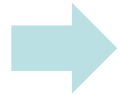# Stochastic Local Search Algorithms

CPSC 322 – CSP 7

Textbook §4.8

February 11, 2011

Lecture Overview

Announcements

- Recap: stochastic local search (SLS)

- Types of SLS algorithms

- Algorithm configuration

- AI in the news: IBM Watson

Announcements

- AIspace is being improved
  - Developers would like to track usage to focus their efforts
  - Please use this link from now on: http://www.aispace.org/cs322/

- Final exam is scheduled: Apr 11, 3:30 pm
  - First day of exams!
  - Stay on the ball

- Reminder: midterm is on Monday Feb 28
  - one week after reading break

- Assignment 2 is due Wednesday after reading break
  - It's probably the biggest of the 4 assignments
    - 2 programming questions
  - Don't leave it to the last minute
    - Can only use 2 late days

Practice exercises

- Who has used them?

- Do you know that there are solutions?
    - In WebCT, right after you submit your answer
        - General Feedback: <solution>
        - Score: 0/0

- Would you prefer PDF versions for the practice exercises?

Lecture Overview

- Announcements
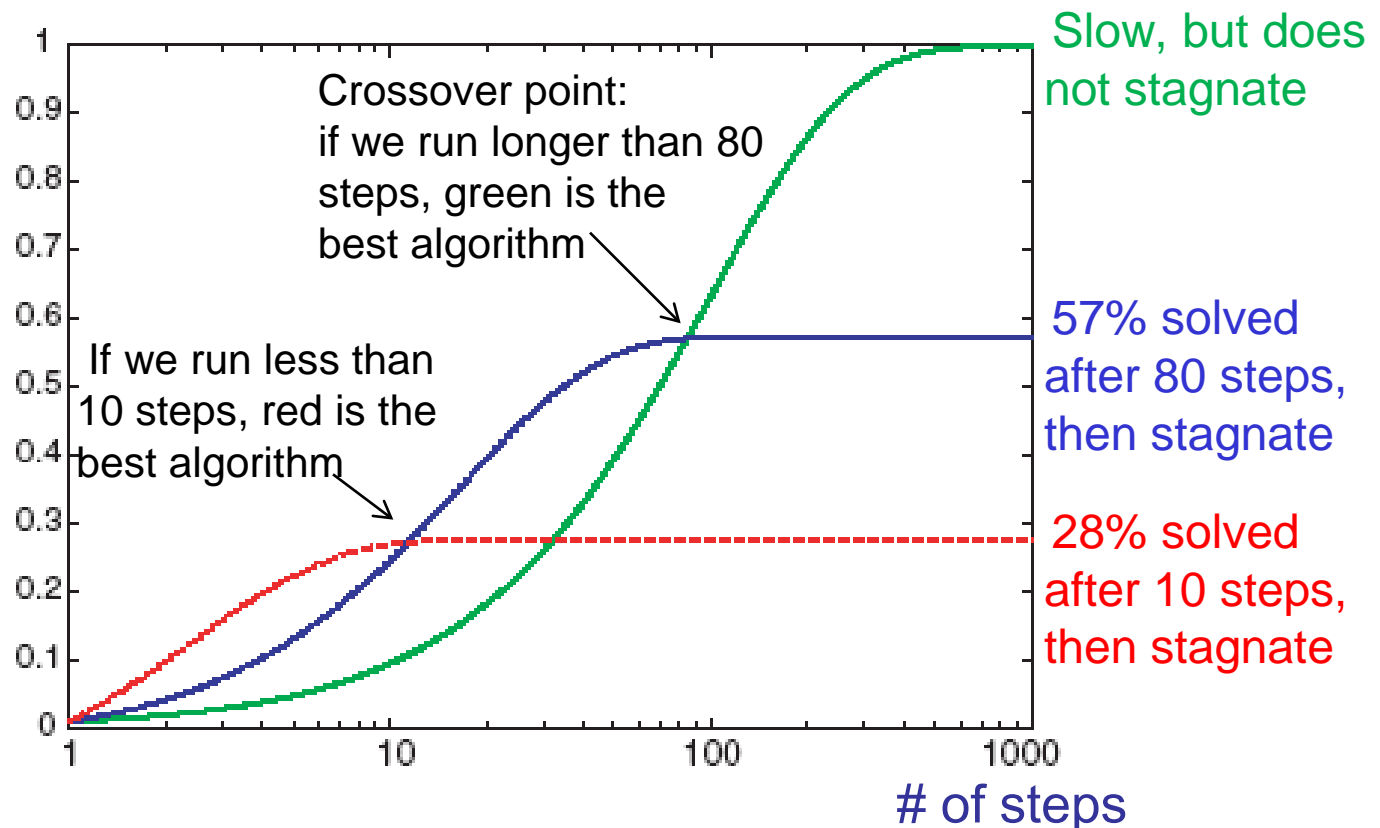
- Recap: stochastic local search (SLS)

- Types of SLS algorithms

- Algorithm configuration

- AI in the news: IBM Watson

Comparing runtime distributions

- SLS algorithms are randomized
  - The time taken until they solve a problem is a random variable

- Runtime distributions
  - x axis: runtime (or number of steps, typically log scale)
  - y axis: proportion (or number) of runs solved in that runtime

Fraction of solved runs, i.e.

P(solved by this time)

Crossover point: if we run longer than 80 steps, green is the best algorithm

If we run less than 10 steps, red is the best algorithm

Slow, but does not stagnate

57% solved after 80 steps, then stagnate

28% solved after 10 steps, then stagnate

# of steps

Pros and Cons of SLS

- Typically no guarantee to find a solution even if one exists
  - Most SLS algorithms can sometimes stagnate
    - Not clear whether problem is infeasible or the algorithm stagnates
    - Very hard to analyze theoretically
  - Some exceptions: guaranteed to find global minimum as time $\rightarrow \infty$
    - In particular random sampling and random walk:
      strictly positive probability of making N lucky choices in a row

- Anytime algorithms
  - maintain the node with best h found so far (the "incumbent")
  - given more time, can improve their incumbent

- Generality: can optimize arbitrary functions with n inputs
  - Example: constraint optimization
  - Example: RNA secondary structure design

- Generality: dynamically changing problems

Lecture Overview

- Announcements

- Recap: stochastic local search (SLS)

  Types of SLS algorithms

- Algorithm configuration

- AI in the news: IBM Watson

Many different types of local search

- There are many different SLS algorithms
  - Each could easily be a lecture by itself
  - We will only touch on each of them very briefly
  - Only need to know them on a high level
  - You will have to choose and implement one of them for the programming assignment "SLS for scheduling"

- For more details, see
  - UBC CS grad course "Stochastic Local Search" by Holger Hoos
  - Book "Stochastic Local Search: Foundations and Applications" by Holger H. Hoos & Thomas Stützle, 2004 (in reading room)

Simulated Annealing

- **Annealing**: a metallurgical process where metals are hardened by being slowly cooled

- Analogy:
  - start with a high "temperature": high tendency to take random steps
  - Over time, cool down: only take random steps that are not too bad

- Details:
  - At node n, select a random neighbour n'
  - If $h(n') < h(n)$, move to n'   (i.e. accept all improving steps)
  - Otherwise, adopt it with a probability depending on
    - How much worse n' is then n
    - the current temperature T: high T tends to accept even very bad moves
    - Probability of accepting worsening move: $\exp(\ (h(n) - h(n') / T\ )$
  - Temperature reduces over time, according to an annealing schedule
    - "Finding a good annealing schedule is an art"
    - E.g. geometric cooling: every step multiply T by some constant < 1

Tabu Search

- Mark partial assignments as tabu (taboo)
  - Prevents repeatedly visiting the same (or similar) local minima

  - Maintain a queue of k Variable=value assignments that are taboo
  - E.g., when changing V_7's value from 2 to 4, we cannot change V_7 back to 2 for the next k steps

  - k is a parameter that needs to be optimized empirically

Iterated Local Search

- Perform iterative best improvement to get to local minimum
- Perform perturbation step to get to different parts of the search space
  - E.g. a series of random steps
  - Or a short tabu search
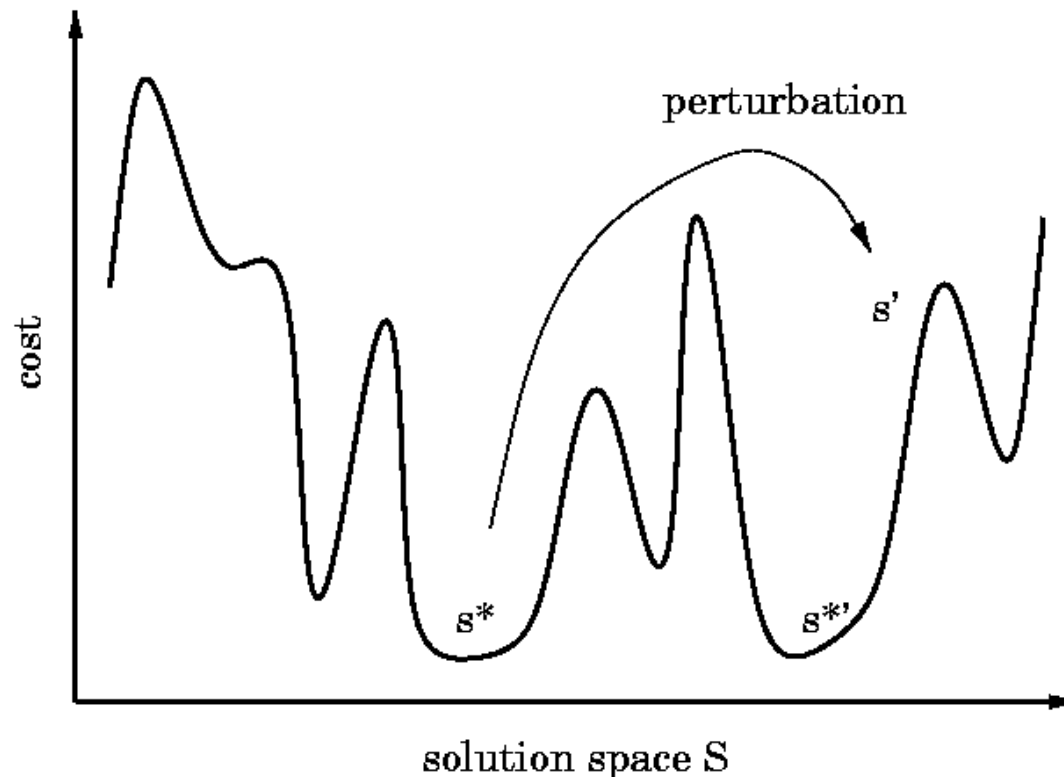
Beam Search

- Keep not only 1 assignment, but k assignments at once
    - A "beam" with k different assignments (k is the "beam width")

- The neighbourhood is the union of the k neighbourhoods
    - At each step, keep only the k best neighbours
    - Never backtrack

- When k=1, this is identical to:

Greedy descent   Breadth first search   Best first search

    - Single node, always move to best neighbour: greedy descent

- When k=∞, this is basically:

Greedy descent   Breadth first search   Best first search

    - At step k, the beam contains all nodes k steps away from the start node
    - Like breadth first search,
      but expanding a whole level of the search tree at once

- The value of k lets us limit space and parallelism

Stochastic Beam Search

- Like beam search, but you probabilistically choose the k nodes at the next step ("generation")

- The probability that neighbour n is chosen depends on h(n)
  - Neighbours with low h(n) are chosen more frequently
  - E.g. rank-based: node n with lowest h(n) has highest probability
    - probability only depends on the order, not the exact differences in h
  - This maintains diversity amongst the nodes

- Biological metaphor:
  - like asexual reproduction:
    each node gives its mutations and the fittest ones survive

Genetic Algorithms

- Like stochastic beam search, but pairs of nodes are combined to create the offspring

- For each generation:
  - Choose pairs of nodes n_1 and n_2 ("parents"), where nodes with low h(n) are more likely to be chosen
  - For each pair (n_1, n_2), perform a cross-over: create offspring combining parts of their parents
  - Mutate some values for each offspring
  - Select from previous population and all offspring which nodes to keep in the population

Example for Crossover Operator

- Given two nodes:

  $X_1 = a_1,\ X_2 = a_2,\ \ldots,\ X_m = a_m$
  $X_1 = b_1;\ X_2 = b_2,\ \ldots,\ X_m = b_m$

- Select i at random, form two offspring:

  $X_1 = a_1,\ X_2 = a_2,\ \ldots, X_i = a_i,\ X_{i+1} = b_{i+1},\ \ldots,\ X_m = b_m$
  $X_1 = b_1,\ X_2 = b_2,\ \ldots, X_i = b_i,\ X_{i+1} = a_{i+1},\ \ldots,\ X_m = a_m$

- Many different crossover operators are possible

- Genetic algorithms is a large research field
  – Appealing biological metaphor
  – Several conferences are devoted to the topic

Lecture Overview

- Announcements

- Recap: stochastic local search (SLS)

- Types of SLS algorithms

  Algorithm configuration

- AI in the news: IBM Watson

Parameters in stochastic local search

- ## Simple SLS
  - Neighbourhoods, variable and value selection heuristics, percentages of random steps, restart probability

- ## Tabu Search
  - Tabu length (or interval for randomized tabu length)

- ## Iterated Local Search
  - Perturbation types, acceptance criteria

- ## Genetic algorithms
  - Population size, mating scheme, cross-over operator, mutation rate

- ## Hybridizations of algorithms: many more parameters

The Algorithm Configuration Problem

Definition

- Given:
  - Runnable algorithm A, its parameters and their domains
  - Benchmark set of instances B
  - Performance metric m
- Find:
  - Parameter setting ("configuration") of A optimizing m on B

My PhD thesis topic (Hutter, 2009): Automated configuration of algorithms for solving hard computational problems

Motivation for automated algorithm configuration

Customize versatile algorithms
for different application domains

- Fully automated
  - Saves valuable human time
  - Can improve performance dramatically

Solver
config 1

Solver
config 2

Generality of Algorithm Configuration

Arbitrary problems, e.g.

– SAT, MIP, Timetabling, Probabilistic Reasoning, Protein Folding, AI Planning, etc

Arbitrary parameterized algorithms, e.g.

– Local search
  - Neighbourhoods, restarts, perturbation types, tabu length, etc
– Genetic algorithms & evolutionary strategies
  - Population size, mating scheme, crossover operators, mutation rate, hybridizations, etc
– Systematic tree search
  (advanced versions of arc consistency + domain splitting)
  - Branching heuristics, no-good learning, restart strategy, pre-processing, etc

Simple Manual Approach for Configuration

Start with some configuration

repeat

> *Modify a single parameter*
>
> **if** *results on benchmark set improve* **then**
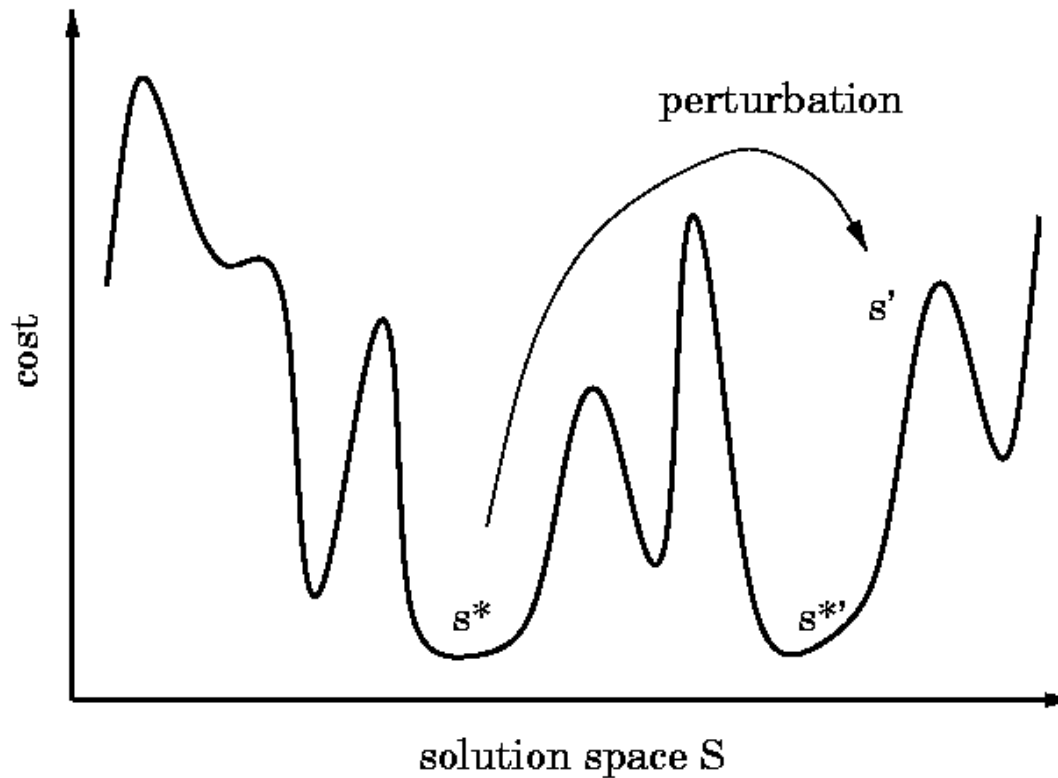>
> > *keep new configuration*

until *no more improvement possible (or "good enough")*

$\rightarrow$ *Manually executed local search*

The ParamILS Framework

Iterated Local Search in parameter configuration space:

$\rightarrow$ Perfoms biased random walk over local optima

Example application for ParamILS: solver for mixed integer programming (MIP)

MIP: NP-hard constraint optimization problem

$$\min \quad c^\top x$$
$$\text{s. t.} \quad Ax \leq b$$
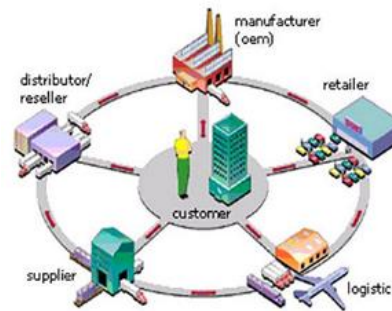$$x_i \in \mathbb{Z} \text{ for i} \in I$$

Commercial state-of-the-art MIP solver IBM ILOG CPLEX:

– licensed by > 1 000 universities and 1 300 corporations, including ⅓ of the Global 500

Transportation/Logistics:
SNCF, United Airlines, UPS, United States Postal Service, …

Supply chain management software:
Oracle, SAP, …

Production planning and optimization:
Airbus, Dell, Porsche, Thyssen Krupp, Toyota, Nissan, …

Up to 50-fold speedups just by optimizing the parameters !!

Learning Goals for local search (started)

- Implement local search for a CSP.
  - Implement different ways to generate neighbors
  - Implement scoring functions to solve a CSP by local search through either greedy descent or hill-climbing.
- Implement SLS with
  - random steps (1-step, 2-step versions)
  - random restart
- Compare SLS algorithms with runtime distributions

- Coming up
  - Assignment #2 is due Wednesday, Feb 23rd
  - Midterm is Monday, Feb 28th
  - After reading break: planning.
    - Only Sections 8.0-8.2 & 8.4

Lecture Overview

- Announcements

- Recap: stochastic local search (SLS)

- Types of SLS algorithms

- Algorithm configuration

   AI in the news: IBM Watson

IBM's Watson

- Automated AI system participating in real *Jeopardy!*
  - Won practice round against two all-time Jeopardy champions
  - 3-day match on air February 14-16 (Monday-Wednesday)
    - Jeopardy on CBC 7:30-8pm every weekday (same as US version??)

- Jeopardy website with videos: http://www.jeopardy.com/minisites/watson/

- NYTimes article: "What Is I.B.M.'s Watson?
  http://www.nytimes.com/2010/06/20/magazine/20Computer-t.html?_r=2&ref=opinion

- Wired magazine:
  "IBM's Watson Supercomputer Wins Practice Jeopardy Round"
  http://www.wired.com/epicenter/2011/01/ibm-watson-jeopardy/#

- More technical: AI magazine
  "Building Watson: An Overview of the DeepQA Project"
  http://www.stanford.edu/class/cs124/AIMagzine-DeepQA.pdf

IBM Watson: some videos

- "IBM and the Jeopardy Challenge":
http://www.youtube.com/watch?v=FC3IryWr4c8

- "IBM's Supercomputer Beats Miles O'Brien at Jeopardy":
http://www.youtube.com/watch?v=otBeCmpEKTs

- Video of practice round:
http://www.engadget.com/2011/01/13/ibms-watson-supercomputer-destroys-all-humans-in-jeopardy-pract/
  – Watson won against Jeopardy champions
    Ken Jennings and Brad Rutter (by a small margin)
  – Including interview describing some of the underlying AI
    - But if you're really interested, see the AI magazine article

Watson as an intelligent agent (see lecture 1)

Knowledge Representation
Machine Learning

Mix of knowledge representations.
Machine learning to rate *confidence* from each system
Learned confidence from 10000s example questions

prior knowledge

past experiences

goals/values

observations

Agent

Reasoning +
Decision Theory

Actions

Betting strategy!

Some, fairly simple

Natural Language
Generation

Natural Language
Understanding
+
Computer Vision
Speech Recognition
+
Physiological Sensing
Mining of Interaction Logs

Environment

+
Robotics
+
Human Computer
/Robot
Interaction

State of the art NLP components
Combination and tuning of over 100 (!) approaches.