Outline

- Introduction
- Constraint propagation
- Backtracking search
- Local search

Local search

- Applies to both satisfaction and optimization problems
- Incomplete (non-systematic) algorithms
  - do *not* come with a guarantee that a solution will be found if one exists
  - *cannot* be used to find a provably optimal solution
- Finds locally optimal solutions that are not necessarily globally optimal

Notation and definitions

S	Set of states
$\mathrm{c} : S \rightarrow \Re$	Cost function
$N : S \rightarrow 2^S$	Neighborhood function

Definition: A solution $s^* \in S$ is *globally optimal* iff $\mathrm{c}(s^*) \leq \mathrm{c}(s)$, for all $s \in S$.

Definition: A solution $s^+ \in S$ is *locally optimal* iff $\mathrm{c}(s^+) \leq \mathrm{c}(s)$, for all $s \in N(s^+)$.

Graph interpretation

- Local search can be viewed as a walk in a directed, node-labeled graph
  - nodes are the elements of set of states S
  - nodes are labeled with cost values
  - arcs are given by the neighborhood function

Local search for CSPs

Given a CSP, we consider

- some constraints hard (must be satisfied)
    - set of solutions of hard constraints gives set of states *S*;   i.e., nodes in the search graph
- remaining constraints soft (moved into cost function)
    - cost function is +1 for each constraint that is not satisfied

Constraint model for 4-queens

variables:

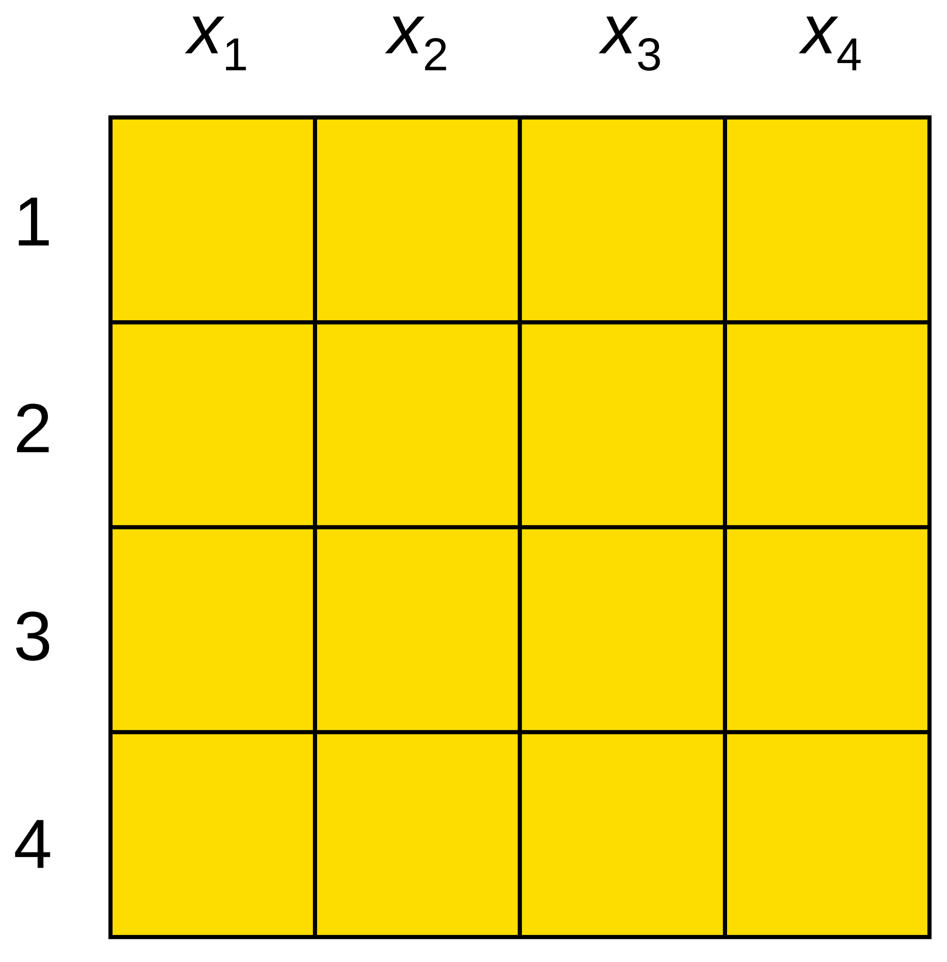
x_1, x_2 , x_3 , x_4

domains:

$\{1, 2, 3, 4\}$

constraints:

$$x_1 \neq x_2 \;\wedge\; |x_1 - x_2| \neq 1$$
$$x_1 \neq x_3 \;\wedge\; |x_1 - x_3| \neq 2$$
$$x_1 \neq x_4 \;\wedge\; |x_1 - x_4| \neq 3$$
$$x_2 \neq x_3 \;\wedge\; |x_2 - x_3| \neq 1$$
$$x_2 \neq x_4 \;\wedge\; |x_2 - x_4| \neq 2$$
$$x_3 \neq x_4 \;\wedge\; |x_3 - x_4| \neq 1$$



Local search algorithm template

$s \leftarrow$ some initial complete assignment
$k \leftarrow 0$
repeat
    $r \leftarrow$ select a neighbor of s
    if $c(r) - c(s) < t_k$ then
        $s \leftarrow r$
    $k \leftarrow k + 1$
until stopping criteria satisfied
return best s

Stopping criteria

- Maximum iterations
- Solution of low enough cost found
- Number of iterations since last (big enough) improvement is too large

Choices

- How to get an initial feasible solution?
  - random or “good”
- What neighborhood function?
  - small neighborhood is easily explored, but low quality solution may be found
  - large neighborhood is expensive to explore
- How to select “r”, the neighbor to move to?
  - first-improvement (first improving solution is selected)
  - best-improvement (solution with lowest cost is selected)

Thresholds

- Iterative improvement
  - only cost improving neighbors are accepted; i.e., $t_k = 0$, $k = 0, 1, \ldots$
- Threshold accepting
  - worst cost neighbors are accepted, but diminishes; i.e., $t_k \geq 0$, $t_k \geq t_{k+1}$
  - variation: simulated annealing. Worst cost neighbors accepted with a probability that is gradually decreased over time
  - variation: tabu search. Worst cost neighbors accepted but only if it is a legal neighbor. Set of legal neighbors restricted by a tabu list to prevent going back to a recently visited node

Improvements

- Multi-starts
  - restart the algorithm with different starting solutions, keep best solution found from all runs
- Multi-level
  - start the search in a neighborhood with a solution selected from a different neighborhood

Neighborhoods for 8-queens

Consider a permutation representation of 8-queens

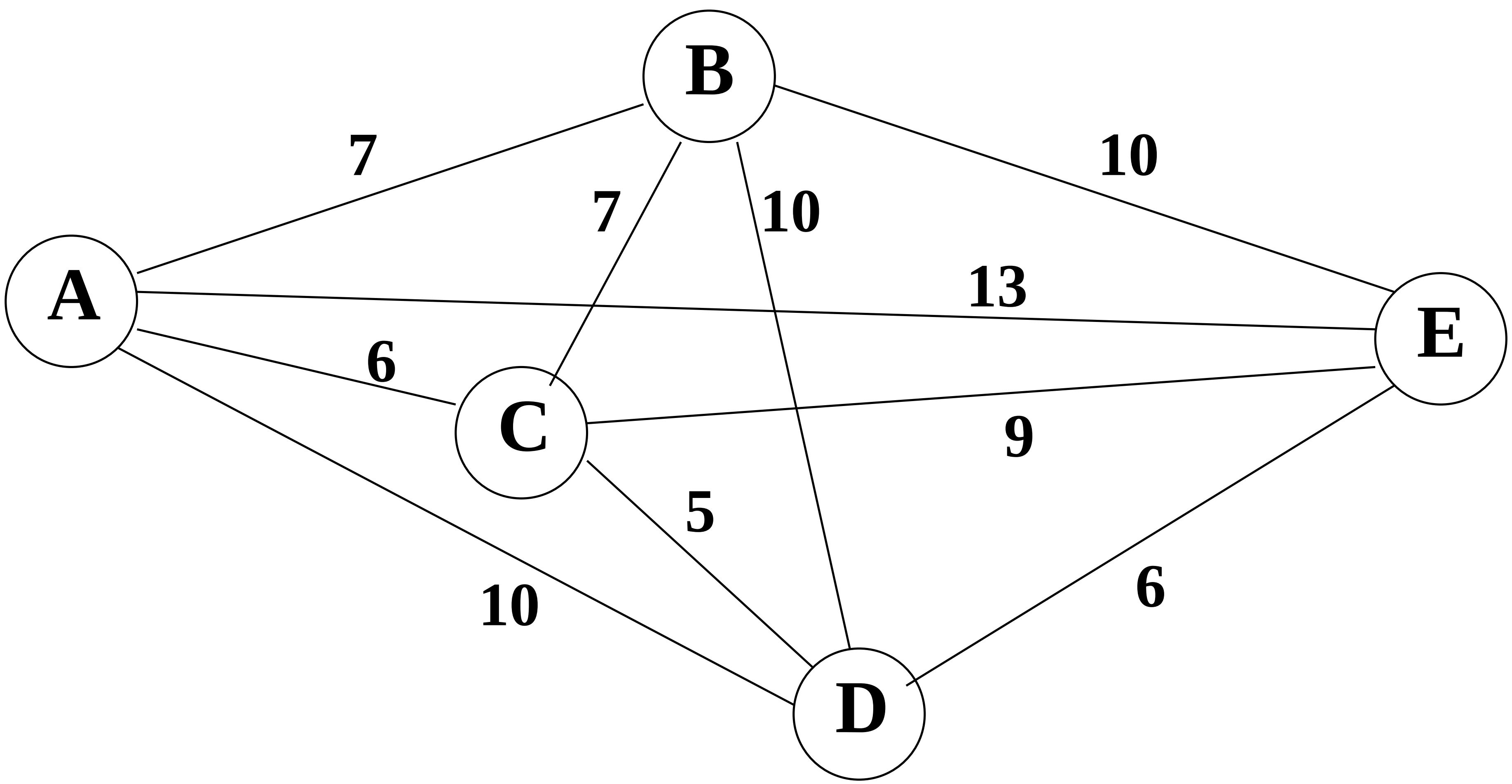
<1, 2, 3, 4, 5, 6, 7, 8>

What could be its neighbors?

<u>Transpose</u>	swap two adjacent queens e.g., <1, 2, <u>4</u>, <u>3</u>, 5, 6, 7, 8> is a neighbor	$O(n)$
<u>Insert</u>	move a queen e.g., <1, <u>5</u>, 2, 3, 4, 6, 7, 8> is a neighbor	$O(n^2)$
<u>Swap</u>	swap two queens (not necessarily adjacent) e.g., <1, <u>6</u>, 3, 4, 5, <u>2</u>, 7, 8> is a neighbor	$O(n^2)$
<u>Block insert</u>	move a subsequence of queens e.g., <1, <u>4</u>, <u>5</u>, 2, 3, 6, 7, 8> is a neighbor	$O(n^3)$

Local search for TSP

Starting at city A, find a route of minimal distance that visits each of the cities only once and returns to A.



TSP: Example theoretical results

- Definition: exact neighborhood
  - every local optimum is also a global optimum
- Exact neighborhoods
  - exact neighborhoods for TSP must be exponential in size
  - unless P = NP, polynomially searchable (an improving move can be found in polynomial time) exact neighborhoods cannot exist
- Non-exact neighborhoods
  - if a neighborhood is not exact, the cost of a local optimum can be arbitrarily far from a global optimum
  - if a neighborhood is not exact, local search can take an exponential number of steps to read a local optimum

TSP: Example empirical results

- Best local search algorithms
  - get within 1.5-2.5% of optimal on random and benchmark instances
  - can solve instances with 1m cities in under 1 hour