Local and Stochastic Search

Some material based on D Lin, B Selman
Search Overview

- Introduction to Search
- Blind Search Techniques
- Heuristic Search Techniques
- Constraint Satisfaction Problems
- Local Search (Stochastic) Algorithms
  - Motivation
  - Hill Climbing
  - Issues
  - SAT ... Phase Transition, GSAT, ...
  - Simulated Annealing, Tabu, Genetic Algorithms
- Game Playing search
A Different Approach

- So far: systematic exploration:
  - Explore full search space (possibly) using principled pruning (A*, ...)
- Best such algorithms (IDA*) can handle
  - $10^{100}$ states; ≈500 binary-valued variables (ballpark figures only!)
- but... some real-world problem have
  - 10,000 to 100,000 variables; $10^{30,000}$ states
- We need a completely different approach:
  - Local Search Methods
  - Iterative Improvement Methods
Local Search Methods

- Applicable when seeking Goal State
  ...& don't care how to get there

- E.g.,
  - *N*-queens, map coloring, VLSI layout,
    planning, scheduling, TSP, time-tableing, ...

- Many (most?) real Operations Research problems are solved using local search!
  - E.g., schedule for Delta airlines, ...
Example #1: 4 Queen

- **States**: 4 queens in 4 columns (256 states)
- **Operators**: move queen in column
- **Goal test**: no attacks
- **Evaluation**: $h(n) = \text{number of attacks}$
Example#2: Graph Coloring

1. Start with random coloring of nodes
2. Change color of one node to reduce #conflicts
Graph Coloring Example
Graph Coloring Example

<table>
<thead>
<tr>
<th>Iteration</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th># conflicts</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>b</td>
<td>g</td>
<td>g</td>
<td>r</td>
<td>b</td>
<td>r</td>
<td>2 {AE, DF}</td>
</tr>
</tbody>
</table>
### Graph Coloring Example

<table>
<thead>
<tr>
<th>Iteration</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th># conflicts</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>b</td>
<td>g</td>
<td>g</td>
<td>r</td>
<td>b</td>
<td>r</td>
<td>2 {AE, DF}</td>
</tr>
<tr>
<td>2</td>
<td>b</td>
<td>g</td>
<td>g</td>
<td>B</td>
<td>b</td>
<td>r</td>
<td>1 {AE}</td>
</tr>
</tbody>
</table>
Graph Coloring Example

<table>
<thead>
<tr>
<th>Iteration</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th># conflicts</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>b</td>
<td>g</td>
<td>g</td>
<td>r</td>
<td>b</td>
<td>r</td>
<td>2 {AE, DF}</td>
</tr>
<tr>
<td>2</td>
<td>b</td>
<td>g</td>
<td>g</td>
<td>b</td>
<td>b</td>
<td>r</td>
<td>1 {AE}</td>
</tr>
<tr>
<td>3</td>
<td>R</td>
<td>g</td>
<td>g</td>
<td>b</td>
<td>b</td>
<td>r</td>
<td>0 {}</td>
</tr>
</tbody>
</table>
“Local Search”

1. Select (random) initial state (initial guess at solution)
2. While GoalState not found (& more time)
   - Make *local modification* to improve current state

Requirements:
- Generate a random (probably-not-optimal) guess
- Evaluate quality of guess
- Move to other states (well-defined neighborhood function)
... and do these operations quickly...
function `HILL-CLIMBING(problem)` returns a solution state

inputs: problem, a problem

static: current, a node

next, a node

current ← MAKE-NODE(INITIAL-STATE[problem])

loop do

next ← a highest-valued successor of current

if VALUE[next] < VALUE[current] then return current

current ← next

end
If Continuous ....

- Situation
  - State = \langle v_1, \ldots, v_n \rangle \in \mathbb{R}^n
  - quality \ h : \mathbb{R}^n \mapsto \mathbb{R}
    \quad h(\text{state}) \in \mathbb{R}

- To find optimum:
  
  Guess random initial state \( \vec{v}^0 \in \mathbb{R}^n \)
  
  While \( \exists i \ \frac{\partial h(X)}{\partial X_i} \bigg|_{X=\vec{v}} \neq 0 \) do
    
    For \( i = 1..n \)
      
      \( \vec{v}_i := \vec{v}_i - \eta \frac{\partial h}{\partial X_i} \bigg|_{X=\vec{v}} \)
  
  Return \( \vec{v} \)

  - May have other termination conditions
  - If \( \eta \) too small: very slow
  - If \( \eta \) too large: overshoot
  - May have to approximate derivatives from samples
<table>
<thead>
<tr>
<th>Iteration</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th># conflicts</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>r</td>
<td>g</td>
<td>b</td>
<td>r</td>
<td>b</td>
<td>b</td>
<td>3 {CE, CF, EF}</td>
</tr>
</tbody>
</table>
But ...  

- Pure “Hill Climbing” will not work!
- Need “Plateau Walk”

<table>
<thead>
<tr>
<th>Iteration</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th># conflicts</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>r</td>
<td>g</td>
<td>b</td>
<td>r</td>
<td>b</td>
<td>b</td>
<td>3 {CE, CF, EF}</td>
</tr>
<tr>
<td>2</td>
<td>r</td>
<td>g</td>
<td>G</td>
<td>r</td>
<td>b</td>
<td>b</td>
<td>1 {EF}</td>
</tr>
<tr>
<td>Iteration</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td># conflicts</td>
</tr>
<tr>
<td>-----------</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>------------------</td>
</tr>
<tr>
<td>1</td>
<td>r</td>
<td>g</td>
<td>b</td>
<td>r</td>
<td>b</td>
<td>b</td>
<td>3 {CE, CF, EF}</td>
</tr>
<tr>
<td>2</td>
<td>r</td>
<td>g</td>
<td>G</td>
<td>r</td>
<td>b</td>
<td>b</td>
<td>1 {EF}</td>
</tr>
<tr>
<td>3</td>
<td>r</td>
<td>g</td>
<td>g</td>
<td>r</td>
<td>b</td>
<td>R</td>
<td>1 {DF}</td>
</tr>
</tbody>
</table>
But ...

<table>
<thead>
<tr>
<th>Iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
</tbody>
</table>
Problems with Hill Climbing

- Pure “Hill Climbing” does not always work!
- Often need “Plateau Walk”
- Sometimes: Climb DOWN-HILL!

... trying to find the top of Mount Everest in a thick fog while suffering from amnesia ...
Problems with Hill Climbing

- **Foothills / Local Optimal:** No neighbor is better, but not at global optimum
  - Maze: may have to move AWAY from goal to find best solution
- **Plateaus:** All neighbors look the same.
  - 8-puzzle: perhaps no action will change # of tiles out of place
- **Ridge:** going up only in a narrow direction.
  - Suppose no change going South, or going East, but big win going SE
- **Ignorance of the peak:** Am I done?
Goal is to find GLOBAL optimum.

1. How to avoid LOCAL optima?
2. How long to *plateau walk*?
3. When to stop?
4. Climb down hill? When?
Local Search Example: SAT

- Many real-world problems \( \approx \) propositional logic
  \((A \lor B \lor C) \land (\neg B \lor C \lor D) \land (A \lor \neg C \lor D)\)
- Solved by finding truth assignment to
  \((A, B, C, \ldots)\) that satisfy formula
- Applications
  - planning and scheduling
  - circuit diagnosis and synthesis
  - deductive reasoning
  - software testing
  - ...
Obvious Algorithm

\[(A \lor C) \land (\neg A \lor C) \land (B \lor \neg C) \land (A \lor \neg B)\]
Satisfiability Testing

*Davis-Putnam Procedure* (1960)

- Backtracking depth-first search (DFS) through space of truth assignments (+ unit-propagation)
  - *fastest* sound + complete method
    - ... best-known systematic method ...
  - ... but ...
    - ∃ classes of formulae where it scales badly...
Greedy Local Search

- Why not just HILL-CLIMB??

- Given
  - formula: \( \varphi = (A \lor C) \land (\neg A \lor C) \land (B \lor \neg C) \)
  - assignment: \( \sigma = \{ \neg a, \neg b, +c \} \)
  - \( \text{Score}(\varphi, \sigma) = \# \text{clauses unsatisfied} \ldots = 0 \)
  - Just flip variable that helps most!

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>((A \lor C) \land (\neg A \lor C) \land (B \lor \neg C))</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>x (\land) + (\land) +</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>+</td>
<td>+ (\land) + (\land) x</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>+</td>
<td>+</td>
<td>+ (\land) + (\land) +</td>
<td>0</td>
</tr>
</tbody>
</table>
Greedy Local Search: GSAT

1. Guess random truth assignment
2. Flip value assigned to the variable that yields the greatest # of satisfied clauses. (Note: Flip even if no improvement)
3. Repeat until all clauses satisfied, or have performed “enough” flips
4. If no sat-assign found, repeat entire process, starting from a new initial random assgmt

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>((A \lor C) \land (\neg A \lor C) \land (B \lor \neg C))</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>(x) + +</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>+</td>
<td>+ + (x)</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>+</td>
<td>+</td>
<td>+ + +</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>+</td>
<td>+</td>
<td>+ + +</td>
<td>0</td>
</tr>
</tbody>
</table>
Does GSAT Work?

- First intuition:
  GSAT will get stuck in local minima, with a few unsatisfied clauses.

- Very bad...
  “almost satisfying assignments” are worthless (Eg, plan with one “magic" step is useless)
  ...ie, NOT optimization problem

- Surprise: GSAT often found global minimum!
  Ie, satisfying assignment!
  10,000+ variables; 1,000,000+ constraints!

- No good theoretical explanation yet...
# GSAT vs. DP on Hard Random Instances

<table>
<thead>
<tr>
<th>form. vars</th>
<th>m. flips</th>
<th>GSAT retries</th>
<th>time</th>
<th>Davis-Putnam choices</th>
<th>depth</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>250</td>
<td>6</td>
<td>0.5 sec</td>
<td>77</td>
<td>11</td>
<td>1 sec</td>
</tr>
<tr>
<td>70</td>
<td>350</td>
<td>11</td>
<td>1 sec</td>
<td>42</td>
<td>15</td>
<td>15 sec</td>
</tr>
<tr>
<td>100</td>
<td>500</td>
<td>42</td>
<td>6 sec</td>
<td>$10^3$</td>
<td>19</td>
<td>3 min</td>
</tr>
<tr>
<td>120</td>
<td>600</td>
<td>82</td>
<td>14 sec</td>
<td>$10^5$</td>
<td>22</td>
<td>18 min</td>
</tr>
<tr>
<td>140</td>
<td>700</td>
<td>53</td>
<td>14 sec</td>
<td>$10^6$</td>
<td>27</td>
<td>5 hrs</td>
</tr>
<tr>
<td>150</td>
<td>1500</td>
<td>100</td>
<td>45 sec</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>200</td>
<td>2000</td>
<td>248</td>
<td>3 min</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>300</td>
<td>6000</td>
<td>232</td>
<td>12 min</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>500</td>
<td>10000</td>
<td>996</td>
<td>2 hrs</td>
<td>$10^{30}$</td>
<td>$&gt;100$</td>
<td>$10^{19}$ yrs</td>
</tr>
</tbody>
</table>

**Notes:** Define “Hard” later
Only “satisfiable” formulae
(else GSAT does not terminate)
Systematic vs. Stochastic

- **Systematic search:**
  - DP systematically checks all possible assignments
  - Can determine if the formula is unsatisfiable

- **Stochastic search:**
  - Once we find it, we're done!
  - Guided random search approach
  - Can't determine unsatisfiability
What Makes a SAT Problem Hard?

- Randomly generate formula $\varphi$ with
  - $n$ variables; $m$ clauses with $k$ variables each

- $\#\text{possible_clauses} = \binom{n}{k} \times 2^k$

- Will $\varphi$ be satisfied??
  - If $n << m$: ??
  - If $n >> m$: ??
Phase Transition

For 3-SAT

- $m/n < 4.2$, under constrained $\Rightarrow$ nearly all formulae sat.
- $m/n > 4.3$, over constrained $\Rightarrow$ nearly all formulae unsat.
- $m/n \sim 4.26$, critically constrained $\Rightarrow$ need to search.
Under-constrained problems are easy: just guess an assignment

Over-constrained problems are easy: just say “unsatisfiable”
(... often easy to verify using Davis-Putnam)

At $m/n \approx 4.26$, there exists a phase transition between these two different types of easy problems.

This transition sharpens as $n$ increases.

For large $n$, hard problems are extremely rare (in some sense)
Hard problems are at Phase Transition!!
Improvements to Basic Local Search

Issues:
- How to move more quickly to successively better plateaus?
- Avoid “getting stuck” / local minima?

Idea: Introduce uphill moves (“noise”) to escape from plateaus/local minima

Noise strategies:
1. Simulated Annealing
   - Kirkpatrick et al. 1982; Metropolis et al. 1953
2. Mixed Random Walk
   - Selman and Kautz 1993
Simulated Annealing

Pick a random variable
If flip improves assignment: do it.
Else flip with probability $p = e^{-\delta/T}$ (go the wrong way)

- $\delta =$ #of additional clauses becoming unsatisfied
- $T =$ “temperature”
  - Higher temperature = greater chance of wrong-way move
  - Slowly decrease $T$ from high temperature to near 0
- Q: What is $p$ as $T$ tends to infinity?
  - ... as $T$ tends to 0?
  - For $\delta = 0$?
Simulated Annealing Algorithm

current, next: nodes/states
T: “temperature” controlling prob. of downward steps
schedule: mapping from time to “temperature”
h: heuristic evaluation function

current \rightarrow \text{initial state}
for \ t \leftarrow 1..\infty \ do
    T \leftarrow \text{schedule}[t]
    \text{if } T = 0 \text{ then return } current
    next \leftarrow \text{randomly selected successor of } current
    \Delta E \leftarrow h(next) - h(current)
    \text{if } \Delta E > 0 \text{ then } current \leftarrow next
    \text{else } current \leftarrow next \text{ only with probability } e^{\Delta E / T}
Notes on SA

- Noise model based on statistical mechanics
  - Introduced as analogue to physical process of growing crystals
    - Kirkpatrick et al. 1982; Metropolis et al. 1953

- Convergence:
  1. W/ exponential schedule, will converge to global optimum
  2. No more-precise convergence rate
    (Recent work on rapidly mixing Markov chains)

- Key aspect: **upwards / sideways** moves
  - Expensive, but (if have enough time) can be best

- Hundreds of papers/ year;
  - Many applications: VLSI layout, factory scheduling, ...
Pure WalkSat

PureWalkSat( formula )

Guess initial assignment

While (unsatisfied) do

Select unsatisfied clause \( c = \pm X_i \lor \pm X_j \lor \pm X_k \)

Select variable \( v \) in unsatisfied clause \( c \)

Flip \( v \)
Example:

Eg: \((A \lor B) \& (\neg A \lor C) \& (\neg B \lor \neg D) \& \ldots\)

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>+</td>
<td></td>
</tr>
</tbody>
</table>

Clause \((A \lor B)\) not satisfied.
so flip either \(A\) or \(B\)... say \(A\)

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>+</td>
<td>0</td>
<td>0</td>
<td>+</td>
<td></td>
</tr>
</tbody>
</table>

\((A \lor B)\) now satisfied.
... but \((\neg A \lor C)\) is now NOT satisfied!
Mixing Random Walk with Greedy Local Search

MixedWalkSat\(_p\)( formula )
    Guess initial assignment
    While unsatisfied do
      W/ prob \( p \), walk
        (flip var in an unsatisfied clause)
      W/ prob \( 1 - p \), greedy
        (flip var producing fewest unsatisfied clauses)

- Usual issues:
  - Termination conditions
  - Multiple restarts

- Determine value of \( p \) empirically
  ... finding best setting for problem class
Finding the best value of $p$

- Let
  - $Q[p, c]$ be quality of using $\text{WalkSat}[p]$ on problem $c$

\[
Q[p, c] = \text{Time to return answer, or}
\]
\[
= 1 \text{ if WalkSat}[p] \text{ returns (correct) answer within 5mins and 0 otherwise, or}
\]
\[
= \ldots \text{ perhaps some combination of both } ...
\]

- $Q[p] = \sum_{c \in S} Q[p, c]$

- Set $p^* = \arg\max_p QQ[p]$
Experimental Results: Hard Random 3CNF

- Time in seconds (SGI Challenge)
- Effectiveness: prob. that random initial assignment leads to a solution
- Complete methods, such as DP, up to 400 variables
- Mixed Walk ... better than Simulated Annealing
  - better than Basic GSAT
  - better than Davis-Putnam

<table>
<thead>
<tr>
<th>vars</th>
<th>GSAT basic</th>
<th>GSAT walk</th>
<th>Simul. Ann.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>time</td>
<td>eff.</td>
<td>time</td>
</tr>
<tr>
<td>100</td>
<td>.4</td>
<td>.12</td>
<td>.2</td>
</tr>
<tr>
<td>200</td>
<td>22</td>
<td>.01</td>
<td>4</td>
</tr>
<tr>
<td>400</td>
<td>122</td>
<td>.02</td>
<td>7</td>
</tr>
<tr>
<td>600</td>
<td>1471</td>
<td>.01</td>
<td>35</td>
</tr>
<tr>
<td>800</td>
<td>*</td>
<td>*</td>
<td>286</td>
</tr>
<tr>
<td>1000</td>
<td>*</td>
<td>*</td>
<td>1095</td>
</tr>
<tr>
<td>2000</td>
<td>*</td>
<td>*</td>
<td>3255</td>
</tr>
</tbody>
</table>
Overcoming Local Optima and Plateaus

✓ Simulated annealing
✓ Mixed-in random walk
✓ Random restarts
✓ Tabu search
✓ Genetic alg/programming
✓ …
Random Restarts

- Restart at new random state after pre-defined # of local steps.
- Useful with “Heavy Tail” distribution
- Done by GSAT
Tabu Search

- Avoid returning quickly to same state
- Implementation:
  - Keep fixed length queue (tabu list)
  - Add most recent step to queue; drop oldest step
  - Never make step that's on current tabu list
- Example:
  - without tabu:
  - with tabu (length 4):
- Tabu very powerful;
  - competitive w/ simulated annealing or random walk (depending on the domain)
Genetic Algorithms

- Class of probabilistic optimization algorithms
  - A genetic algorithm maintains a population of candidate solutions for the problem at hand, and makes it evolve by iteratively applying a set of stochastic operators
- Inspired by the biological evolution process
- Uses concepts of “Natural Selection” and “Genetic Inheritance” (Darwin 1859)
- [John Holland, 1975]
Examples: Recipe

To find optimal quantity of three major ingredients (sugar, wine, sesame oil)

- Use an alphabet of 1-9 denoting ounces
- Solutions might be
  - 1-1-1
  - 2-1-4
  - 3-3-1
  - ...

Standard Genetic Algorithm

- Randomly generate an initial population
- For $i=1..N$
  - Select parents and "reproduce" the next generation
  - Evaluate fitness of the new generation
  - Replace some of the old generation with the new generation
Stochastic Operators

- **Cross-over**
  - decomposes two distinct solutions
  - then randomly mixes their parts to form novel solutions

- **Mutation**
  - randomly perturbs a candidate solution
Genetic Algorithm Operators
Mutation and Crossover

Parent 1
1 0 1 0 1 1 1

Parent 2
1 1 0 0 0 1 1

Child 1
1 0 1 0 0 1 1

Child 2
1 1 0 0 1 1 0

Mutation
Examples

- **Mutation:**
  In recipe example, 1-2-3 may be changed to
  - 1-3-3 or
  - 3-2-3

- **Parameters to adjust**
  - How often?
  - How many digits change?
  - How big?
More examples:

- Crossover

  In recipe example:
  - Parents 1-3-3 & 3-2-3
    Crossover point after the first digit
  - Generate two offspring: 3-3-3 and 1-2-3

Can have one or two point crossover
Local Search Summary

- Surprisingly efficient search technique
- Wide range of applications
- Formal properties elusive
- Intuitive explanation:
  - Search spaces are too large for systematic search anyway... 
- Area will most likely continue to thrive