Introduction to

Artificial Intelligence (AI)

Computer Science cpsc502, Lecture 4

Sep, 20, 2011

Systematically solving CSPs: Summary

- Build Constraint Network
- Apply Arc Consistency
  - One domain is empty → no sol
  - Each domain has a single value → unique sol
  - Some domains have more than one value → ?!

    may or may not have a solution
- Apply Depth-First Search with Pruning
- Split the problem in a number of disjoint cases
  - Apply Arc Consistency to each case

Limitations of Systematic Approaches

- Many CSPs (scheduling, DNA computing, more later) are simply too big for systematic approaches
- If you have 10^5 vars with $\text{dom}(var_i) = 10^4$

- Systematic Search

branching

$b = 10^4$

$d = 10^5$

depth

branching factor

$(10^4)^{10^5}$

- Constraint Network

var nodes

constraint nodes

$\underline{10^5 + 10^5 * 10^5}$

10^{10} max # of nodes

var domain size

Complexity of AC

$d^3 n^2 = (10^4)^3 \cdot (10^5)^2 = 10^{22}$

# of vars

- but if solutions are densely distributed…….

Today Sept 20

Stochastic Local Search (SLS)

- Local Search & Constrained Optimization
- SLS
- SLS variants
- Comparing SLS

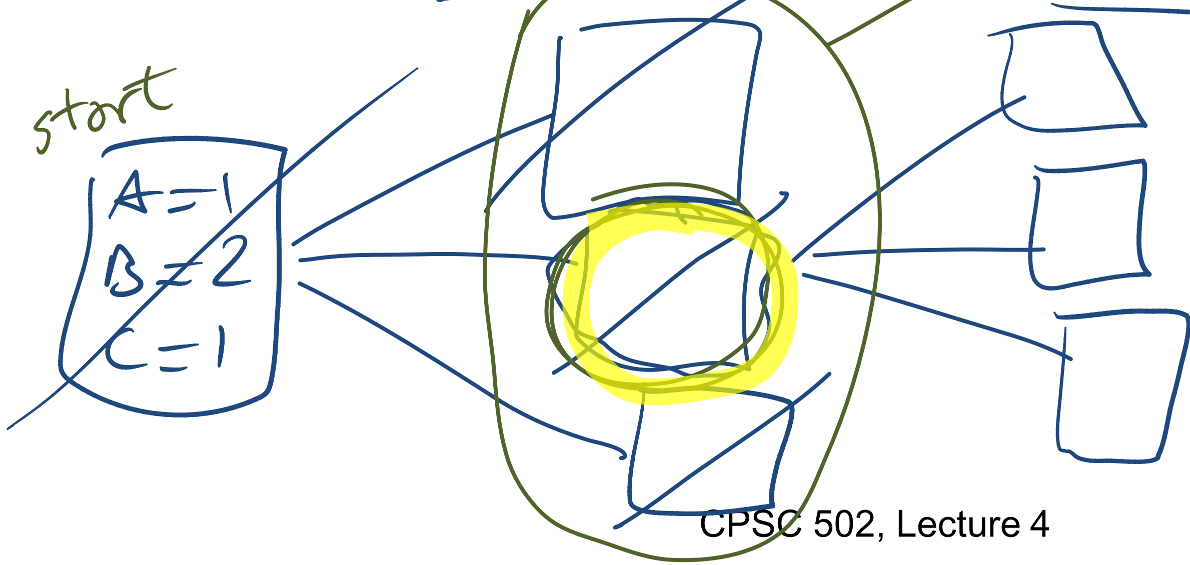
Local Search: General Method

Remember , for CSP a solution is a **possible world**

- Start from a possible world (not a path)
- Generate some neighbors (“similar” possible worlds)
- Move from the current node to a neighbor, selected according to a particular strategy

neighbors of start

  - Example: A,B,C same domain {1,2,3}



Local Search: Selecting Neighbors

How do we determine the neighbors?

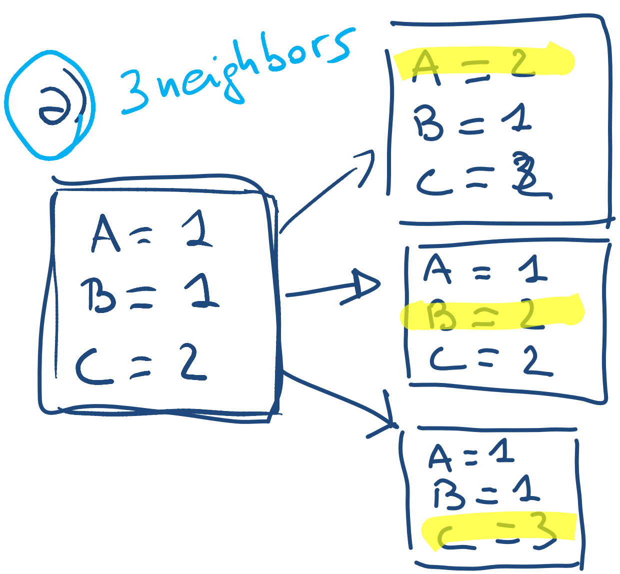
- Usually this is simple: some small incremental change to the variable assignment
  a) assignments that differ in one variable's value, by (for instance) a value difference of +1
  b) assignments that differ in one variable's value
  c) assignments that differ in two variables' values, etc.

6 neighbors

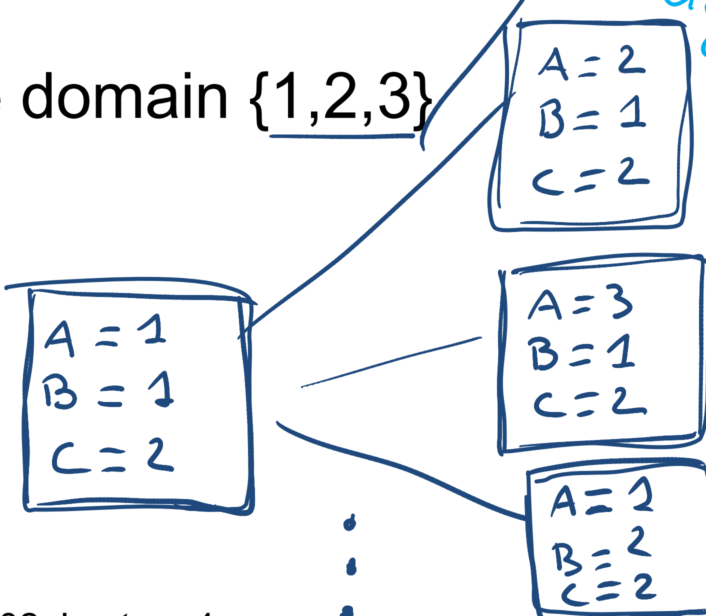
c) 12 neighbors

- Example: A,B,C same domain {1,2,3}

a) 3 neighbors

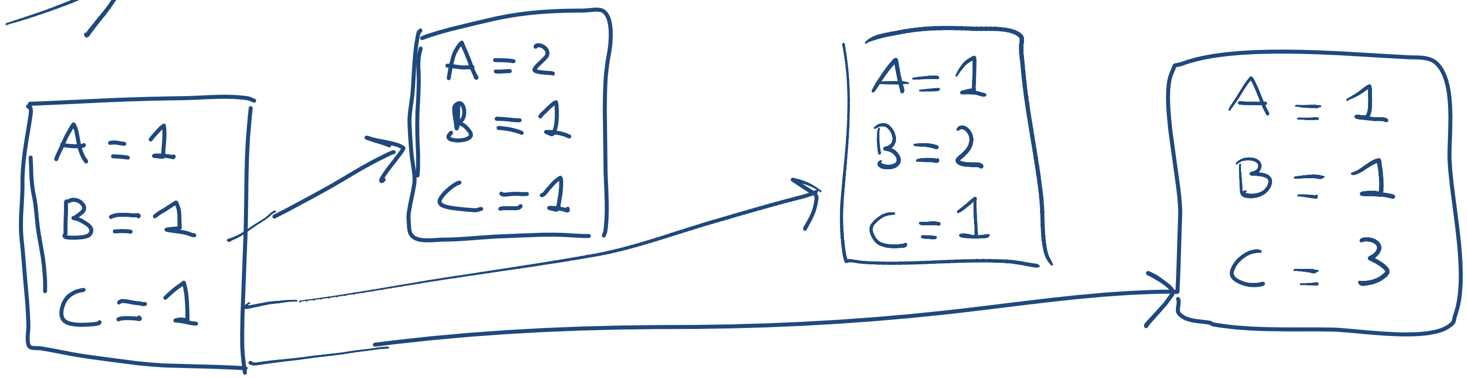


b)



Selecting the best neighbor

- Example: A,B,C same domain {1,2,3} , (A=B, A>1, C≠3)

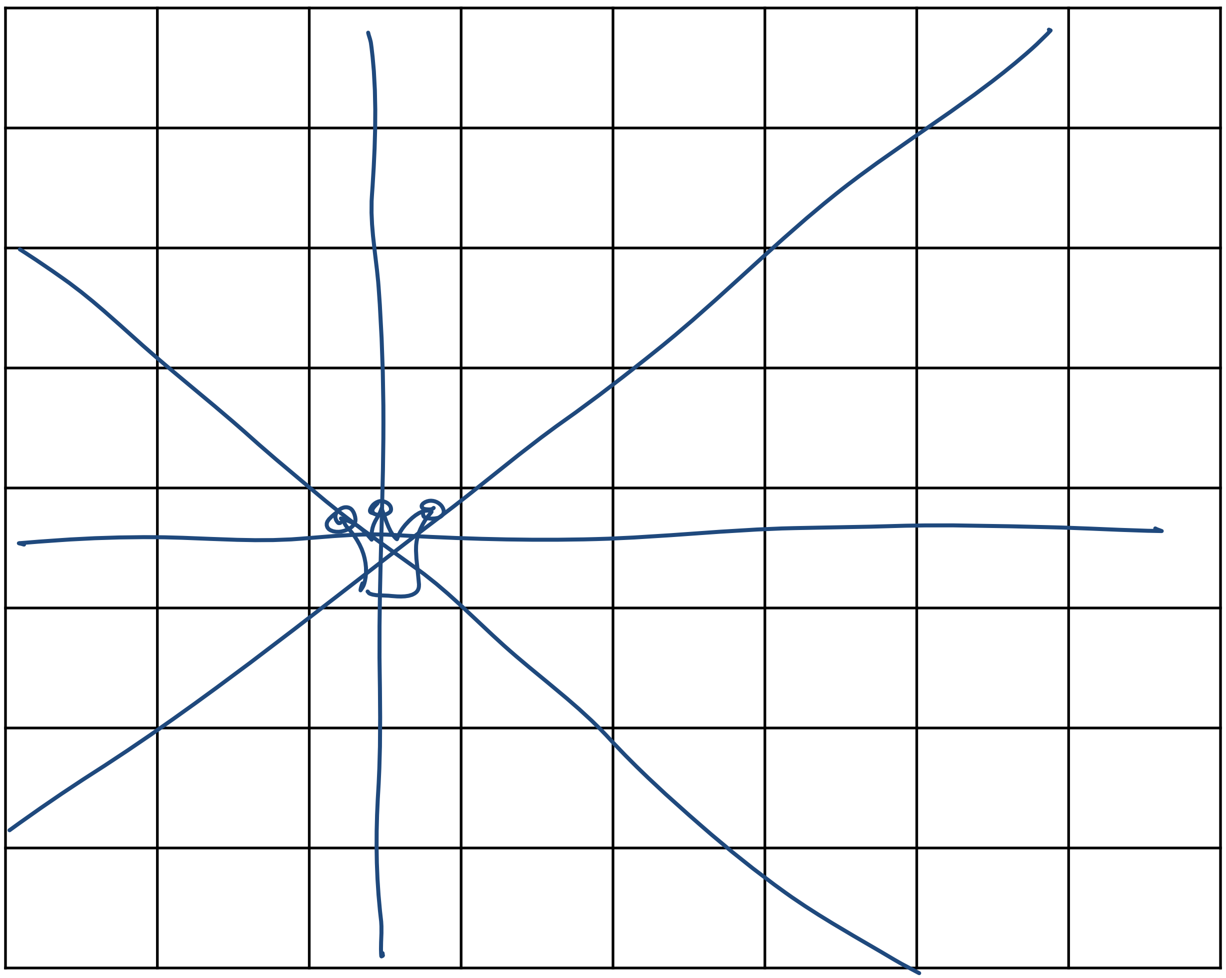


A common component of the scoring function (heuristic) => select the neighbor that results in the ……

- the min conflicts heuristics

Queens in Chess

Positions a queen can attack



Example: *n*-queens

Put *n* queens on an *n* × *n* board with no two queens on the same row, column, or diagonal (i.e attacking each other)

Example: 4-Queens

~~States~~: 4 queens in 4 columns ($4^4 = 256$ states)

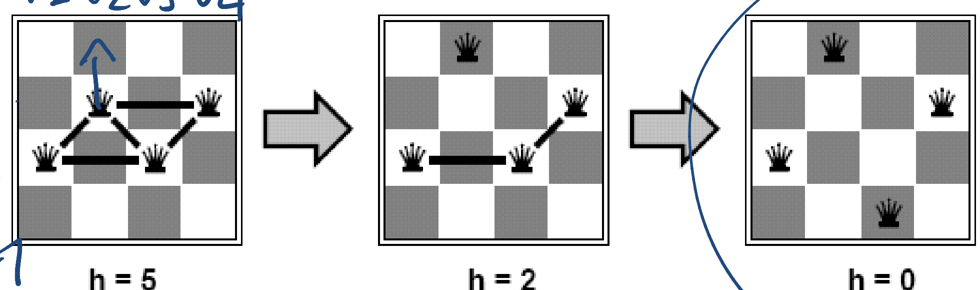
Operators: move queen in column (to generate neighbors)

Goal test: no attacks

Evaluation: $h(n)$ = number of attacks

V_1 V_2 V_3 V_4

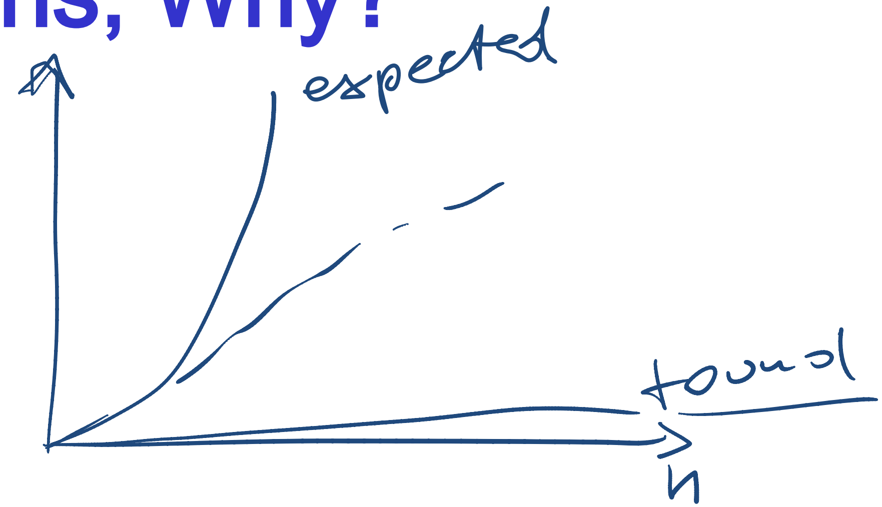
0
1
2
3



solution

$V_1 = 2$
$V_2 = 1$
$V_3 = 2$
$V_4 = 1$

n-queens, Why?



Why this problem?

Lots of research in the 90' on local search for CSP was generated by the observation that the run-time of local search on n-queens problems is **independent of problem size!**

Given random initial state, can solve n-queens in almost constant time for arbitrary n with high probability (e.g., $n = 10{,}000{,}000$)

Constrained Optimization Problems

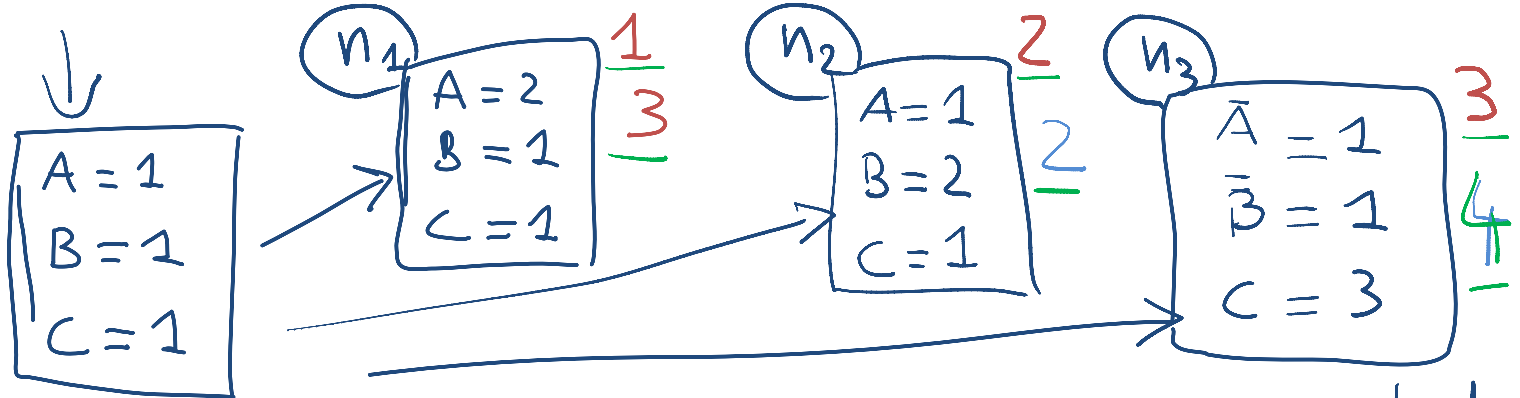
So far we have assumed that we just want to find a possible world that satisfies all the constraints.

But sometimes solutions may have different values / costs

- We want to find the optimal solution that
  - maximizes the value or
  - minimizes the cost

Constrained Optimization Example

- Example: A,B,C same domain {1,2,3} , (A=B, A>1, C≠3)
- Value = (C+A) so we want a solution that maximize that



The scoring function we'd like to maximize might be: select n1

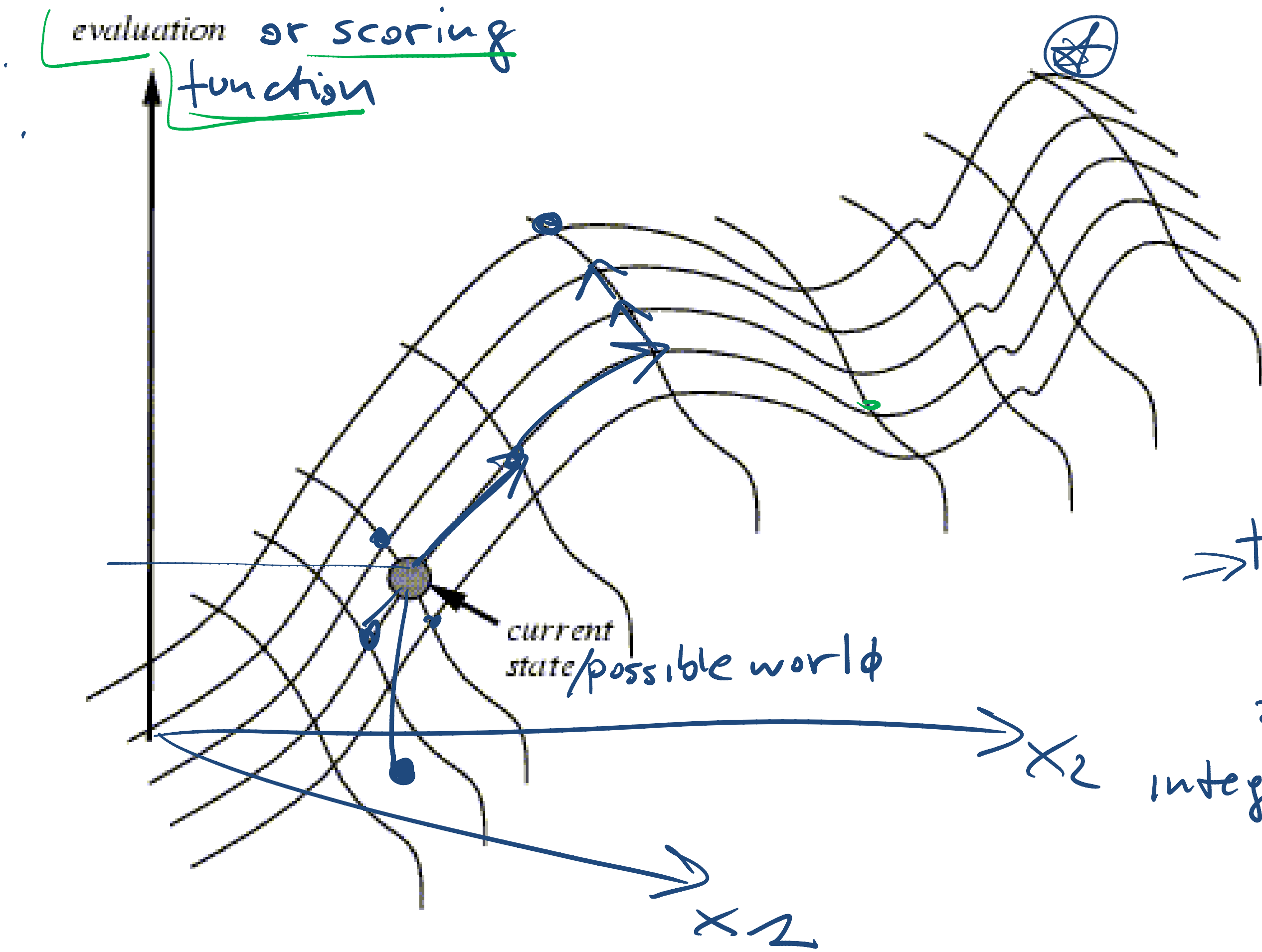
f(n) = (C + A) - #-of-conflicts   f(n1)=2   f(n2)=0 f(n3)=1

Hill Climbing means selecting the neighbor which best improves a (value-based) scoring function.

Greedy Descent means selecting the neighbor which minimizes a (cost-based) scoring function. cost + # of conflicts

Hill Climbing

NOTE: Everything that will be said for Hill Climbing is also true for Greedy Descent

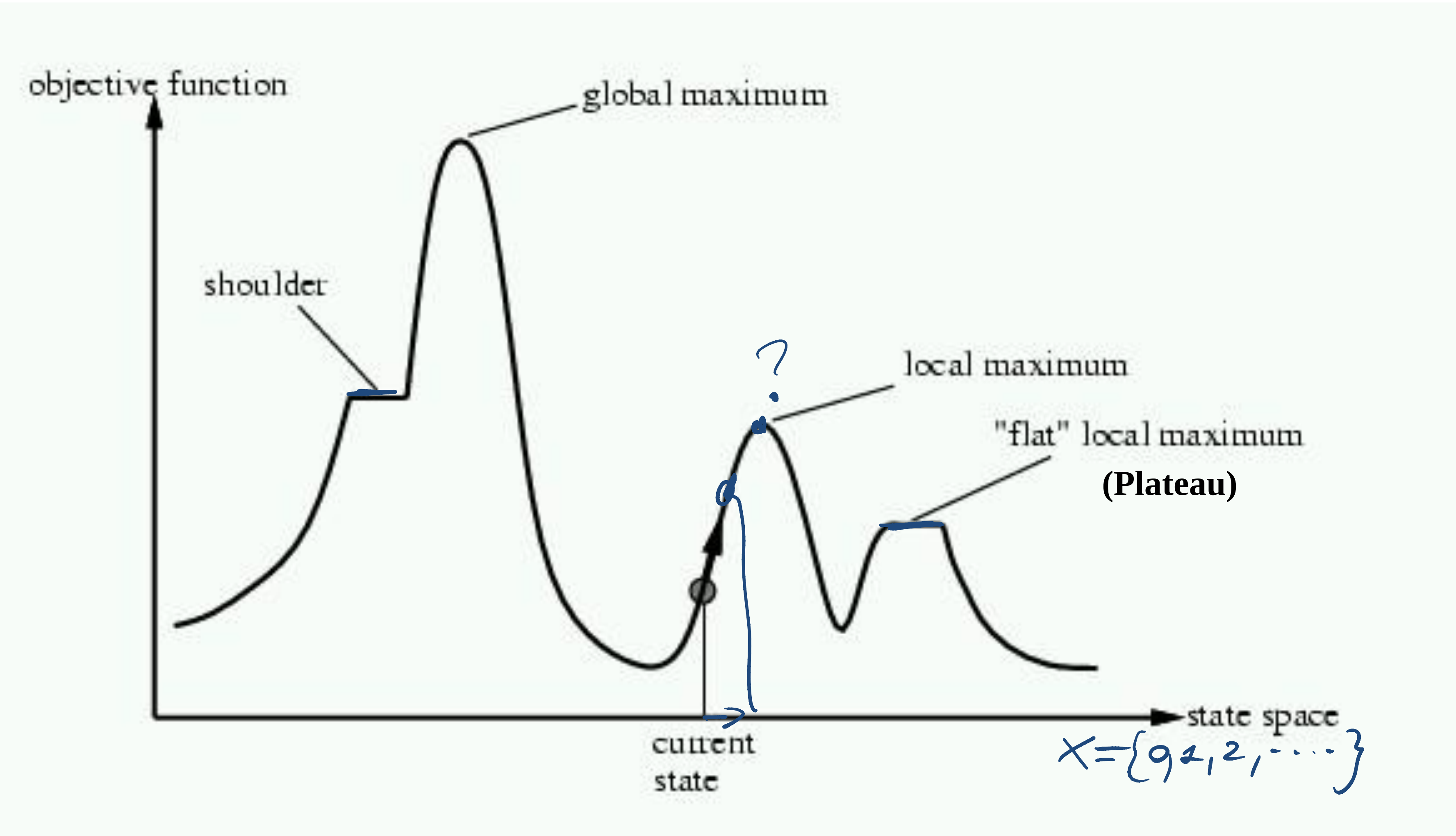


→ two vars
X_1 X_2
assume domain
integer [0, 1, 2 . . . .]

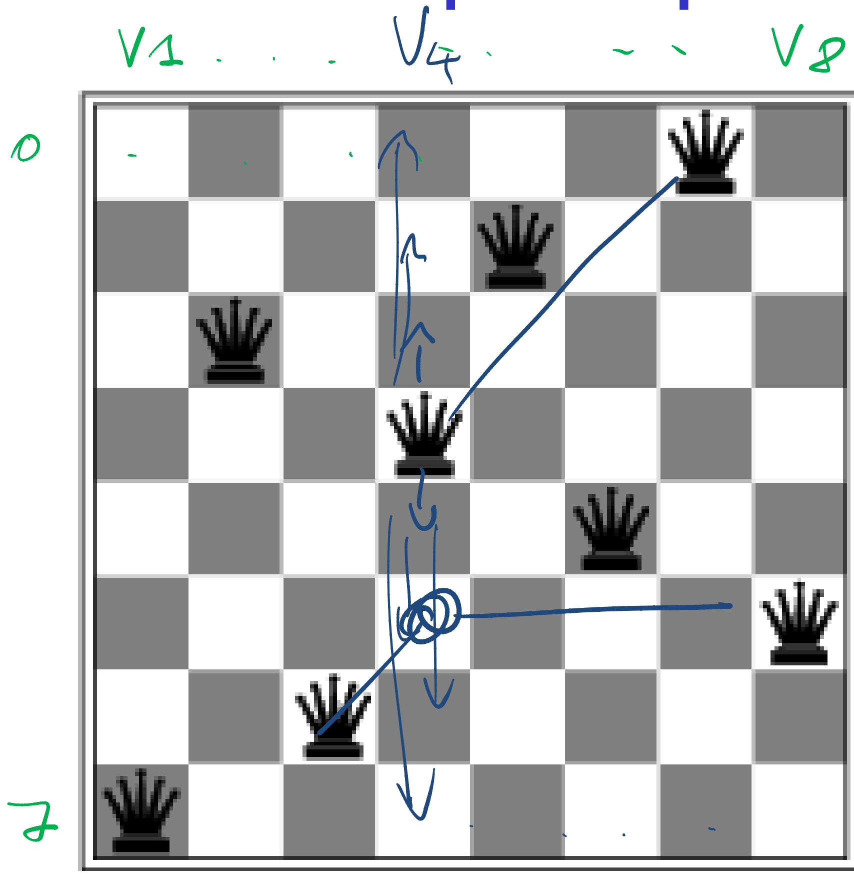
Problems with Hill Climbing

Local Maxima.

Plateau - Shoulders



Corresponding problem for GreedyDescent Local minimum example: 8-queens problem



for all the moves (neighbors)

h > 1

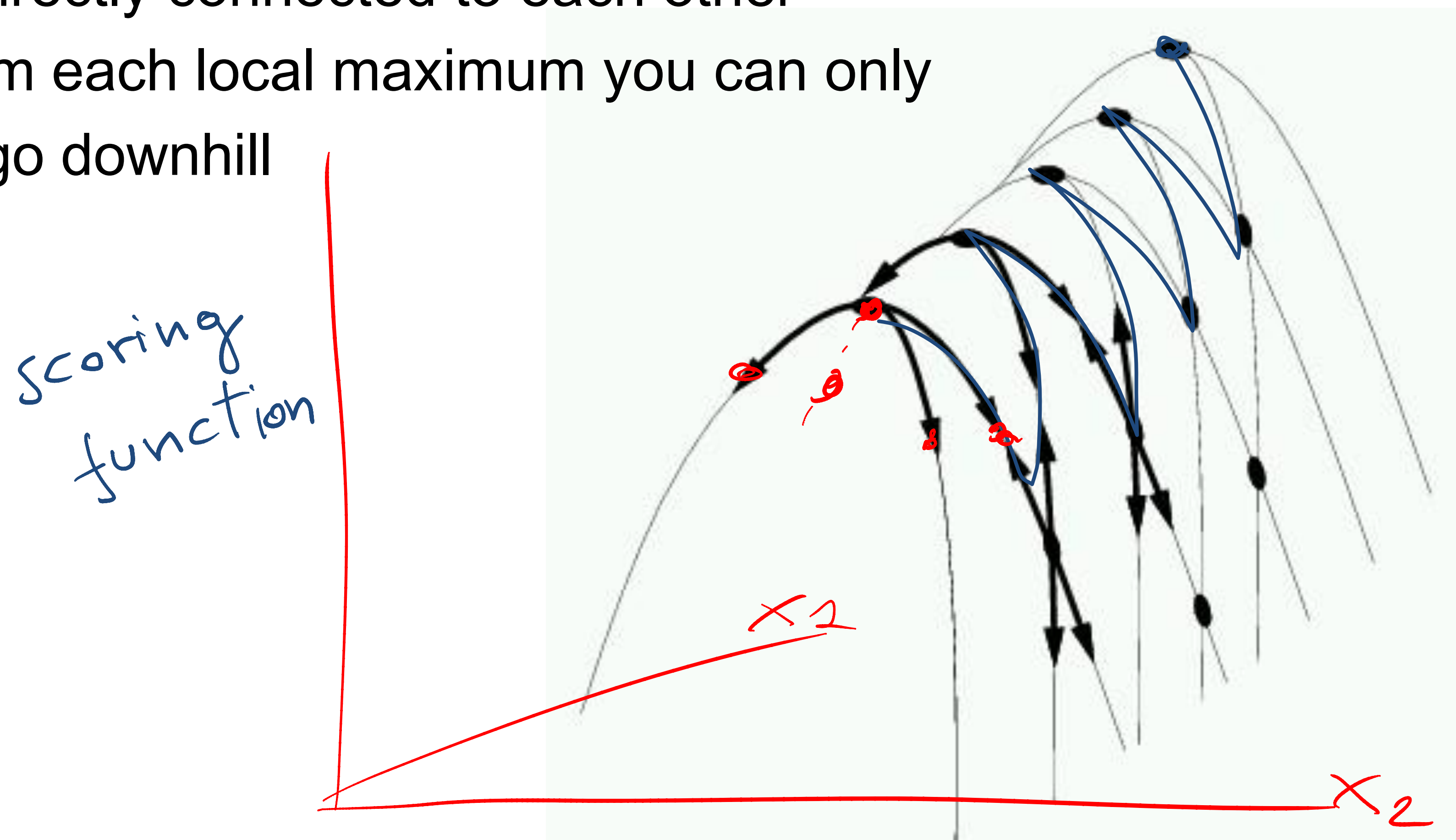
A local minimum with *h = 1*

h = 0
for solution

Even more Problems in higher dimensions

E.g., Ridges – sequence of local maxima not directly connected to each other

From each local maximum you can only go downhill



Today Sept 20

Stochastic Local Search (SLS)

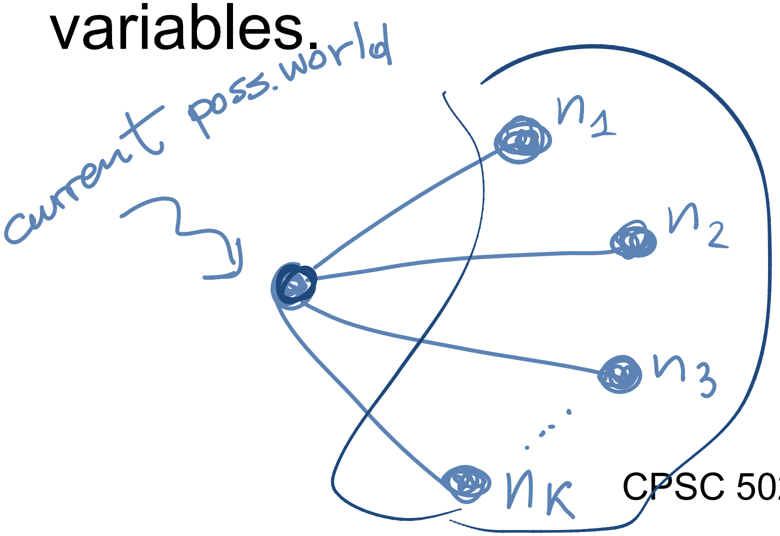
- Local Search & Constrained Optimization
- **SLS**
- SLS variants
- Comparing SLS

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?

$\boxed{n\,(d-1)}$

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

E.G in 8-queen

- How many neighbors? $\boxed{8 \cdot 7 = 56}$
- ……..

1. choose one of the circled ones

2 choose randomly one of the 56

8 variables

8 values

# of conflicts

	V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8
1	18	12	14	13	13	12	14	14
2	14	16	13	15	12	14	12	16
3	14	12	18	13	15	12	14	14
4	15	14	14	♛	13	16	13	16
5	♛	14	17	15	♛	14	16	16
6	17	♛	16	18	15	♛	15	♛
7	18	14	♛	15	15	14	♛	16
8	14	14	13	17	12	14	12	18

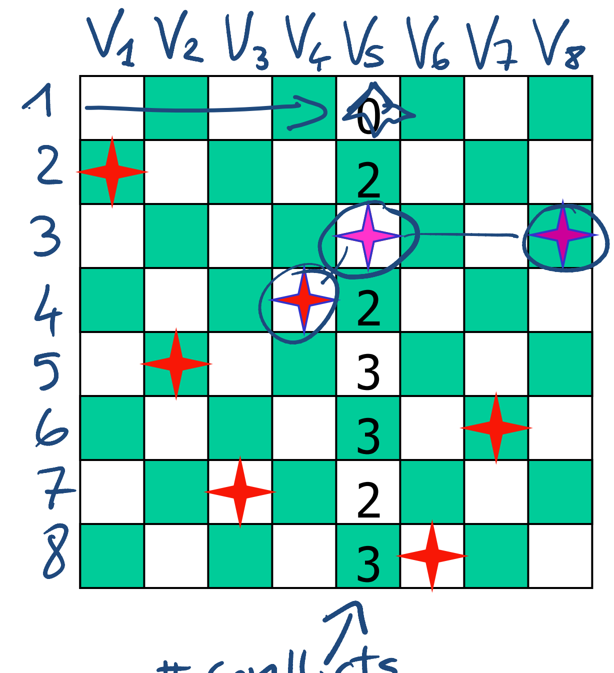
Random Steps (Walk): two-step

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 $V_5 = 1$



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

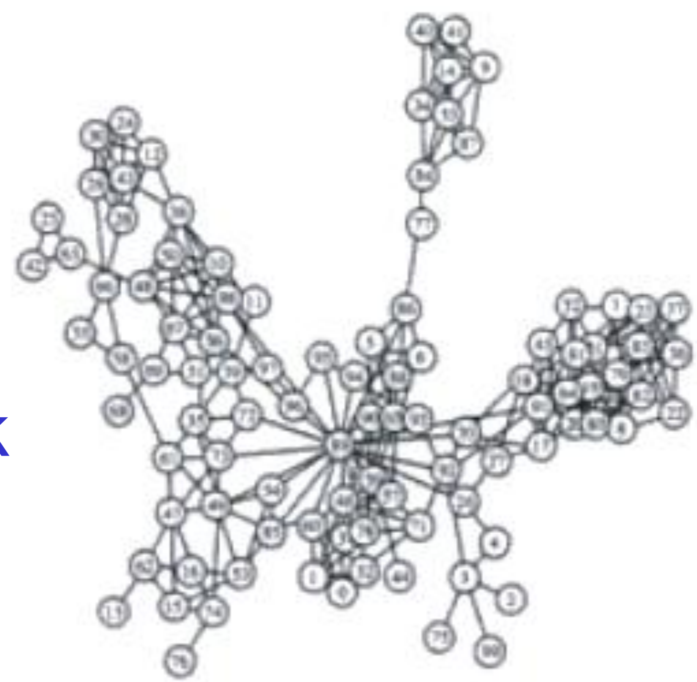
CSPs: Radio link frequency assignment

Assigning frequencies to a set of radio links defined between pairs of sites in order **to avoid interferences**.

Constraints on frequency depend on **position of the links** and on **physical environment** .

Source: *INRIA*

Sample Constraint network



Example: SLS for RNA secondary structure design

RNA strand made up of four bases: cytosine (C), guanine (G), adenine (A), and uracil (U)

2D/3D structure RNA strand folds into is important for its function

Predicting structure for a strand is “easy”: $O(n^3)$

But what if we want a strand that folds into a certain structure?

- Local search over strands
  - ✓ Search for one that folds into the right structure
- Evaluation function for a strand
  - ✓ Run $O(n^3)$ prediction algorithm
  - ✓ Evaluate how different the result is from our target structure
  - ✓ Only defined implicitly, but can be evaluated by running the prediction algorithm

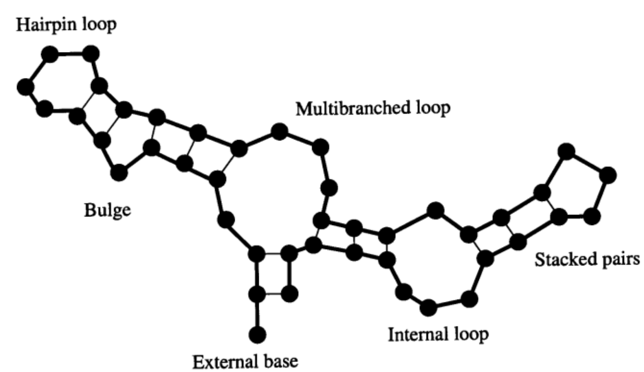
RNA strand

GUCCCAUAGGAUGUCCCAUAGGA

Easy

Hard

Secondary structure



Best algorithm to date: Local search algorithm RNA-SSD developed at UBC
[Andronescu, Fejes, Hutter, Condon, and Hoos, Journal of Molecular Biology, 2004]

SLS:Limitations

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

- Not able to show that no solution exists

Today Sept 20

Stochastic Local Search (SLS)

- Local Search & Constrained Optimization
- SLS
- **SLS variants**
- Comparing SLS

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

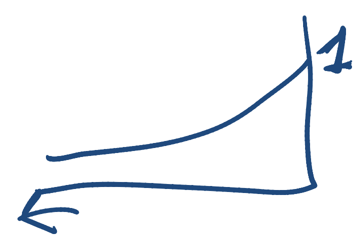
Here's how it works (for maximizing): h

- 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.

$h(n') < h(n)$; $h(n') - h(n) < 0$

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

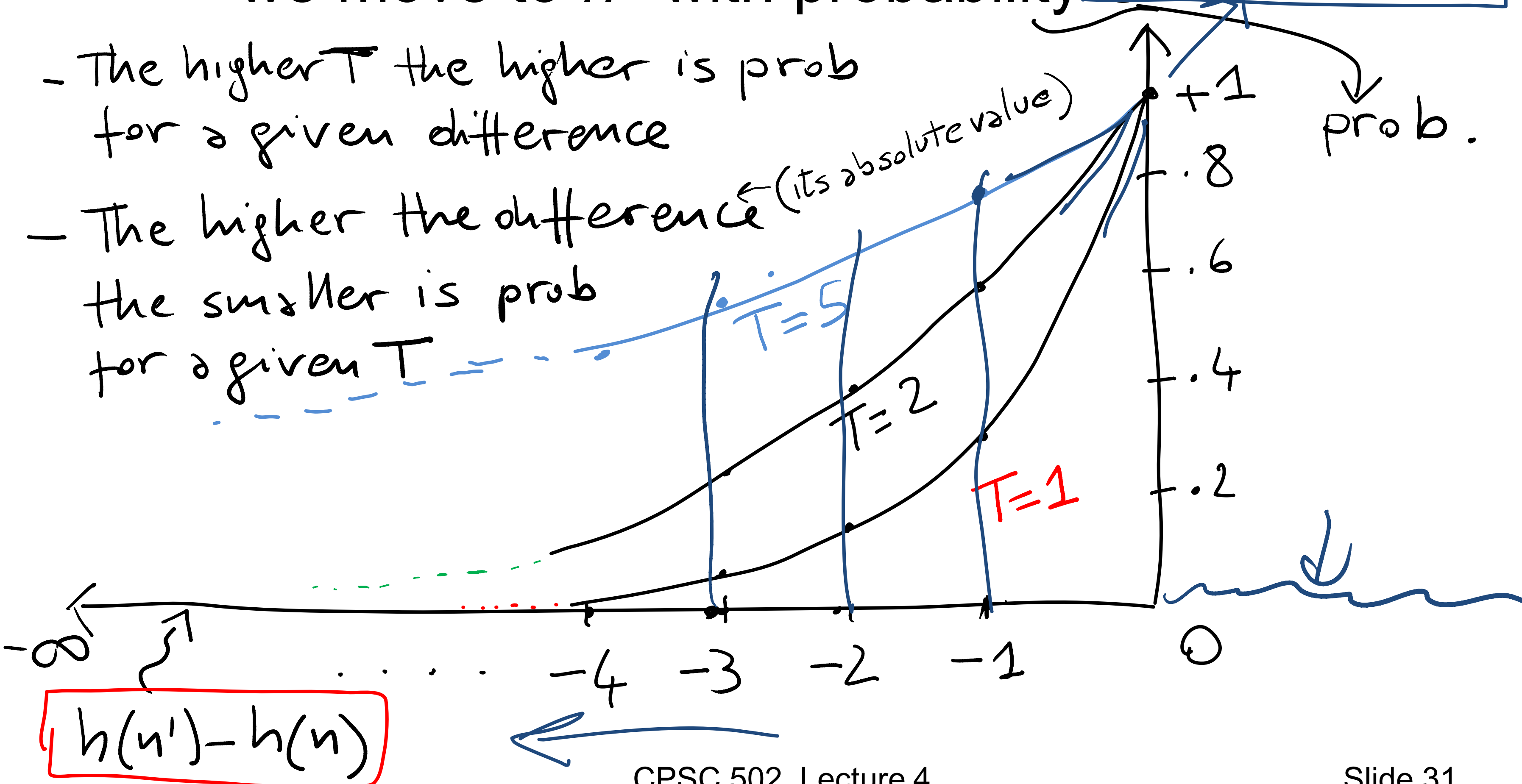
see next slide



- If it isn't an improvement, adopt it probabilistically depending on the difference and a temperature parameter, *T*.
  - we move to n' with probability $\boxed{e^{(h(n')-h(n))/T}}$

– The higher T the higher is prob for a given difference

– The higher the difference (its absolute value) the smaller is prob for a given T



Properties of simulated annealing search

One can prove: If *T* decreases slowly enough, then simulated annealing search will find a global optimum with probability approaching 1

Widely used in VLSI layout, airline scheduling, etc.

Population Based SLS

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

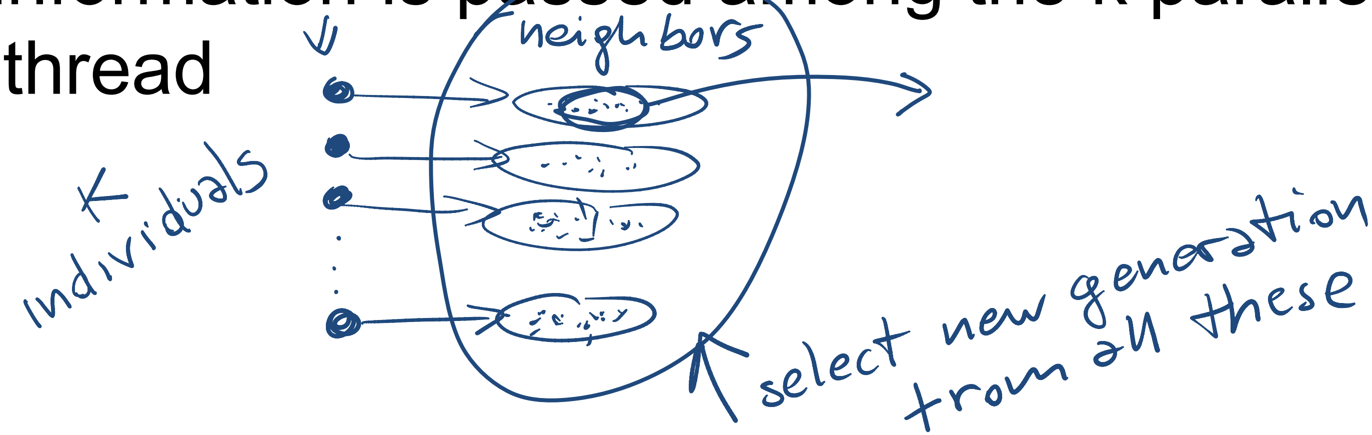
Key Idea: maintain a population of *k* individuals

- At every stage, update your population.
- Whenever one individual is a solution, report it.

Population Based SLS: Beam Search

Non Stochastic

- Start with *k* individuals, and 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)

m neighbors $\underbrace{\{n_1 \ldots n_m\}}$

h: scoring function

Probability of selecting $(n_J) = \dfrac{h(n_J)}{\sum_{n_i} h(n_i)}$

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)

Population Based SLS: Genetic Algorithms

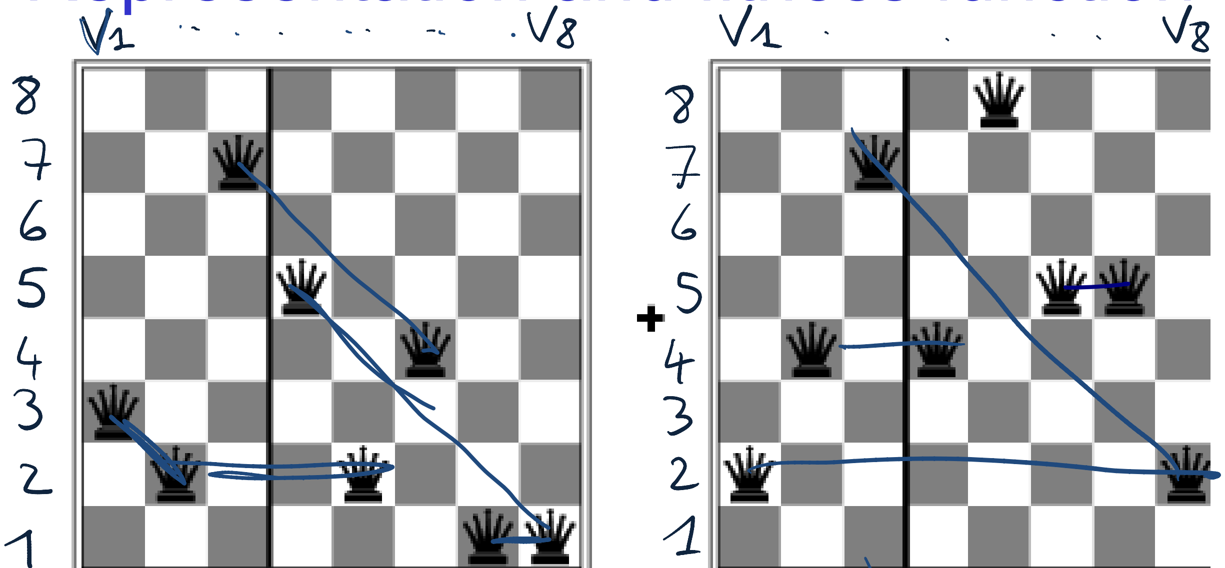
- Start with *k* randomly generated individuals (population)
- An individual is represented as a string over a finite alphabet (often a string of 0s and 1s)
- A successor is generated by combining two parent individuals (loosely analogous to how DNA is spliced in sexual reproduction)
- Evaluation/Scoring function (fitness function). Higher values for better individuals.
- Produce the next generation of individuals by selection, crossover, and mutation

Genetic algorithms: Example 8-queen

Representation and fitness function

of queen pairs possibly attacking each other

$$\frac{8 \cdot 7}{2} = 28$$



5 attacks

4 attacks

28 − 4

24748552	24
32752411	23

(28 − 5)

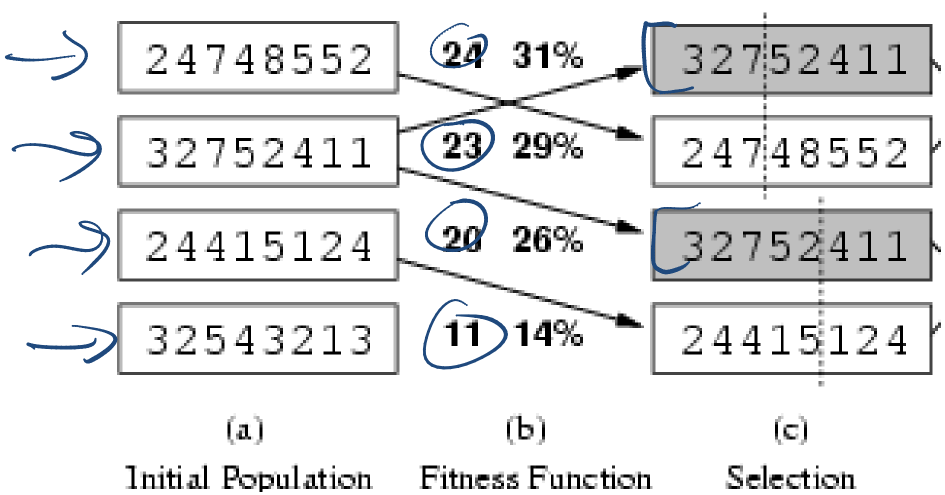
State: string over finite alphabet

Fitness function: higher value better states.

queen pairs not attacking each other

Genetic algorithms: Example

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



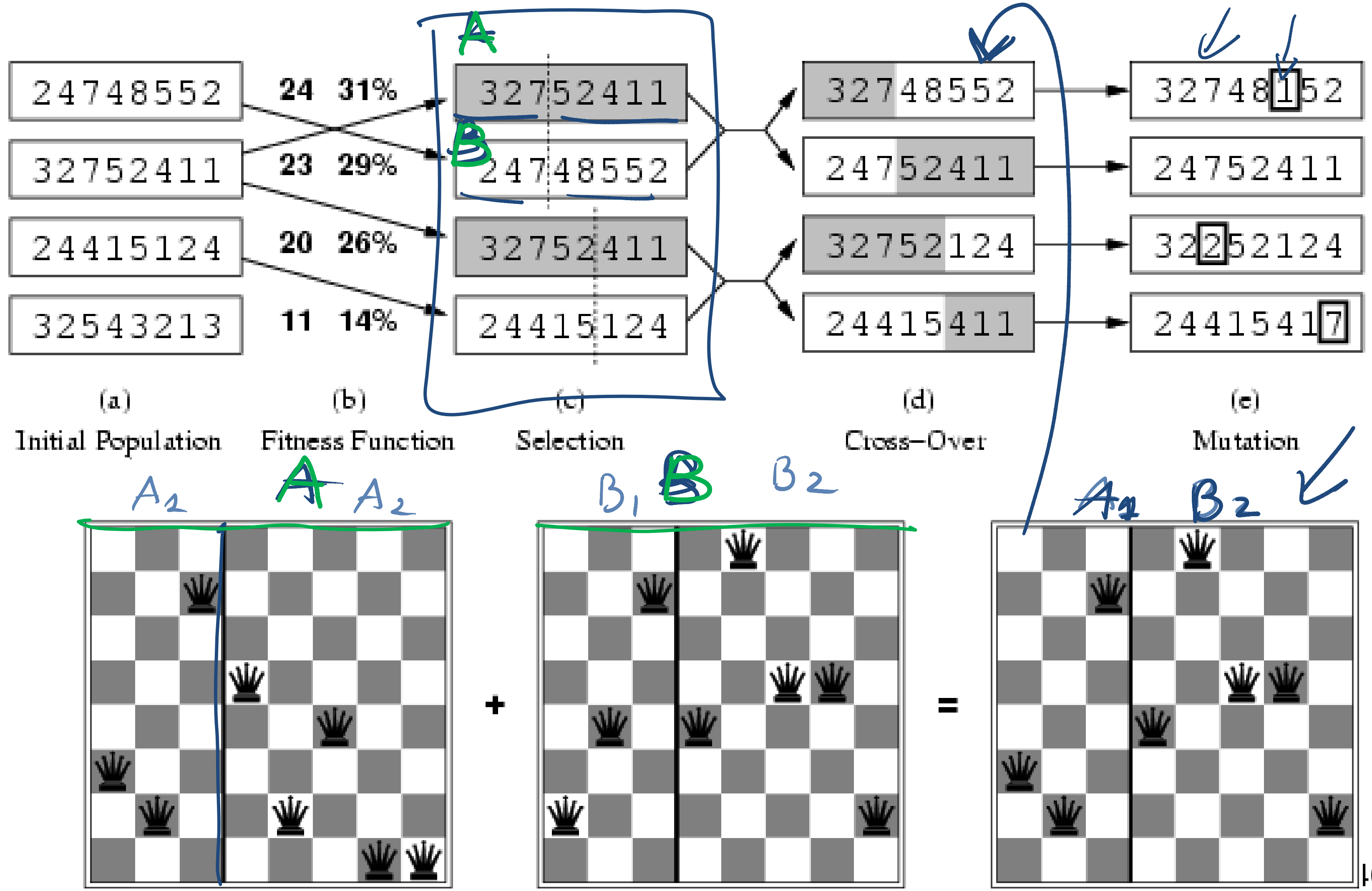
24/(24+23+20+11) = 31%

23/(24+23+20+11) = 29% etc

same as Beam Search

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!)

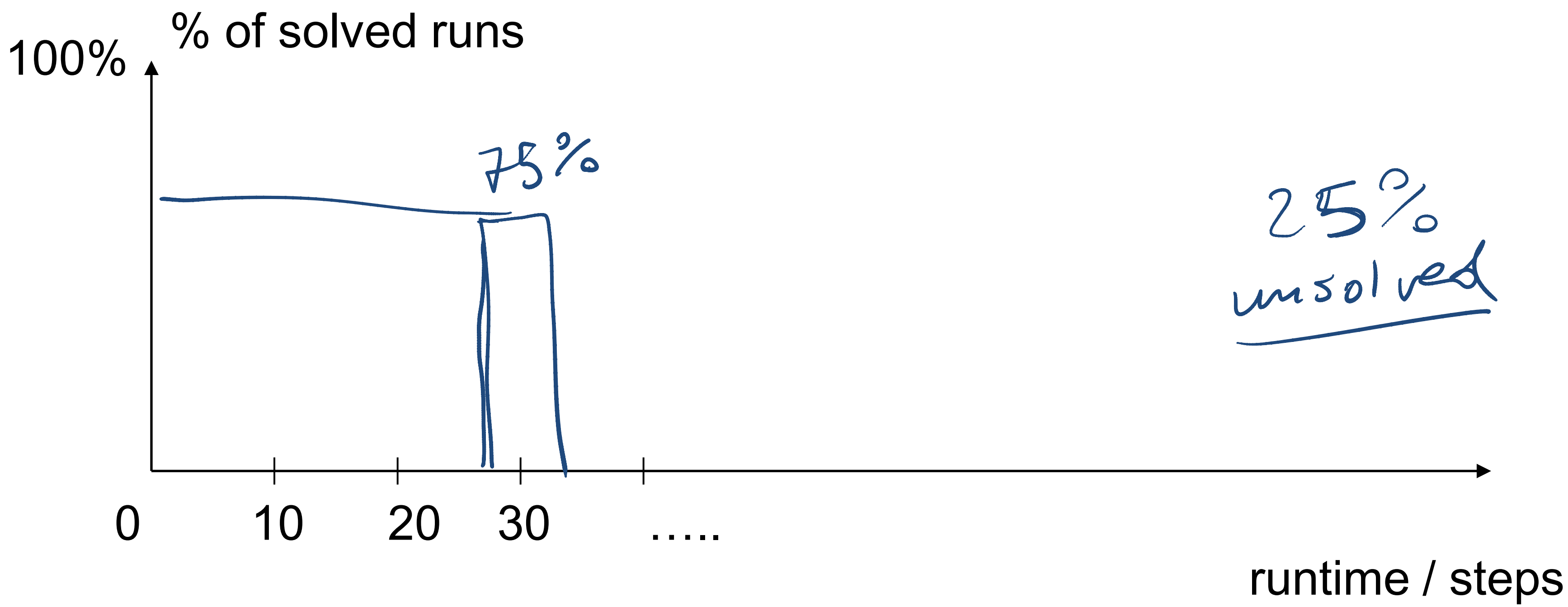
Today Sept 20

Stochastic Local Search (SLS)

- Local Search & Constrained Optimization
- SLS
- SLS variants
- Comparing SLS

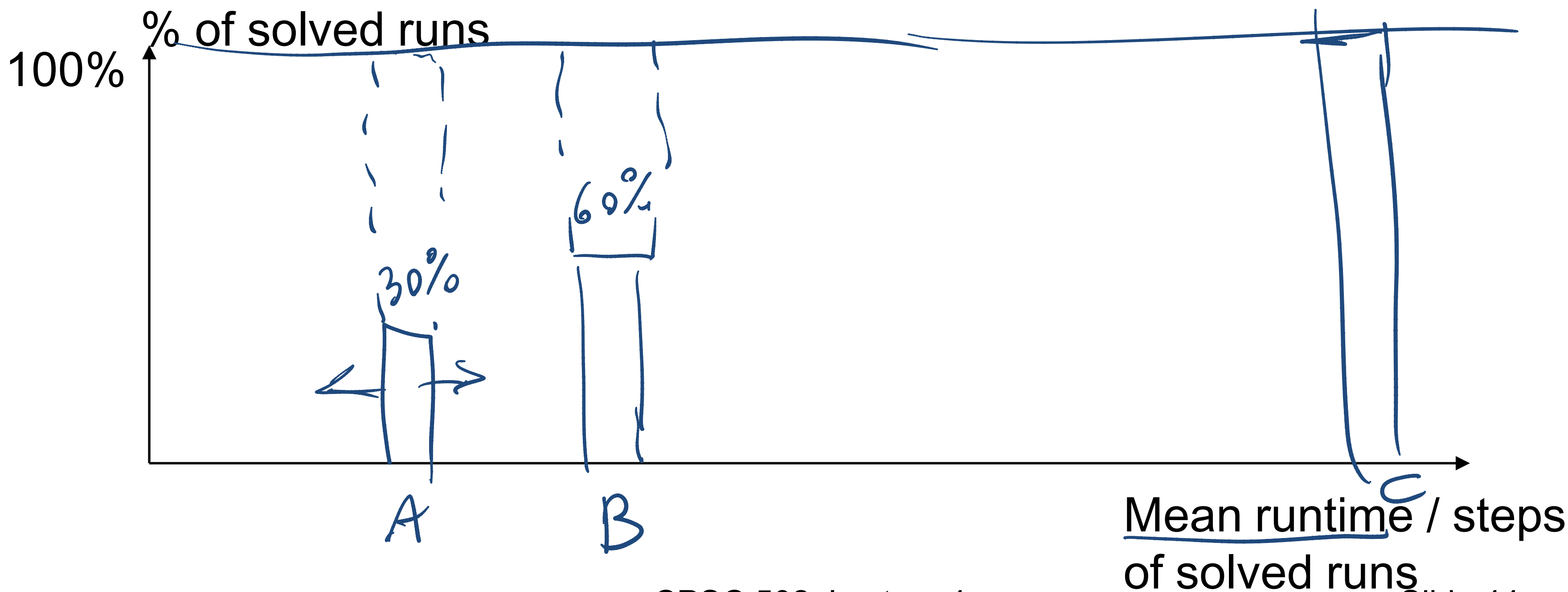
Comparing Stochastic Algorithms: Challenge

- 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….

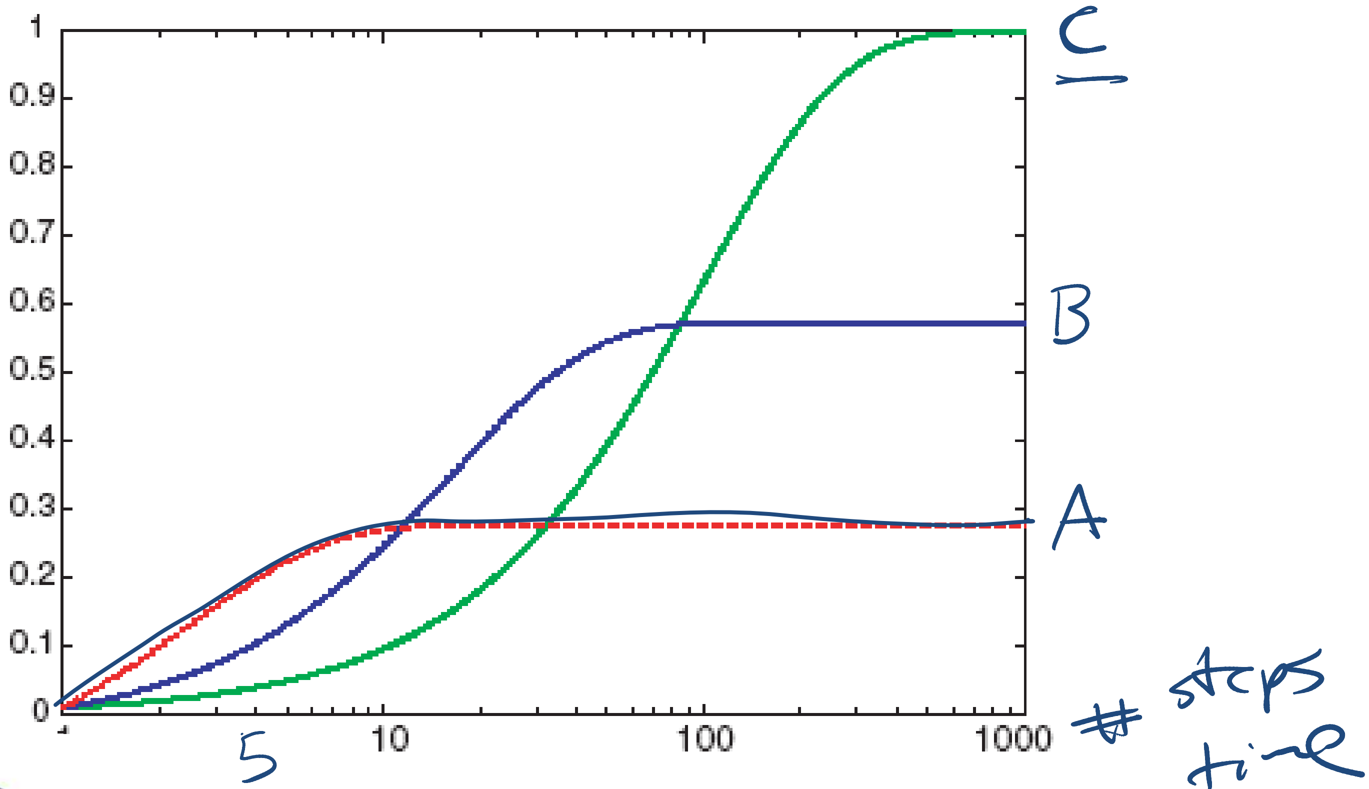
- How can you compare three algorithms when
  A. one solves the problem 30% of the time very quickly but doesn't halt for the other 70% of the cases
  B. one solves 60% of the cases reasonably quickly but doesn't solve the rest
  C. one solves the problem in 100% of the cases, but slowly?



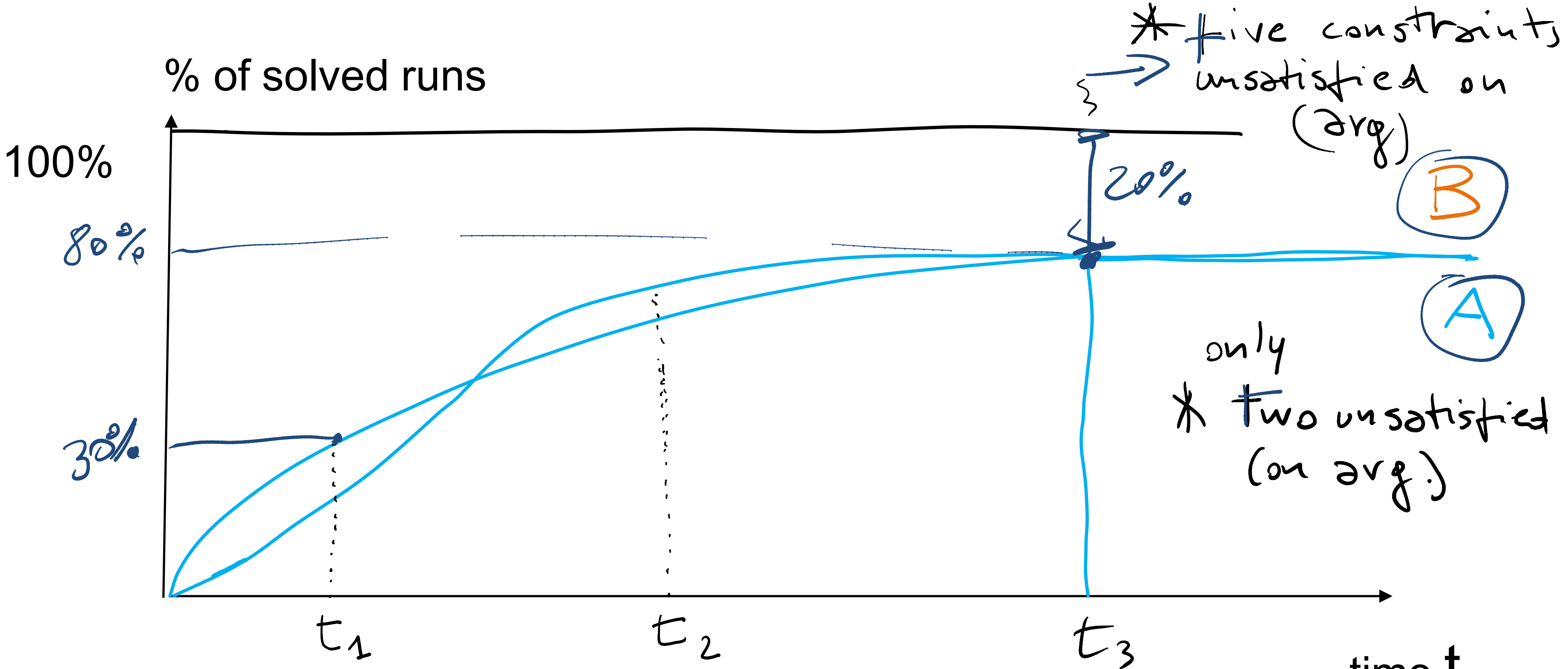
Runtime Distributions are even more informative

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



Runtime Distributions



Which one would you use if you could only wait

- t = $\boxed{t_1}$? A
- t = t_2? B
- t = t_3? A (because of the quality of the answers on unsolved problems see *)

Stochastic Local Search

- **Key Idea:** combine greedily improving moves with randomization
  - As well as improving steps we can allow a "small probability" of: e.g.
    - Random steps: move to a random neighbor. 1%
    - Random restart: reassign random values to all variables. 5%
  - Always keep best solution found so far
  - Stop when
    - Solution is found (in vanilla CSP pw that satisfies all C)
    - Run out of time (return best solution so far)

CSPs summary

Find a single variable assignment that satisfies all of our constraints (atemporal)

- Systematic Search approach
  - Constraint network support $n^2 d^3$
    - ✓ inference e.g., Arc Consistency (can tell you if solution does not exist)
    - ✓ Decomposition (loop, more AC)
  - Heuristic Search (degree, min-remaining)
- (Stochastic) Local Search (search space …..?)
  - Huge search spaces and highly connected constraint network but solutions densely distributed
  - No guarantee to find a solution (if one exists).
  - Unable to show that no solution exists

R&Rsys we'll cover in this course

Environment

Problem		Deterministic	Stochastic
Static	Constraint Satisfaction	*Vars + Constraints* Arc Consistency Search SLS	
	Query	*Logics* → Propositional → First Order → . . . . Search	*Belief Nets* Var. Elimination Approx. Inference Temporal. Inference
Sequential	Planning	*STRIPS* actions precds effects Search	aka influence diagrams *Decision Nets* Var. Elimination *Markov Processes* Value Iteration

Representation

Reasoning Technique

TODO for this Thur

Read Chp 8 of textbook (Planning with Certainty)

Do exercise 4.C

http://www.aispace.org/exercises.shtml

Please, look at solutions only after you have tried hard to solve them!

Arc Consistency Algorithm: Complexity

- Let's determine Worst-case complexity of this procedure (compare with DFS d^n)
  - let the max size of a variable domain be ***d***
  - let the number of variables be ***n***
  - The max number of binary constraints is……. $n(n-1)/2$
- How many times the same arc can be inserted in the ToDoArc list? d

$\boxed{O(d^3 n^2)}$

- How many steps are involved in checking the consistency of an arc? d^2

$\{X_1 \ldots X_d\}$ $\{Y_1 \ldots Y_d\}$

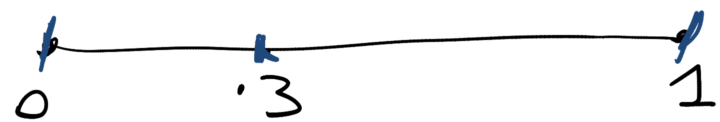
OVERALL COMPLEXITY

Sampling a discrete probability distribution

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

$P = .3$

generate random number in [0,1]



If < .3 accept n'

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

SAME HERE

→ n_1 $\boxed{P_1 = .1}$

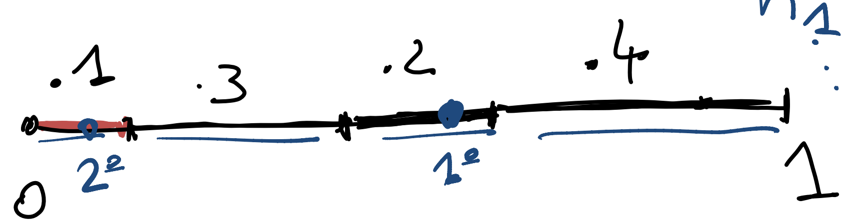
→ n_2 $\boxed{P_2 = .3}$

→ n_3 $\boxed{P_3 = .2}$

→ n_4 $\boxed{P_4 = .4}$

n_3 first sample

n_1 second sample



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