Stochastic Local Search Variants

Computer Science cpsc322, Lecture 16

(Textbook Chpt 4.8)

June, 1, 2017
Lecture Overview

- Recap SLS
- SLS variants
Announcements

• Assignment 2 has been posted on Connect due June 8
• Midterm on June 8 – first block of class
  • Search
  • CSP
  • SLS
  • Planning
  • Possibly simple/minimal intro to logics
Stochastic Local Search

- **Key Idea**: combine greedily improving moves with randomization

- As well as improving steps, we can allow a “small probability” of:
  - **Random steps**: move to a random neighbor.
  - **Random restart**: reassign random values to all variables.

- Always keep **best solution found so far**

- Stop when:
  + Solution is found (in vanilla CSP)
  + Run out of time (return best solution so far)
Lecture Overview

- Recap SLS
- SLS variants
  - Tabu lists
  - Simulated Annealing
  - Beam search
  - Genetic Algorithms
Tabu lists

- To avoid search to
  - Immediately going back to previously visited candidate
  - To prevent cycling

- Maintain a tabu list of the $k$ last nodes visited.
  - Don’t visit a poss. world that is already on the tabu list.

- Cost of this method depends on $\_\_\_\_k$
Simulated Annealing

- Key idea: Change the degree of randomness.

- Annealing: a metallurgical process where metals are hardened by being slowly cooled.
  - Analogy: start with a high "temperature": a high tendency to take random steps
  - Over time, cool down: more likely to follow the scoring function

- Temperature reduces over time, according to an annealing schedule
Simulated Annealing: algorithm

Here’s how it works (for maximizing):

- You are in node $n$. Pick a variable at random and a new value at random. You generate $n'$.
- If it is an improvement i.e., $h(n') \geq h(n)$, adopt it.
- If it isn’t an improvement, adopt it probabilistically depending on the difference and a temperature parameter, $T$.
  
  $\begin{cases} h(n') < h(n) \land h(n') - h(n) < 0 \rightarrow \text{we move to } n' \text{ with probability } e^{(h(n') - h(n))/T} \end{cases}$

see next slide
• If it isn’t an improvement, adopt it probabilistically depending on the difference and a temperature parameter, $T$.

  - we move to $n'$ with probability $e^{(h(n')-h(n))/T}$

- The higher $T$, the higher is prob for a given difference.
- The higher the difference, the smaller is prob for a given $T$. 

\[
\left| h(n') - h(n) \right|
\]
Properties of simulated annealing search

One can prove: If $T$ decreases slowly enough, then simulated annealing search will find a \textit{global optimum} with probability approaching 1.

Widely used in \textit{VLSI layout}, \textit{airline scheduling}, etc.

Finding the ideal cooling schedule is unique to each class of problems.
Lecture Overview

- Recap SLS
- SLS variants
  - Simulated Annealing
  - Population Based
    ✓ Beam search
    ✓ Genetic Algorithms
Population Based SLS

Often we have more memory than the one required for current node (+ best so far + tabu list)

Key Idea: maintain a population of $k$ individuals
- At every stage, update your population.
- Whenever one individual is a solution, report it.

Simplest strategy: Parallel Search
- All searches are independent
- No information shared

but more memory!

no reasons to use it!
Population Based SLS: Beam Search

Non Stochastic

- Like parallel search, with $k$ individuals, but you choose the $k$ best out of all of the neighbors.
- Useful information is passed among the $k$ parallel search thread

**Troublesome case:** If one individual generates several good neighbors and the other $k-1$ all generate bad successors, the next generation will comprise very similar individuals.
Population Based SLS: Stochastic Beam Search

- **Non Stochastic** Beam Search may suffer from lack of diversity among the k individual (just a more expensive hill climbing).

- **Stochastic** version alleviates this problem:
  - Selects the k individuals at random
  - But probability of selection proportional to their value (according to scoring function)

\[
P_{\text{selecting } n_j} = \frac{n_j}{\sum_i n_i}
\]

\[
\frac{\sum_i h(n_i)}{h(n_j)}
\]
Stochastic Beam Search: Advantages

• It maintains diversity in the population.
• Biological metaphor (asexual reproduction):
  ✓ each individual generates “mutated” copies of itself (its neighbors)
  ✓ The scoring function value reflects the fitness of the individual
  ✓ the higher the fitness the more likely the individual will survive (i.e., the neighbor will be in the next generation)
Lecture Overview

- Recap SLS
- SLS variants
  - Simulated Annealing
  - Population Based
    - Beam search
    - Genetic Algorithms
Population Based SLS: Genetic Algorithms

- Start with $k$ randomly generated individuals (population)

- An individual is represented as a string over a finite alphabet (often a string of 0s and 1s)

- A successor is generated by combining two parent individuals (loosely analogous to how DNA is spliced in sexual reproduction)

- Evaluation/Scoring function (fitness function). Higher values for better individuals.

- Produce the next generation of individuals by selection, crossover, and mutation
**Genetic algorithms: Example**

**Representation and fitness function**

State: string over finite alphabet

Fitness function: higher value \# queen pairs not attacking each other

8-queen

# of queen pairs possibly attacking each other

\[
\frac{8 \times 7}{2} = 28
\]

28 - 4

24

32752411

28 - 5

23

24748552
Genetic algorithms: Example

**Selection:** common strategy, probability of being chosen for reproduction is directly proportional to fitness score

\[
\text{24}/(24+23+20+11) = 31\%
\]

\[
\text{23}/(24+23+20+11) = 29\%
\]

\[
\text{20}/(24+23+20+11) = 26\%
\]

\[
\text{11}/(24+23+20+11) = 14\%
\]

Same as Beam Search slide 14
Genetic algorithms: Example

Reproduction: cross-over and mutation
Genetic Algorithms: Conclusions

- Their performance is very sensitive to the choice of state representation and fitness function
- **Extremely slow** (not surprising as they are inspired by evolution!)
Sampling a discrete probability distribution

e.g. Sim. Annealing. Select n' with probability P

\[ P = 0.3 \]

Generate random number in \([0, 1]\)

If \(< 0.3\) accept n'

e.g. Beam Search: Select K individuals. Probability of selection proportional to their value

\[ \begin{align*}
\rightarrow n_1 & \quad P_1 = 0.1 \\
\rightarrow n_2 & \quad P_2 = 0.3 \\
\rightarrow n_3 & \quad P_3 = 0.2 \\
\rightarrow n_4 & \quad P_4 = 0.4
\end{align*} \]

SAME HERE

CPSC 322, Lecture 16
Learning Goals for today’s class

You can:

• Implement a tabu-list.
• Implement the simulated annealing algorithm
• Implement population based SLS algorithms:
  • Beam Search
  • Genetic Algorithms.
• Explain pros and cons of different SLS algorithms.
Modules we’ll cover in this course: R&Rsys

**Problem**
- Static
  - Constraint Satisfaction
- Query
- Sequential
  - Planning

**Environment**
- Deterministic
  - Arc Consistency
  - Search
  - SLS
- Stochastic
  - Belief Nets
    - Var. Elimination
  - Decision Nets
    - Var. Elimination
  - Markov Processes
    - Value Iteration

**Representation**
- Reasoning Technique
Next class

How to select and organize a sequence of actions to achieve a given goal...

Start Planning (Chp 8.1–8.2 *Skip 8.1.1–2*)