Chapter 7
Stochastic Local Search

Michaja Pressmar
13.11.2014
Motivation

\( n \)-queens with Backtracking:

- guarantees to find all solutions
- reaches limit for big problems: Best backtracking methods solve up to 100-queens
- Stochastic search: 1 million queens solvable in less than a minute
Systematic vs. Stochastic Search

$q_1$
- 1000
- 2000
- 3000
- 4000

$q_{2,3}$

$q_4$
- 1232
- 1233
- 2311
- 2413
- 3142
- 4233
- 4333
Greedy Local Search

- usually runs on complete instantiations (leaves)
- starts in a randomly chosen instantiation
- assignments aren't necessarily consistent

Progressing:
- Local changes (of one variable assignment)
- *Greedy*, minimizing cost function (#broken constraints)

Stopping Criterion:
- Assignment is consistent (const function = 0)
**Greedy SLS: Algorithm**

**procedure:** SLS  

**Input:** A constraint network \( \mathcal{R} = (X, D, C) \). A cost function defined on full assignments.  

**Output:** A solution (no guarantee to terminate)  

**initialization:** let \( \bar{a} = (a_1, \ldots, a_n) \) be a random initial assignment to all variables.  

**while \( \bar{a} \) is not consistent do**  

- let \( Y = (x_i, a'_i) \) be the set of variable-value pairs that when \( x_i \) is assigned \( a'_i \), give a maximum improvement in the cost of the assignment  
- pick a pair \( x_i, a'_i \in Y \).  
- \( \bar{a} \leftarrow (a_1, \ldots a_{i-1}, a'_i, a_{i+1}, \ldots a_n) \) (just flip \( a_i \) to \( a'_i \))  

end  

**return** \( \bar{a} \)
Example

4-queens with SLS:

➢ starts in a randomly chosen instantiation
➢ random change of one assignment
➢ minimize #broken constraints
➢ stop when cost function = 0

Cost function value: 6
Example

4-queens with SLS:

- starts in a randomly chosen instantiation
- random change of one assignment
- minimize #broken constraints
- stop when cost function = 0

<table>
<thead>
<tr>
<th></th>
<th>3</th>
<th>5</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Cost function value: **4**
Example

4-queens with SLS:

- starts in a randomly chosen instantiation
- random change of one assignment
- minimize #broken constraints
- stop when cost function = 0

Cost function value: 2
Example

4-queens with SLS:

- starts in a randomly chosen instantiation
- random change of one assignment
- minimize #broken constraints
- stop when cost function = 0

Cost function value: 1
Problem with SLS

➢ Search can get stuck in a *local minimum* or on a *plateau*

→ Algorithm never terminates

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>4</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td></td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td></td>
<td>4</td>
</tr>
</tbody>
</table>

Cost function value: 2

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td></td>
<td>3</td>
</tr>
</tbody>
</table>

Cost function value: 1
Plateaus & Local Minima

Plateau

Local Minimum

Global Minimum

1234  1244  1242  1342  1142

11 - 29
1. Plateau Search

➢ Allow non-improving sideway steps

➢ Problem: running in circles
2. Tabu search

- Store last $n$ variable-value assignments
- Use list to prevent backward moves

$q_2 : 1$
$q_2 : 3$
$q_3 : 4$
3. Random Restarts

- Restart algorithm in new random initialisation
- Can be combined with other escape-techniques
- Suggestions for restart:
  - when no improvement is possible
  - after $max_{\text{flips}}$ steps without improvement (Plateau search)
  - increase $max_{\text{flips}}$ after every improvement
- Achieve guarantee to find a solution
4. Constraint weighting

- Cost function: \( F(\vec{a}) = \sum_i w_i \cdot C_i(\vec{a}) \)

- Increasing weights of a violated constraint in local minima
Problem: Undetermined Termination

➢ Set a limit \textit{max\_tries} for the algorithm when to stop
➢ \textbf{but}: we lose guarantee to find a solution

Anytime Behaviour

➢ Store best assignment found so far (minimal \#broken constraints)
➢ Return assignment when we need one (no solution)
Random Walks

procedure: RandomWalk

Input : A network $\mathcal{R} = (X, D, C)$, probability $p$.
Output: A solution iff the problem is consistent.
start with a random initial assignment $\bar{a}$.
while $\bar{a}$ is not a solution do
    (i) pick a violated constraint $C$, randomly
    (ii) choose with probability $p$ a variable-value pair $\langle x, a' \rangle$ for $x \in \text{scope}(C)$, or, with probability $1 - p$, choose a variable-value pair $\langle x, a' \rangle$ that minimizes the value of the cost function when the value of $x$ is changed to $a'$.
    (iii) Change $x$’s value to $a'$.
end
return $\bar{a}$.

Eventually hits a satisfying assignment (if exists)
p and Simulated Annealing

- Optimal p values for specific problems

Extension: **Simulated Annealing**

- Decrease p over time (by „cooling the temperature“)
  - more random jumps in earlier stages
  - more greedy progress later
SLS + Inference

Goal: Smaller search space

➢ use Inference methods as with systematic search

➢ constraint propagation: performance varies
  ➢ very helpful for removing many near-solutions
  ➢ not good for uniform problem structures
Recap: Cycle-cutset decomposition
SLS with Cycle-Cutset

Idea: Replace systematic search on cutset with SLS

➢ Start with random cutset assignment

Repeat:

➢ calculate minimal cost in trees:

\[
C(z_i \rightarrow a_i) = \sum_{\text{children } z_j} \min_{a_j \in D_{z_j}} (C(z_j \rightarrow a_j) + R(z_i \rightarrow a_i, z_j \rightarrow a_j))
\]

➢ assign values with minimal cost to tree variables

➢ greedily optimize cutset assignment (Local Search)
Example: Binary domains

1. Assign values to cutset variables
SLS with Cycle-Cutset

Set a **Root** for each tree

Random init.

= 1
SLS with Cycle-Cutset

2. From leaves to root:

Calculate minimal cost values

\[ C(z_i \rightarrow a_i) = \sum_{\text{children } z_j} \min_{a_j \in D_{z_j}} (C(z_j \rightarrow a_j) + R(z_i \rightarrow a_i, z_j \rightarrow a_j)) \]
3. From root to leaves:

Assign values with minimal cost
SLS with Cycle-Cutset

1. Assign values to cutset variables
2. From leaves to root:

Calculate minimal cost values

\[ C(z_i \rightarrow a_i) = \sum_{\text{children } z_j} \min_{a_j \in D_{z_j}} (C(z_j \rightarrow a_j) + R(z_i \rightarrow a_i, z_j \rightarrow a_j)) \]
SLS with Cycle-Cutset

3. From root to leaves:

Assign values with minimal cost
Summary

Stochastic Local Search

- Approximates systematic search
- Greedy algorithms: Techniques to escape local minima
- Random Walk: combines greedy + random choices
- Combination with Inference methods can help

- Can work very well
- but no guarantee of termination AND finding a solution