$i$)) do
    $m$ := SelectMove($s_i$,$F$,$N$);
    if (AcceptableMove($m$,$s_i$,$F$)) then
      $s_{i+1} := s_i \oplus m$
    end if;
    $i := i + 1$
  end while
end procedure
```

Local Search meta-heuristics

- Basic:
  - Hill Climbing
  - Simulated Annealing
  - Tabu Search
- Composite:
  - Iterated Local Search
  - Neighborhood portfolio approach
- Hybrid:
  - With constructive algorithms: GRASP, backtracking-free hybrid
  - With genetic algorithms: ~~Memetic algorithms~~

Local search meta-heuristics for combinatorial problems