Stochastic greedy local search

Chapter 7

ICS-275
Spring 2010
Example: 8-queen problem
Main elements

- Choose a full assignment and iteratively improve it towards a solution
- Requires a cost function: number of unsatisfied constraints or clauses. Neural networks use energy minimization
- Drawback: local minimas
- Remedy: introduce a random element
- Cannot decide inconsistency
Algorithm Stochastic Local search (SLS)

Procedure SLS

Input: A constraint network $\mathcal{R} = (X, D, C)$, number of tries $\text{MAX TRIES}$. A cost function.

Output: A solution iff the problem is consistent, ”false” otherwise.

1. for $i=1$ to $\text{MAX TRIES}$
   
   • initialization: let $\bar{a} = (a_1, \ldots, a_n)$ be a random initial assignment to all variables.
   
   • repeat
     
     (a) if $\bar{a}$ is consistent, return $\bar{a}$ as a solution.
     
     (b) else 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 $a'_i$).
   
   • until the current assignment cannot be improved.

2. endfor

3. return false
Example: CNF

Example 7.1 Consider the formula $\varphi = \{(\neg C)(\neg A \lor \neg B \lor C)(\neg A \lor D \lor E)(\neg B \lor \neg C)\}$. Assume that in the initial assignment all variables are assigned the value "1". This assignment violates two clauses, the first and the last, so the cost is 2. Next we see that flipping A, E or D will not remove any inconsistency. Flipping C to "0" will satisfy the two violated clauses but will violate the clause $(\neg A \lor \neg B \lor C)$, yielding a cost of 1. Flipping B to $\neg B$ will remove one inconsistency and has a cost of 1 as well. If we flip C to $\neg C$, and subsequently flipping B to $\neg B$ yields a cost of 0 – and a solution.

- Example: $z$ divides $y, x, t$ $z = \{2,3,5\}$, $x, y = \{2,3,4\}$, $t = \{2,5,6\}$
Heuristics for improving local search

- **Plateau search**: at local minima continue search sideways.
- **Constraint weighting**: use weighted cost function
  - The cost $C_i$ is 1 if no violation. At local minima increase the weights of violating constraints.
  \[ F(\tilde{a}) = \sum w_i C_i(\tilde{a}) \]
- **Tabu search**: prevent backwards moves by keeping a list of assigned variable-values. Tie-breaking rule may be conditioned on historic information: select the value that was flipped least recently
- **Automating Max-flips**: based on experimenting with a class of problems
  - Given a progress in the cost function, allow the same number of flips used up to current progress.
Random walk strategies

- Combine random walk with greediness
  - At each step:
    - choose randomly an unsatisfied clause.
    - with probability $p$ flip a random variable in the clause, with $(1-p)$ do a greedy step minimizing the breakout value: the number of new constraints that are unsatisfied.
Figure 7.2: Algorithm WalkSAT

Procedure WalkSAT

Input: A network $\mathcal{R} = (X, D, C)$, number of flips MAX_FLIPS, MAX_TRIES, probability $p$.

Output: True iff the problem is consistent, false otherwise.

1. For i= 1 to MAX_TRIES do

2. Compare best assignment with $\bar{a}$ and retain the best.

   a) start with a random initial assignment $\bar{a}$.

   b) for i=1 to MAX_FLIPS

      a. if $\bar{a}$ is a solution, return true and $\bar{a}$.
      b. else,
         i. pick a violated constraint $C$, randomly
         ii. choose with probability $p$ a variable-value pair $< x, a' >$ for
             $x \in scope(C)$, or, with probability $1-p$, choose a variable-value
             pair $< x, a' >$ that minimizes the number of new constraints
             that break when the value of $x$ is changed to $a'$, (minus 1 if the
             current constraint is satisfied).
         iii. Change $x$’s value to $a'$.

3. endfor

4. return false and the best current assignment.
Example of walkSAT:
start with assignment of true to all vars

Example 7.2 Following our earlier example 7.1.1, we will first select an unsatisfied clause, such as $(\neg B \lor \neg C)$, and then select a variable. If we try to minimize the number of additional constraints that would be broken, we will select $B$ and flip its value. Subsequently, the only unsatisfied clause is $\neg C$ which is selected and flipped. 

$$(\neg C), (\neg A \lor \neg B \lor C)(\neg A \lor D \lor E)(\neg B \lor \neg C)$$
Simulated Annealing (Kirkpatrick, Gellat and Vecchi (1983))

- Pick randomly a variable and a value and compute delta: the change in the cost function when the variable is flipped to the value.
- If change improves execute it,
- Otherwise it is executed with probability $e^{-\frac{\delta}{T}}$ where T is a temperature parameter.
- The algorithm will converge if T is reduced gradually.
Properties of local search

- Guarantee to terminate at local minima
- Random walk on 2-sat is guaranteed to converge with probability 1 after \(N^2\) steps, when \(N\) is the number of variables.

Proof:
- A random assignment is on the average \(N/2\) flips away from a satisfying assignment.
- There is at least \(\frac{1}{2}\) chance that a flip of a 2-clause will reduce the distance to a given satisfying assignment by 1.
- Random walk will cover this distance in \(N^2\) steps on the average.

- Analysis breaks for 3-SAT
- Empirical evaluation shows good performance compared with complete algorithms (see chapter and numerous papers)
Hybrids of local search and Inference

- We can use exact hybrids of search+inference and replace search by SLS (Kask and Dechter 1996)
  - Good when cutset is small
- The effect of preprocessing by constraint propagation on SLS (Kask and Dechter 1995)
  - Great improvement on structured problems
  - Not so much on uniform problems
SLS and Local Consistency

- **Structured** (hierarchical 3SAT cluster structures) vs. **(uniform) random**.

**Basic scheme:**
- Apply preprocessing (resolution, path consistency)
- Run SLS
- Compare against SLS alone
SLS and Local Consistency

A 3SAT clause over variables $x$, $y$ and $z$
SLS and Local Consistency

<table>
<thead>
<tr>
<th>C/cluster</th>
<th>Before Resolution</th>
<th>After Resolution</th>
<th>DP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Solved</td>
<td>Time</td>
<td>Flips</td>
</tr>
<tr>
<td>30</td>
<td>100</td>
<td>0.52 sec</td>
<td>4.5K</td>
</tr>
<tr>
<td>31</td>
<td>100</td>
<td>0.71</td>
<td>5.1K</td>
</tr>
<tr>
<td>32</td>
<td>100</td>
<td>1.03</td>
<td>8.4K</td>
</tr>
<tr>
<td>33</td>
<td>100</td>
<td>1.54</td>
<td>12K</td>
</tr>
<tr>
<td>34</td>
<td>100</td>
<td>3.44</td>
<td>26K</td>
</tr>
<tr>
<td>35</td>
<td>100</td>
<td>6.38</td>
<td>49K</td>
</tr>
<tr>
<td>36</td>
<td>90</td>
<td>21.7</td>
<td>161K</td>
</tr>
<tr>
<td>37</td>
<td>41</td>
<td>35.5</td>
<td>252K</td>
</tr>
<tr>
<td>38</td>
<td>3</td>
<td>28.4</td>
<td>202K</td>
</tr>
<tr>
<td>39</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>40</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 1: Bound-3 Resolution and GSAT

SLS and Local Consistency

| C   | Solvable | Algorithm   | Solved | Tries | Flips  | PPC Time | Total Time | BJ-DVO
<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>265</td>
<td>88.5 %</td>
<td>GSAT</td>
<td>139</td>
<td>336</td>
<td>147K</td>
<td>0 sec</td>
<td>36 sec</td>
<td>19 min</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PPC + GSAT</td>
<td>152</td>
<td>292</td>
<td>140K</td>
<td>8 sec</td>
<td>66 sec</td>
<td></td>
</tr>
<tr>
<td>270</td>
<td>66 %</td>
<td>GSAT</td>
<td>78</td>
<td>406</td>
<td>191K</td>
<td>0 sec</td>
<td>45 sec</td>
<td>33 min</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PPC + GSAT</td>
<td>83</td>
<td>381</td>
<td>195K</td>
<td>14 sec</td>
<td>92 sec</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>N=30, K=64, T=2048/4096, 100 instances, MaxFlips = 128K, Ccrit=180?</th>
</tr>
</thead>
<tbody>
<tr>
<td>163</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

Table 2: Partial Path Consistency and GSAT

<table>
<thead>
<tr>
<th>Uniform random 3SAT, N=600, C=2550, 100 instances, MaxFlips = 512K</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm</td>
</tr>
<tr>
<td>-------------------------</td>
</tr>
<tr>
<td>GSAT</td>
</tr>
<tr>
<td>BR-3 + GSAT</td>
</tr>
</tbody>
</table>

Table 3: Bound-3 Resolution and GSAT
SLS and Local Consistency

Summary:

- For structured problems, enforcing local consistency will improve SLS.
- For uniform CSPs, enforcing local consistency is not cost effective: performance of SLS is improved, but not enough to compensate for the preprocessing cost.
SLS and Cutset Conditioning

**Background:**

- Cycle cutset technique improves backtracking by conditioning only on cutset variables.

\[
X_i = \{0,1\} \\
X_j = \{0,1\}
\]
SLS and Cutset Conditioning

**Background:**
- Tree algorithm is tractable for trees.
- Networks with bounded width are tractable*.

**Basic Scheme:**
- Identify a cutset such that width is reduced to desired value.
- Use search with cutset conditioning.
Local search on Cycle-cutset

Tree variables $X$

$C_{min} = \min_{Y=y} C(y) = \min_{Y=y} \min_{X=x} \{C(X \mid Y = y)\}$
Results GSAT with Cycle-Cutset
(Kask and Dechter, 1996)
SLS and Cutset Conditioning

**Summary:**

- A new combined algorithm of SLS and inference based on cutset conditioning
- Empirical evaluation on random CSPs
- SLS combined with the tree algorithm is superior to pure SLS when the cutset is small
Possible project

- Program variants of SLS+Inference
  - Use the computed cost on the tree to guide SLS on the cutset. This is applicable to optimization
  - Replace the cost computation on the tree with simple arc-consistency or unit resolution
  - Implement the idea for SAT using off-the-shelves code: unit-resolution from minisat, SLS from walksat.

- Other projects: start thinking.
More project ideas

- Combine local search with constraint propagation.
  


- Look at REES