Stochastic Local Search

Computer Science cpsc322, Lecture 15

(Textbook Chpt 4.8)

February, 6, 2009

Announcements

- Thanks for the feedback, we'll discuss it on Mon
- Assignment-2 on CSP will be out tonight (programming!)

Lecture Overview

- **Recap Local Search in CSPs**
- Stochastic Local Search (SLS)
- Comparing SLS algorithms

Local Search: Summary

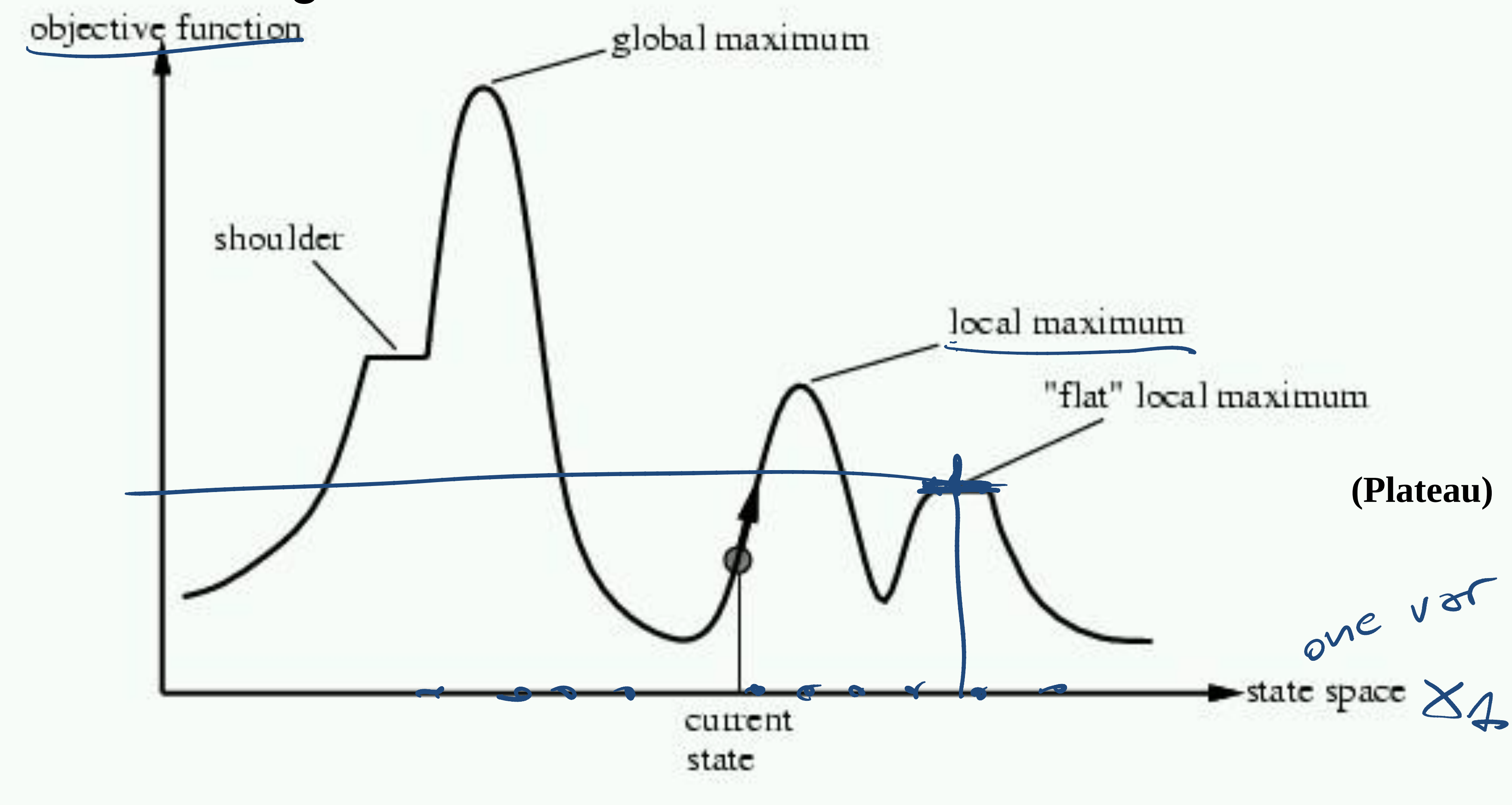
- A useful method in practice for large CSPs
  - Start from a possible world (randomly chosen)
  - Generate some neighbors ("similar" possible worlds)

    e.g. differ from current poss. world only by one variable's value
  - Move from current node to a neighbor, selected to minimize/maximize a scoring function which combines:
    - ✓ Info about how many constraints are violated
    - ✓ Information about the cost/quality of the solution (you want the best solution, not just a solution)

Problems with these strategy…

…called Greedy Descent when selecting the neighbor which minimizes a scoring function.

Hill Climbing when selecting the neighbor which maximizes a scoring function.



Lecture Overview

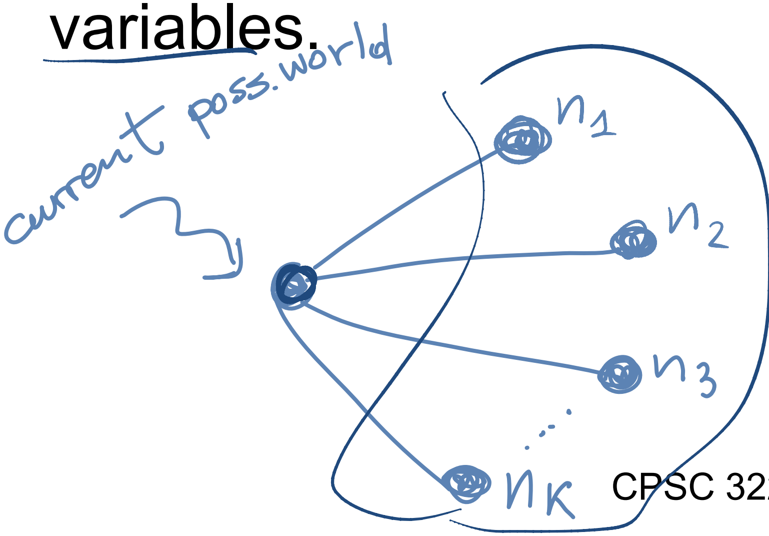
- Recap Local Search in CSPs
- Stochastic Local Search (SLS)
- Comparing SLS algorithms

Stochastic Local Search

GOAL: We want our local search

- to be guided by the scoring function
- Not to get stuck in local maxima/minima, plateaus etc.

- **SOLUTION:** We can alternate
  a) Hill-climbing steps
  b) Random steps: move to a random neighbor.
  c) Random restart: reassign random values to all variables.



→ a) move to n_i which improves scoring function

→ b) select n_i randomly

→ c) Jump to a random poss. world

Two extremes versions

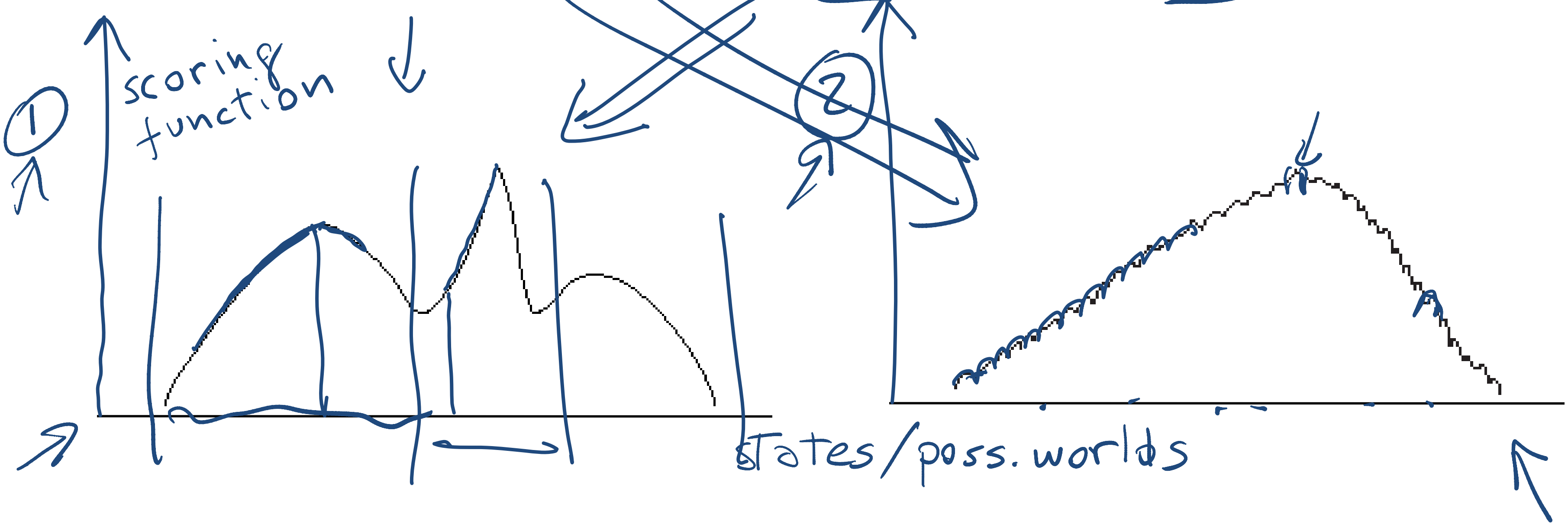
Stochastic local search typically involves both kinds of randomization, but for illustration let's consider

works better in 2

A **hill climbing with random steps**

works better in 1

B **hill climbing with random restart**



Two 1-dimensional search spaces; step right or left:

you do not know how your space will be so combine the two A & B

Random Steps (Walk)

Let's assume that neighbors are generated as

- assignments that differ in one variable's value

How many neighbors there are given n variables with domains with d values?

$n(d-1)$

One strategy to add randomness to the selection variable-value pair. Sometimes choose the pair

1. According to the scoring function
2. A random one

8 variables

E.G in 8-queen

- How many neighbors? $8 \cdot 7 = 56$
- ....

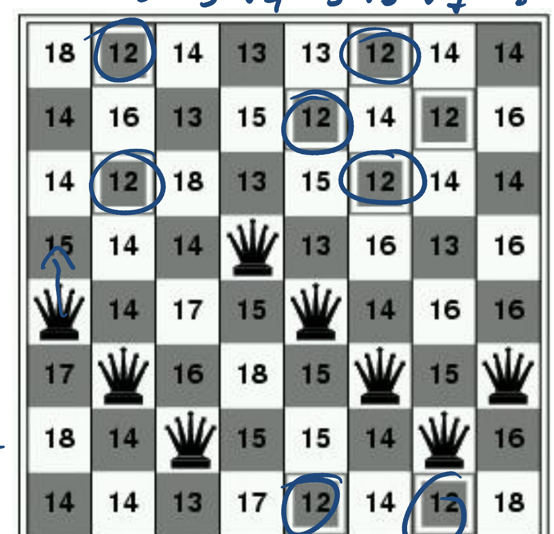
8 values

# of conflicts

1. choose one of the circled ones

2 choose randomly one of the 56

V1 V2 V3 V4 V5 V6 V7 V8



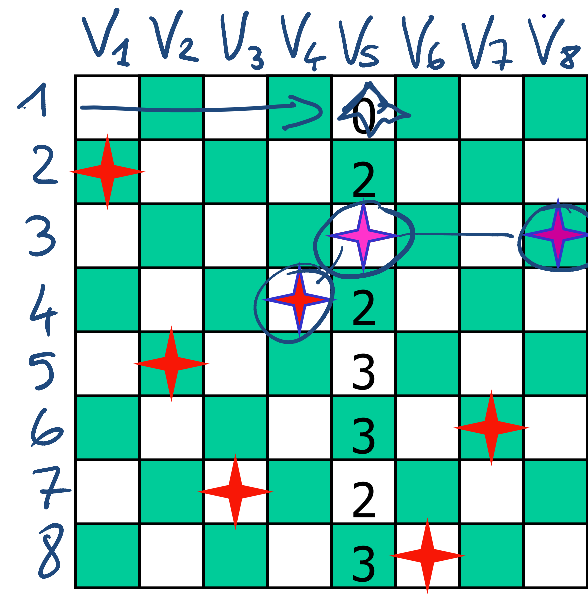
Random Steps (Walk): two-step

Another strategy: select a variable first, then a value:

- Sometimes select variable:
  1. that participates in the largest number of conflicts. V_5
  2. at random, any variable that participates in some conflict. $(V_4\ V_5\ V_8)$
  3. at random V_i
- Sometimes choose value
  - a) That minimizes # of conflicts
  - b) at random

Meth 1 selects V_5

Complete strategy 1.a) would select neighbor with $V_5 = 1$



conflicts

Aispace
2 a: Greedy Descent with
Min-Conflict Heuristic

Successful application of SLS

- **Scheduling of Hubble Space Telescope**: reducing time to schedule 3 weeks of observations:

from one week to around 10 sec.

(Stochastic) Local search advantage: Online setting

- **When the problem can change** (particularly important in scheduling)
- **E.g., schedule for airline:** thousands of flights and thousands of personnel assignment
  - Storm can render the schedule infeasible
- **Goal:** Repair with minimum number of changes
- This can be easily done with a local search starting form the current schedule
- Other techniques usually:
  - require more time
  - might find solution requiring many more changes

SLS:Limitations

- Typically no guarantee they will find a solution even if one exists

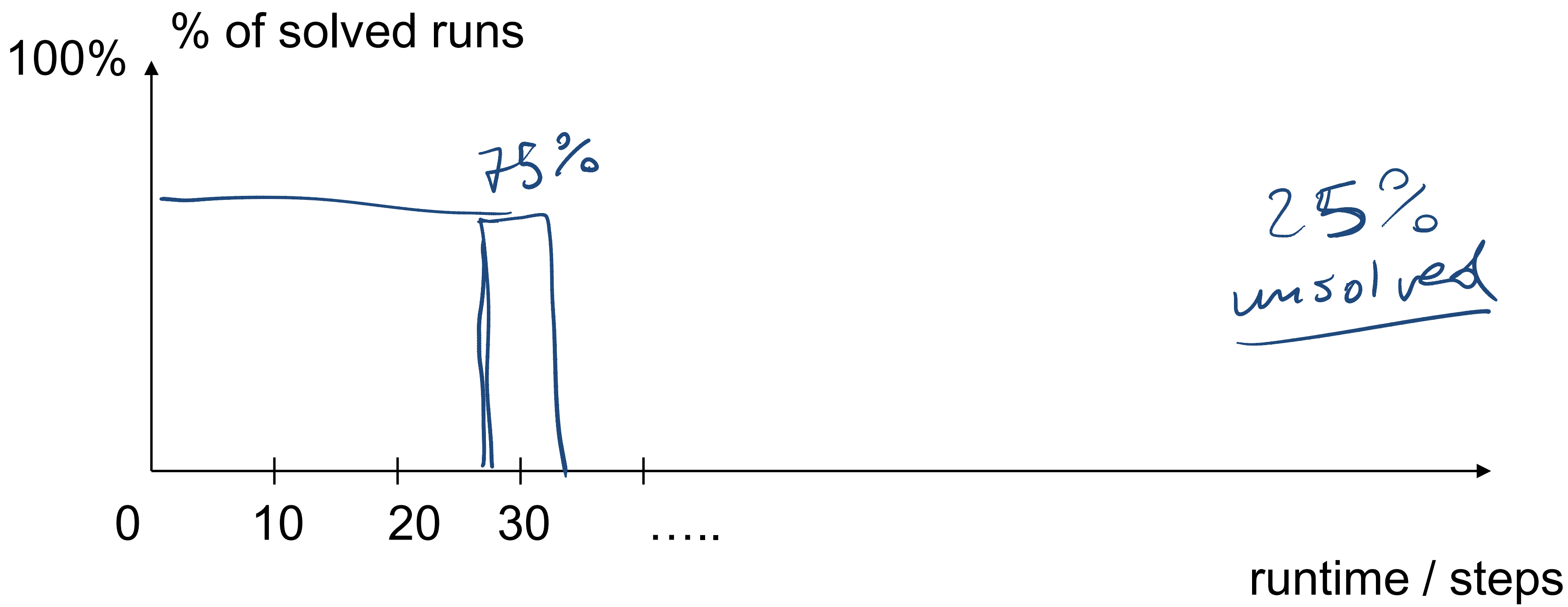
- Not able to show that no solution exists

Lecture Overview

- Recap Local Search in CSPs
- Stochastic Local Search (SLS)
- Comparing SLS algorithms

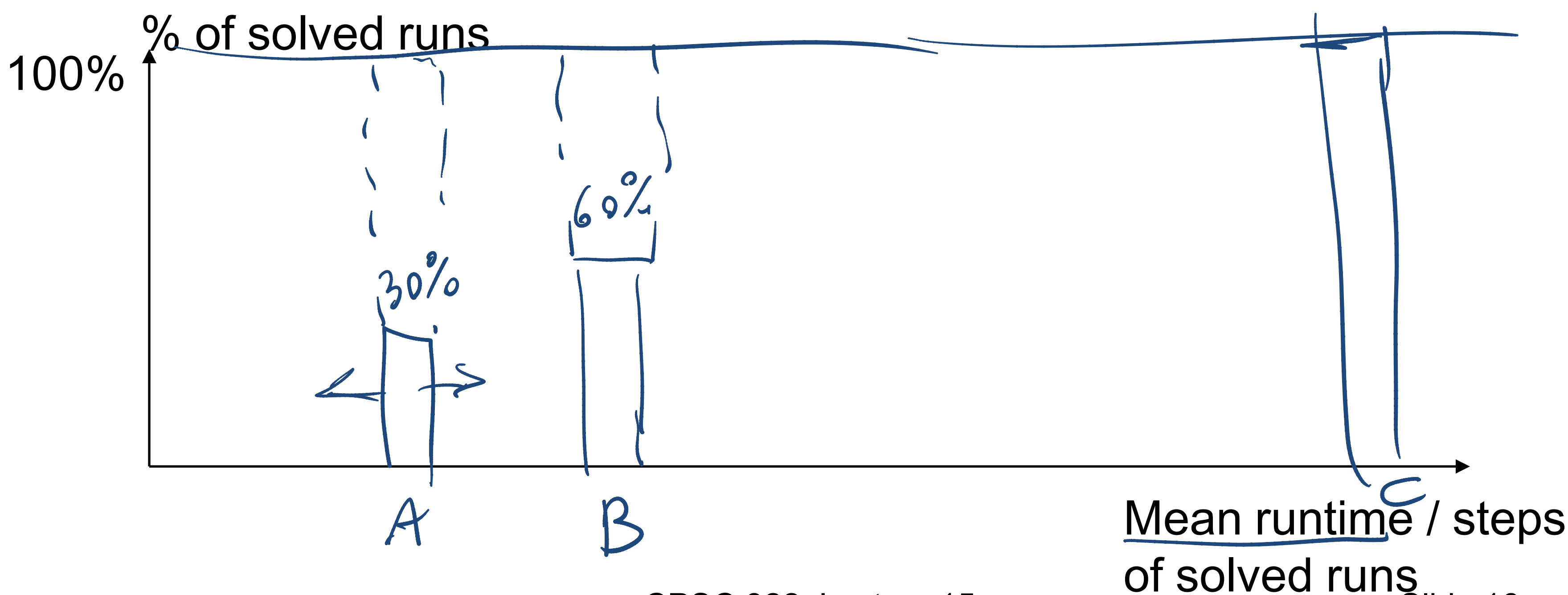
Comparing Stochastic Algorithms: Challenge

- Summary statistics, such as **mean** run time, **median** run time, and **mode** run time don't tell the whole story
  - What is the running time for the runs for which an algorithm ***never*** finishes (infinite? stopping time?)



First attempt….

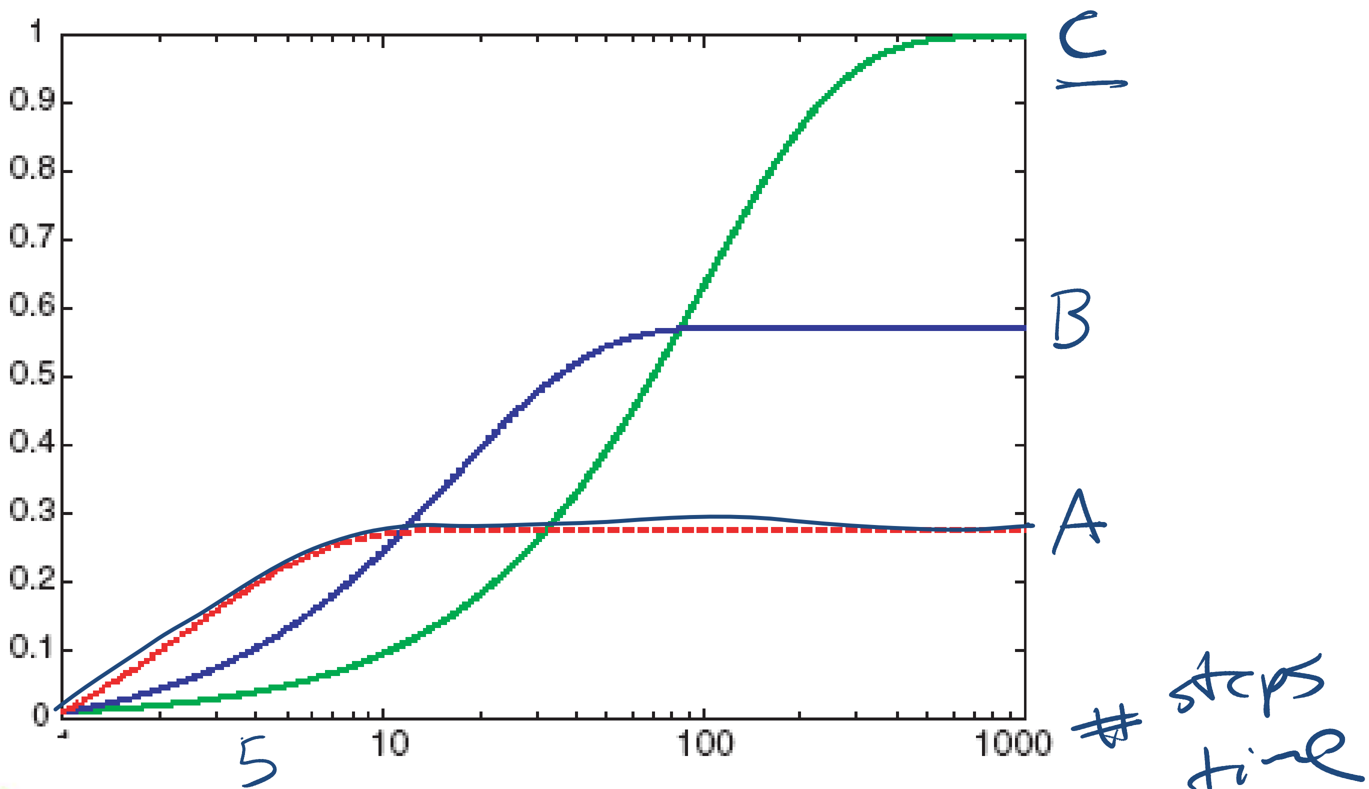
- How can you compare three algorithms when
  A. one solves the problem 30% of the time very quickly but doesn't halt for the other 70% of the cases
  B. one solves 60% of the cases reasonably quickly but doesn't solve the rest
  C. one solves the problem in 100% of the cases, but slowly?



Runtime Distributions are even more effective

Plots runtime (or number of steps) and the proportion (or number) of the runs that are solved within that runtime.

- log scale on the *x* axis is commonly used



What are we going to look at in AIspace

When selecting a variable first followed by a value:

- Sometimes select variable:
  1. that participates in the largest number of conflicts.
  2. at random, any variable that participates in some conflict.
  3. at random
- Sometimes choose value
  a) That minimizes # of conflicts
  b) at random

…..

AIspace terminology

keeps restarting

Random sampling
restart

Random walk 3b

Greedy Descent 1a

Greedy Descent Min conflict 2a

Greedy Descent with random walk 2ab

Greedy Descent with random restart

Learning Goals for today's class

You can:

- Implement SLS with
  - random steps (1-step, 2-step versions)
  - random restart
- Compare SLS algorithms with runtime distributions

Assign-2

- Will be out by tonight
- Assignments will be weighted:

A0 (12%), A1…A4 (22%) each

Next Class

- More SLS variants
- Finish CSPs
- Start planning