Problem Solving and Search in Artificial Intelligence

Local Search, Stochastic Hill Climbing, Simulated Annealing

Nysret Musliu
Database and Artificial Intelligence Group
Institut für Informationssysteme, TU-Wien
Local Search

1. Pick a solution from the search space and evaluate its merit. Define this as current solution.
2. Apply a transformation to the current solution to generate a new solution and evaluate its merit.
3. If the new solution is better than the current solution then exchange it with the current solution.
4. Repeat steps 2 and three until no transformation in the given set improves the current solution.
Local search for SAT

GSAT algorithm is based on flip of variable that results in the largest decrease number of unsatisfied clauses

Procedure GSAT
begin
for i=1 step 1 until MAX-TRIES do
begin
    T<- a randomly generated truth assignment
    for j=1 step 1 until MAX-FLIPS do
        if T satisfies the formula then return(T)
        else make a flip of variable in T that results in the largest decrease in the number of unsatisfied clauses
    end
    return(“no satisfying assignment found”)
end
Local Search and TSP

- One of simplest algorithm is 2-opt algorithm
  - Start with the random permutation of the cities (call this tour T)
  - Tries to improve T based in its neighbourhood
  - Neighbourhood of T is defined as the set of all tours that can be reached by changing two nonadjacent edges in T
  - Move is called 2-interchange
Local Search and TSP

2-interchange move

Diagram showing before and after of a 2-interchange move in a graph.
2-Opt Algorithm

- A new tour T' after the 2-interchange move replaces T if it is better.
- If none of the tours in the neighborhood is better than the tour T, the algorithm terminates.
- The algorithm should be started from several random permutations.
- 2-opt algorithm can be extended to k-opt algorithm.
Lin-Kernighan Algorithm

- Refines the k-opt strategy by allowing k to vary from one iteration to another
- It favors the largest improvement in neighbourhood, not the first improvement like in k-opt
- Generates near optimal solutions for TSPs with up to million cities
- Needs under one hour on a modern workstation
Greedy Algorithms

- Simple algorithms
- Assigns the values for all decisions variables one by one and at every step makes the best available decision
- Heuristic provides the best possible move at each step
- Do not always return the optimum solution
Greedy Algorithm for the SAT

- Possible greedy heuristic for SAT
  - For each variable from 1 to n, in any order, assign the truth value that results in satisfying the greatest number of currently unsatisfied clauses
  - Performance of such greedy algorithm is quite poor
    - For example:
      \[
      \overline{x_1} \land (x_1 \lor x_2) \land (x_1 \lor x_4)
      \]
Greedy Algorithm for the SAT

- Possible improve of previous greedy algorithm
  - Sort all variables on basis of their frequency, from the smallest to the largest
  - For each variable in order, assign a value that would satisfy the greatest number of currently unsatisfied clauses
- Further improves to the greedy algorithm can be done
- There is no good greedy algorithm for the SAT
Greedy Algorithm for the TSP

- Nearest neighbourhood heuristic
  - Start from random city
  - Proceed to the nearest unvisited city
  - Continue with step 2 until every city has been visited
- The tour with this algorithm can be far from perfect
Greedy Algorithm for the TSP

- For example with this heuristic if we start from A the following tour will be generated: A-B-C-D-A (cost=33)
- There exist much better tour A-C-B-D-A (cost=19)
Local search

- Ease of implementation
- Guarantee of local optimality usually in small computational time
- No need for exact model of the problem
- Poor quality of solution due to getting stuck in poor local optima
Modern Heuristics (Metaheuristics)

- These algorithms guide an underlying heuristic/local search to escape from being trapped in a local optima and to explore better areas of the solution space.

Examples:
  - Single solution approaches: Simulated Annealing, Tabu Search, etc.
  - Population based approaches: Genetic algorithm, Memetic algorithm, ACO, etc.

+ Able to cope with inaccuracies of data and model, large sizes of the problem and real-time problem solving

+ Including mechanisms to escape from local optima of their embedded local search algorithms

+ Ease of implementation

+ No need for exact model of the problem

- Usually no guarantee of optimality
Elements of Local Search

- Representation of the solution
- Evaluation function
- Neighbourhood function: to define solutions which can be considered close to a given solution. For example:
  - For optimisation of real-valued functions in elementary calculus, for a current solution $x_0$, neighbourhood is defined as an interval $(x_0 - r, x_0 + r)$
  - In clustering problem, all the solutions which can be derived from a given solution by moving one customer from one cluster to another
Elements of Local Search

- The larger the neighbourhood, the harder it is to explore and the better the quality of its local optimum
- Finding an efficient neighbourhood:
  - balance between the quality of the solution and the complexity of the search
- Neighbourhood search strategy
  - random
  - systematic search
- Acceptance criterion:
  - first improvement
  - best improvement
  - best of non-improving solutions
  - random criteria
Hill Climbing Algorithm

1. Pick a random point in the search space
2. Consider all the neighbours of the current state
3. Choose the neighbour with the best quality and move to that state
4. Repeat 2 through 4 until all the neighbouring states are of lower quality
5. Return the current state as the solution state
The Problem with Hill Climbing

- Gets stuck at local minima
- Possible solutions:
  - Try several runs, starting at different positions
  - Increase the size of the neighbourhood (e.g. in TSP try 3-opt rather than 2-opt)

- Stochastic Hill-Climbing
  - Only one solution from neighbourhood is selected
  - This solution will be accepted for the next iteration with some probability, which depends from the difference between current solution and selected solution
Procedure stochastic hill-climber
begin
  t=0
  select a current string \( v_c \) at random
  evaluate \( v_c \)
repeat
  select the string \( v_n \) from the neighborhood of \( v_c \)
  select \( v_n \) with probability
  \[
  \frac{1}{1 + e^{\frac{\text{eval}(v_c) - \text{eval}(v_n)}{T}}}
  \]
  \( t = t+1 \)
until \( t = \text{MAX} \)
end
Stochastic Hill Climbing

- The neighborhood of a current solution $v_c$ consists from only one solution $v_n$
- The probability of acceptance of the solution $v_n$ depends on:
  - Difference in merit between $v_c$ and $v_n$
  - Parameter $T$

$$p = \frac{1}{1 + e^{\frac{\text{eval}(v_c) - \text{eval}(v_n)}{T}}}$$

- $T$ remains constant during the execution of the algorithm
Role of parameter $T$

- Example:
  - eval($v_c$) = 107, eval($v_n$) = 120
  - maximization problem

$$ p = \frac{1}{1 + e^{-13/T}} $$

<table>
<thead>
<tr>
<th>$T$</th>
<th>$p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.00</td>
</tr>
<tr>
<td>5</td>
<td>0.93</td>
</tr>
<tr>
<td>10</td>
<td>0.78</td>
</tr>
<tr>
<td>20</td>
<td>0.66</td>
</tr>
<tr>
<td>50</td>
<td>0.56</td>
</tr>
<tr>
<td>$10^{10}$</td>
<td>0.5...</td>
</tr>
</tbody>
</table>
Role of parameter $T$

- **Example:**
  - $\text{eval}(v_c)=107$, $\text{eval}(v_n)=120$
  - Maximization problem

$$ p = \frac{1}{1 + e^{-\frac{13}{T}}} $$

<table>
<thead>
<tr>
<th>$T$</th>
<th>$p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.00</td>
</tr>
<tr>
<td>5</td>
<td>0.93</td>
</tr>
<tr>
<td>10</td>
<td>0.78</td>
</tr>
<tr>
<td>20</td>
<td>0.66</td>
</tr>
<tr>
<td>50</td>
<td>0.56</td>
</tr>
<tr>
<td>$10^{10}$</td>
<td>0.5...</td>
</tr>
</tbody>
</table>

The greater the parameter $T$, the smaller the importance of the relative merit of the competing points $v_c$ and $v_n$. 
If $T$ is huge -> search becomes random
T is very small -> stochastic hill-climber reverts into ordinary hill climber
Example:
- \( \text{eval}(v_c) = 107, \ T = 10 \)

<table>
<thead>
<tr>
<th>eval( (v_n) )</th>
<th>eval( (v_c) - \text{eval}(v_n) )</th>
<th>( p )</th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>27</td>
<td>0.06</td>
</tr>
<tr>
<td>100</td>
<td>7</td>
<td>0.33</td>
</tr>
<tr>
<td>107</td>
<td>0</td>
<td>0.50</td>
</tr>
<tr>
<td>120</td>
<td>-13</td>
<td>0.78</td>
</tr>
<tr>
<td>150</td>
<td>-43</td>
<td>0.99</td>
</tr>
</tbody>
</table>
Example:
- eval(v_c)=107, T=10

<table>
<thead>
<tr>
<th>eval(v_n)</th>
<th>eval(v_c)-eval(v_n)</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>27</td>
<td>0.06</td>
</tr>
<tr>
<td>100</td>
<td>7</td>
<td>0.33</td>
</tr>
<tr>
<td>107</td>
<td>0</td>
<td>0.50</td>
</tr>
<tr>
<td>120</td>
<td>-13</td>
<td>0.78</td>
</tr>
<tr>
<td>150</td>
<td>-43</td>
<td>0.99</td>
</tr>
</tbody>
</table>

If eval(v_c)=eval(v_n), the probability of acceptance is 0.5.
Simulated Annealing

- Changes the parameter $T$ during the search
- Starts with high value for $T$ – random search
- The value of $T$ gradually decreases
- To the end $T$ is very small and the SA behaves like an ordinary Hill-climber
Simulated Annealing

- Is based on the analogy from the thermodynamics
- To grow a crystal, the raw material is heated to a molten state
- The temperature of the crystal melt is reduced until the crystal structure is frozen in
- Cooling should not be done too quickly, otherwise some irregularities are locked in the crystal structure
Simulated Annealing

Procedure simulated annealing

begin
  \( t = 0 \)
  Initialize \( T \)
  Select a current string \( v_c \) at random
  Evaluate \( v_c \)
  repeat
    repeat
      Select a new point \( v_n \) in the neighborhood of \( v_c \)
      \[ \text{if} \quad \text{eval}(v_c) < \text{eval}(v_n) \quad \text{then} \quad v_c = v_n \]
      \[ \text{else if} \quad \text{random}[0,1] < e^{\frac{\text{eval}(v_n) - \text{eval}(v_c)}{T}} \quad \text{then} \quad v_c = v_n \]
    until (termination-condition)
    \( T = g(T, t) \)
    \( t = t + 1 \)
  until (halting-criterion)
end
SA – problem specific questions

- What is a solution?
- What are the neighbors of a solution?
- What is a cost of a solution?
- How do we determine the initial solution?
SA – specific questions

- How do we determine the initial “temperature” $T$?
- How do we determine the cooling ration $g(T,t)$?
- How do we determine the termination condition?
- How do we determine the halting criterion?
STEP 1: $T=T_{\text{max}}$

select $v_c$ at random

STEP 2: pick a point $v_n$ from the neighborhood of $v_c$

if eval($v_n$) is better than the val($v_c$)
then select it ($v_c=v_n$)
else select it with probability $e^{-\Delta \text{eval}}_T$

repeat this step $k_T$ times

STEP 3: set $T=rT$

if $T \geq T_{\text{min}}$
then goto STEP 2
else goto STEP 1
Simulated Annealing for SAT problem

Procedure SA-SAT
begin
  tries=0
  repeat
    v <- random truth assignment
    j=0
    repeat
      If v satisfies the clauses then return v
      \[ T = T_{\text{max}} e^{-jr} \]
    for k=1 to the number of variables do
      begin
        compute the increase (decreases) \( \delta \) in the number of clauses made true if \( v_k \) was flipped
        flip variable \( v_k \) with the probability \( \frac{\delta}{1 + e^{T}} \)
        v <- new assignment if the flip is made
      end
      j=j+1
    until \( T \leq T_{\text{min}} \)
    tries=tries+1
  until tries=MAX-TRIES
end
SA for SAT

- $r$ represents a decay rate for the temperature
- Spears (1996) used
  - $T_{\text{max}} = 0.03$ and $T_{\text{min}} = 0.01$
  - $r$ depend on the number of variables and number of tries
- SA-SAT appeared to satisfy at least as many formulas as GSAT, with less work
- Advantage of SA-SAT came from its backward moves
Other application of SA

- Traveling Salesman Problem
- VLSI design
- Production scheduling
- Timetabling problems
- Image processing
- ...
References

- Z. Michalewicz and D. B. Fogel. How to Solve It: Modern Heuristics
  - Chapters 3 (sec. 3.2), 4 (sec. 4.1), 5 (sec. 5.1)

- Other papers
  - Simulated annealing for hard satisfiability problems: W.M. Spears
  - Optimization by Simulated Annealing: An Experimental Evaluation; Part I, Graph Partitioning
    DS Johnson, CR Aragon, LA McGeoch, C Schevon