CMU 15-781
Lecture 2a: Local Search
Teacher: Gianni A. Di Caro
PATH SEARCH VS. LOCAL SEARCH

- The algorithms discussed so far are designed to find a goal state from a start state: the path to the goal constitutes a solution to the search problem.
- In many problems the path doesn’t matter: the goal state itself is the solution.
- **State space** = set of “complete” configurations
  - **Optimization problems**: Find optimal configuration (objective or cost function)
  - **Constraint Satisfaction Problems**: Find configurations satisfying (all or the highest number of) constraints.
PATH SEARCH VS. LOCAL SEARCH

• Local search algorithms at each step consider a single “current” state, and try to improve it by moving to one of its neighbors ➔ Iterative improvement algorithms

• Pros and cons
  o No complete (no optimal), except with random restarts
  o Space complexity $\mathcal{O}(b)$
  o Time complexity $\mathcal{O}(d)$, $d$ can be $\infty$
  o Can perform well also in large (infinite, continuous) spaces
  o Relatively easy to implement
STATE-SPACE LANDSCAPE

Objective function

global maximum

shoulder

local maximum

“flat” local maximum

neighborhood

current state

state space
HILL-CLIMBING SEARCH

Like climbing Everest in thick fog with amnesia

function HILL-CLIMBING(problem) returns a state that is a local maximum
inputs: problem, a problem
local variables: current, a node
neighbor, a node

current ← MAKE-NODE(Initial-State[problem])
loop do
    neighbor ← a highest-valued successor of current
    if VALUE[neighbor] ≤ VALUE[current] then return STATE[current]
    current ← neighbor
end

• Move in the direction of increasing value (up the hill)
• Terminate when no neighbor has higher value
• Greedy (myopic) local search
CSP Example: N-Queens

Put n queens on an $n \times n$ board with no two queens on the same row, column, or diagonal.

*State:* Position of the n queens, one per column (or row)

*Successor states:* generated by moving a single queen to another square in its column ($n(n-1)$)

*Cost of a state:* the number of constraint violations
N-Queens

<table>
<thead>
<tr>
<th></th>
<th>18</th>
<th>12</th>
<th>14</th>
<th>13</th>
<th>13</th>
<th>12</th>
<th>14</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>16</td>
<td>13</td>
<td>15</td>
<td>12</td>
<td>14</td>
<td>12</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>12</td>
<td>18</td>
<td>13</td>
<td>15</td>
<td>12</td>
<td>14</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>14</td>
<td>13</td>
<td>16</td>
<td>13</td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>17</td>
<td>15</td>
<td>14</td>
<td>16</td>
<td>16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>16</td>
<td>18</td>
<td>15</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>14</td>
<td>15</td>
<td>15</td>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>14</td>
<td>13</td>
<td>17</td>
<td>12</td>
<td>14</td>
<td>12</td>
<td>16</td>
<td></td>
</tr>
</tbody>
</table>

State with 17 conflicts, showing the #conflicts by moving a queen within its column, with best moves in red

<p>| | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Local optimum: state that has only one conflict, but every move leads to larger #conflicts
HILL-CLIMBING PERFORMANCE ON N-QUEENS

• Hill-climbing can solve large instances of $n$-queens ($n = 10^6$) in a few (ms)seconds

• 8 queens statistics:
  o State space of size $\approx 17$ million
  o Starting from random state, steepest-ascent hill climbing solves 14% of problem instances
  o It takes 4 steps on average when it succeeds, 3 when it gets stuck
  o When “sideways” moves are allowed, performance improves ...
  o When multiple restarts are allowed, performance improves even more
HILL-CLIMBING CAN GET STUCK!

sideways moves ($M$):

$M=100 \rightarrow 94\%$ solved instances for the 8-queens!

21 steps avg. on success
64 steps avg. on “failure”

+ random restarts:
100% solved instances
28 steps avg.
HILL-CLIMBING CAN GET STUCK!

Diagonal ridges:
From each local maximum all the *available* actions point downhill, but there is an uphill path!

Zig-zag motion, very long ascent time!

**Gradient ascent** doesn’t have this issue: *all* state vector components are (potentially) changed when moving to a successor state, climbing can follow the direction of the ridge.
VARIANTS OF HILL-CLIMBING

• Sideways moves: if no uphill moves, allow moving to a state with the same value as the current one (escape shoulders)

• Stochastic hill-climbing: selection among the available uphill moves is done randomly (uniform, proportional, soft-max, \(\varepsilon\)-greedy, ...) to be “less” greedy

• First-choice hill-climbing: successors are generated randomly, one at a time, until one that is better than the current state is found (deal with large neighborhoods)

• Random-restart hill climbing: probabilistically complete

If at first you don’t succeed, try, try again!
TRAJECTORIES, DIFFICULTIES
NEIGHBORHOOD

• A mapping (rule) that associate two states \((s,s')\)
• It should preserve a certain degree of correlation between the value of \(s\) and that of \(s'\)
• It should balance size and search
GOOD VS. REALISTIC SCENARIOS

With any starting solution Local Search finds the global optimum.

But some starting solutions lead Local Search to a local minimum.
**Example neighborhoods**

1-flip neighborhood, for 0-1 vectors

\[ x = (0,1,0) \]
\[ N(x) \]
\[ (1,1,0) \]
\[ (0,0,0) \]
\[ (0,1,1) \]

2-swap neighborhood, for permutation vectors

\[ x = (2,1,3) \]
\[ N(x) \]
\[ (3,1,2) \]
\[ (2,3,1) \]
\[ (1,2,3) \]

**k-exchange neighborhood** (for TSP and similar problems): The neighborhood \( N(s) \) of a solution \( s \) is the set of solutions \( s' \) that differ from \( s \) up to \( k \) solution components.
Optimization example: TSP

Find the Hamiltonian tour of minimal cost

Every cyclic permutation of $n$ integers is a feasible solution

If two nodes are not connected, they can be seen as connected by an arc of $\infty$ length!

$$\pi_1 = (1, 3, 4, 2, 6, 5, 7, 1), \quad \pi_2 = (2, 3, 4, 5, 6, 7, 1, 2)$$

$$c(\pi_2) = d_{23} + d_{34} + d_{45} + d_{56} + d_{67} + d_{71} + d_{12} = 93$$

Read also as set of edges: $$\{(2,3), (3,4), (4,5), (5,6), (6,7), (7,1), (1,2)\}$$
K-exchange neighborhood:

\( N(s) \) is the set of tours \( s' \) can be obtained from \( s \) by exchanging \( k \) edges in \( s \) with \( k \) edges in \( E \setminus \{s\} \) (\( E \) is the graph’s edge set)

Each \( s' \) is obtained deleting a selected set of \( k \) edges in \( s \) and rewiring the resulting fragments into a complete tour by inserting a different set of \( k \) edges

\[ \binom{n}{k} \text{ possible ways to drop } k \text{ edges in a tour} \]
\[ (k - 1)!2^{k-1} \text{ ways to relink the disconnected paths} \]
2-OPT local search

- Two edges, \((i,j)\) and \((l,k)\), are selected, removed, and replaced by two other edges \((i,k)\) and \((j,l)\) (or, \((k,i)\), \((l,j)\))
- One of the two paths needs to get reverted!
- Gain: \((i,k) + (j,l) - (i,j) - (k,l)\)
- \(n(n-1)=O(n^2)\) possible successors in the 2-exchange neighborhood ➔ quadratic search complexity for each single 2-opt step move
2-OPT LOCAL SEARCH

Initial Tour

Subpaths

Subtour is inverted

→ New edges

--- Edges to be removed

Step 1

--- Edges to be removed

Step 2

Step 3

→ New edges
→ New edges
3-OPT LOCAL SEARCH

- Including the initial solution, as well as 2-opt moves, there is a total of $2^3$ feasible rewirings for each selected triple of edges
- $n(n - 1)(n - 2) = O(n^3)$ successors
- One move does not revert the path → appropriate for asymmetric TSP
2-OPT VS. 3-OPT
4-OPT DOUBLE BRIDGE

- Does not revert the tours
- Computational complexity of a single step: $O(n^2)$
- Often used in conjunction with 2-opt and 3-opt
### (Some) Performance Comparison

<table>
<thead>
<tr>
<th></th>
<th>(N = 10^2)</th>
<th>(10^{2.5})</th>
<th>(10^3)</th>
<th>(10^{3.5})</th>
<th>(10^4)</th>
<th>(10^{4.5})</th>
<th>(10^5)</th>
<th>(10^{5.5})</th>
<th>(10^6)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Random Euclidean Instances</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GR</td>
<td>19.5</td>
<td>18.8</td>
<td>17.0</td>
<td>16.8</td>
<td>16.6</td>
<td>14.7</td>
<td>14.9</td>
<td>14.5</td>
<td>14.2</td>
</tr>
<tr>
<td>CW</td>
<td>9.2</td>
<td>10.7</td>
<td>11.3</td>
<td>11.8</td>
<td>11.9</td>
<td>12.0</td>
<td>12.1</td>
<td>12.1</td>
<td>12.2</td>
</tr>
<tr>
<td>CHR</td>
<td>9.5</td>
<td>9.9</td>
<td>9.7</td>
<td>9.8</td>
<td>9.9</td>
<td>9.8</td>
<td>9.9</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>2-Opt</td>
<td>4.5</td>
<td>4.8</td>
<td>4.9</td>
<td>4.9</td>
<td>5.0</td>
<td>4.8</td>
<td>4.9</td>
<td>4.8</td>
<td>4.9</td>
</tr>
<tr>
<td>3-Opt</td>
<td>2.5</td>
<td>2.5</td>
<td>3.1</td>
<td>3.0</td>
<td>3.0</td>
<td>2.9</td>
<td>3.0</td>
<td>2.9</td>
<td>3.0</td>
</tr>
<tr>
<td><strong>Random Distance Matrices</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GR</td>
<td>100</td>
<td>160</td>
<td>170</td>
<td>200</td>
<td>250</td>
<td>280</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>2-Opt</td>
<td>34</td>
<td>51</td>
<td>70</td>
<td>87</td>
<td>125</td>
<td>150</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>3-Opt</td>
<td>10</td>
<td>20</td>
<td>33</td>
<td>46</td>
<td>63</td>
<td>80</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

SIMULATED ANNEALING

• *Escape from local optima* by accepting, with a probability that decreases during the search, also moves that are worse than the current solution (going downhill!)

• Stochastic, solution-improvement *metaheuristic for global optimization*

• Inspired by the process of *annealing* of solids in metallurgy:
  • The temperature of the solid is increased until it melts
  • The temperature is *slowly decreased through a quasi-static process* until the solid reaches a minimal energy state in which a regular crystal structure appears
SIMULATED ANNEALING

procedure Simulated_Annealing()
    \( S = \{ \text{set of all feasible solutions} \} \);
    \( \mathcal{N} = \text{neighborhood structure defined over } S \);
    \( s \leftarrow \text{Generate a starting feasible solution}; \quad \text{// e.g., with a construction heuristic} \)
    \( s^{\text{best}} \leftarrow s; \)
    \( T \leftarrow \text{Determine a starting value for temperature}; \)
    while (NOT YET frozen) \quad // termination criterion
        while (NOT YET AT \textit{equilibrium FOR THIS TEMPERATURE})
            \( s' \leftarrow \text{Choose a random solution from neighborhood } \mathcal{N}(s); \quad \text{// e.g., select a random 2-opt move} \)
            \( \Delta E \leftarrow f(s') - f(s); \)
            if (\( \Delta E \leq 0 \)) \quad // downhill, locally improving move
                \( s \leftarrow s'; \)
                if (\( f(s) < f(s^{\text{best}}) \))
                    \( s^{\text{best}} \leftarrow s; \)
            else \quad // uphill move
                \( r \leftarrow \text{Choose a random number uniformly from } [0,1]; \)
                if (\( r < e^{-\Delta E / T} \)) \quad // accept the uphill, not improving, move
                    \( s \leftarrow s'; \)
        end if
    end while
    \( T \leftarrow \text{Lower the temperature according to the selected cooling schedule}; \)
end while
return \( s^{\text{best}}; \)
Effect of Temperature

\[ \Delta E = f(s') - f(s) \]

Acceptation probability

- \( T = 1 \)
- \( T = 10 \)
- \( T = 50 \)
- \( T = 100 \)
Properties

- Acceptation probability depends on the current candidate solution and on the previous one → The solution sequence can be seen as a Markov chain.
- If $T_k$ decreases “slowly enough” the algorithm will asymptotically converge in probability to the global optimum → Asymptotically complete and optimal.
- Convergence can be guaranteed if at each step $T$ drops no more quickly than $C / \log n$, $C=constant$, $n$ # of steps so far.
- Cooling schedules that work in practice often lack of convergence properties :-(
- For TSP, $n!$ solutions, the required # of iterations $k = O\left(n^{n^{2n-1}}\right)$. 
A POPULAR TEMPERATURE SCHEDULE: EXPONENTIAL COOLING

• Temperature drops roughly as $C^n$, $C \in (0, 1)$

• A fixed number of moves is performed at each temperature, after which one arbitrarily declares “equilibrium” and reduces the temperature by a standard factor, $T_{k+1} = \gamma T_k$, $\gamma \in [0,1]$ is a constant ($\gamma = 0.95$ is a common choice)

• Under an exponential cooling regime, the temperature reaches values sufficiently close to zero after a polynomially-bounded amount of time and the “frozen” state can be declared
EXPONENTIAL COOLING

Start $T$? Temperature length?
### (Some) Performance Comparison

<table>
<thead>
<tr>
<th>Variant</th>
<th>Average Percent Excess</th>
<th>Running Time in Seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$10^2$</td>
<td>$10^{2.5}$</td>
</tr>
<tr>
<td><strong>SA$_1$ (Baseline Annealing)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\alpha = 1$</td>
<td>3.4</td>
<td>3.7</td>
</tr>
<tr>
<td><strong>SA$_1$ + Pruning</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\alpha = 1$</td>
<td>2.7</td>
<td>3.2</td>
</tr>
<tr>
<td><strong>SA$_1$ + Pruning</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\alpha = 10$</td>
<td>1.7</td>
<td>1.9</td>
</tr>
<tr>
<td><strong>SA$_2$ (Pruning + Low Temp)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\alpha = 10$</td>
<td>1.6</td>
<td>1.8</td>
</tr>
<tr>
<td><strong>SA$_2$</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\alpha = 40$</td>
<td>1.3</td>
<td>1.5</td>
</tr>
<tr>
<td><strong>SA$_2$</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\alpha = 100$</td>
<td>1.1</td>
<td>1.3</td>
</tr>
<tr>
<td><strong>2-Opt</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4.5</td>
<td>4.8</td>
</tr>
<tr>
<td><strong>Best of 1000 2-Opt</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.9</td>
<td>2.8</td>
</tr>
<tr>
<td><strong>Best of 10000 2-Opt</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.7</td>
<td>2.6</td>
</tr>
<tr>
<td><strong>3-Opt</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2.5</td>
<td>2.5</td>
</tr>
<tr>
<td><strong>Best of 1000 3-Opt</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td>1.3</td>
</tr>
<tr>
<td><strong>Best of 10000 3-Opt</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.9</td>
<td>1.2</td>
</tr>
<tr>
<td><strong>Lin-Kernighan</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.5</td>
<td>1.7</td>
</tr>
<tr>
<td><strong>Best of 100 LK’s</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.9</td>
<td>1.0</td>
</tr>
</tbody>
</table>
SUGGESTIONS FOR FURTHER READINGS

M. Gendreau and J.-Y. Potvin (Editors), Handbook of Metaheuristics, Springer 2010


E. Aarts and J. Lenstra (Editors), Local Search in Combinatorial Optimization, Princeton University Press, 2003