Stochastic Local Search Variants

Computer Science cpsc322, Lecture 16

(Textbook Chpt 4.8)

Oct, 11, 2013

Lecture Overview

- **Recap SLS**
- SLS variants

Announcements

- No office hours today. Sorry
- Assignmnet-2 will be posted over the weekend

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 …p.w. satisfying all C……….)
    - Run out of time (return best solution so far)

Runtime distributions in AIspace

- Let's look at some algorithms and their runtime distributions:
  1. Greedy Descent
  2. Random Sampling
  3. Random Walk
  4. Greedy Descent with random walk
- Simple scheduling problem 2 in AIspace:

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

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): 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 $e^{\boxed{(h(n')-h(n))/T}}$

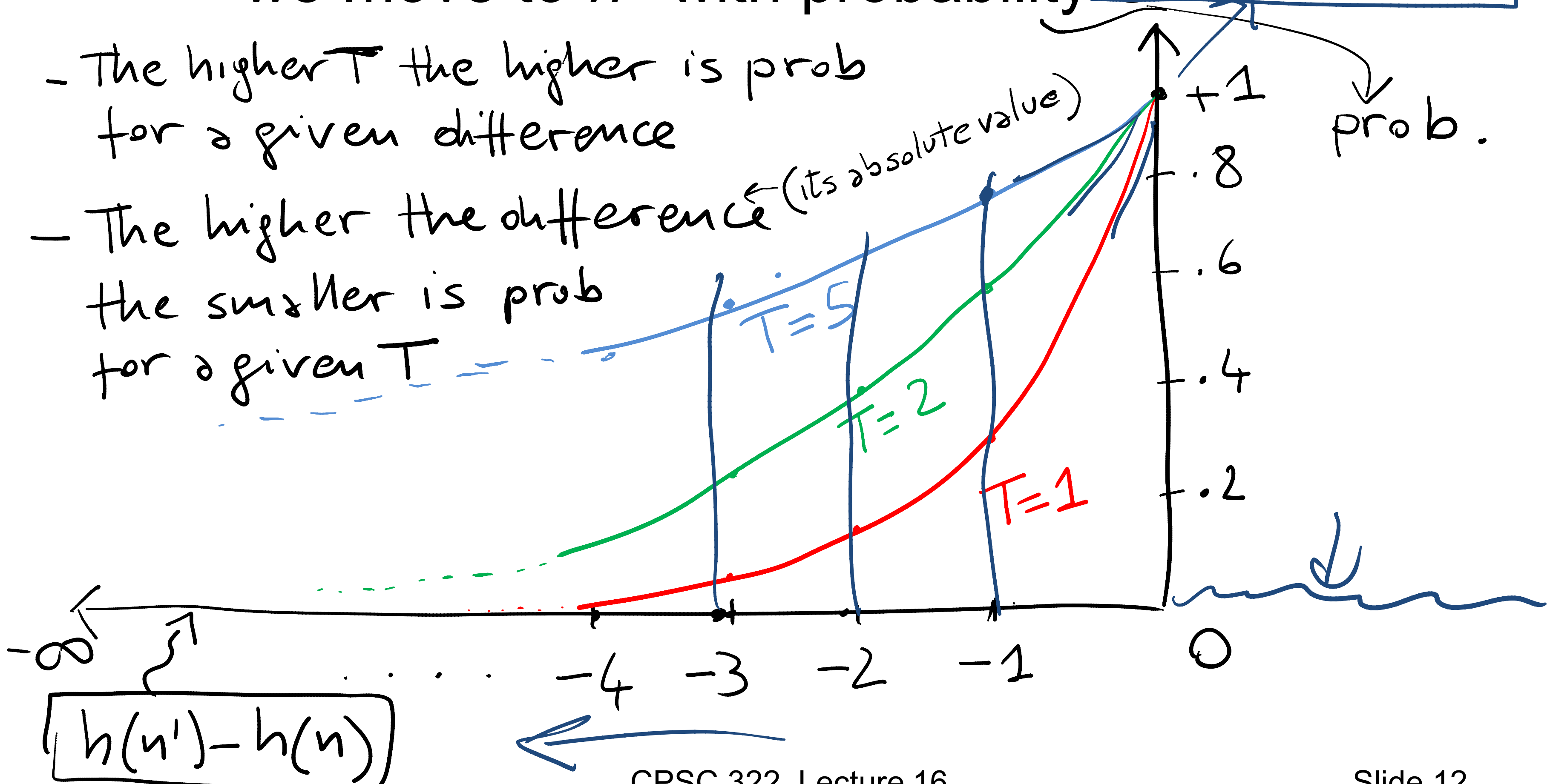
see next slide

- If it isn't an improvement, adopt it probabilistically depending on the difference and a temperature parameter, *T*.

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

- 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.

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

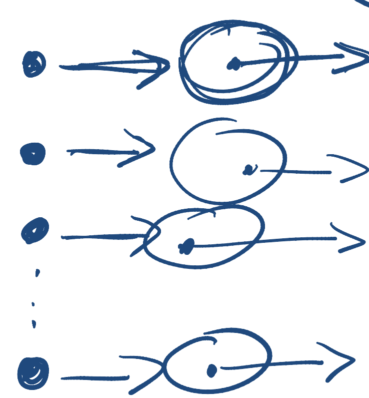
neighbors 4

- All searches are independent
- No information shared

but more memory

no reasons to use it!

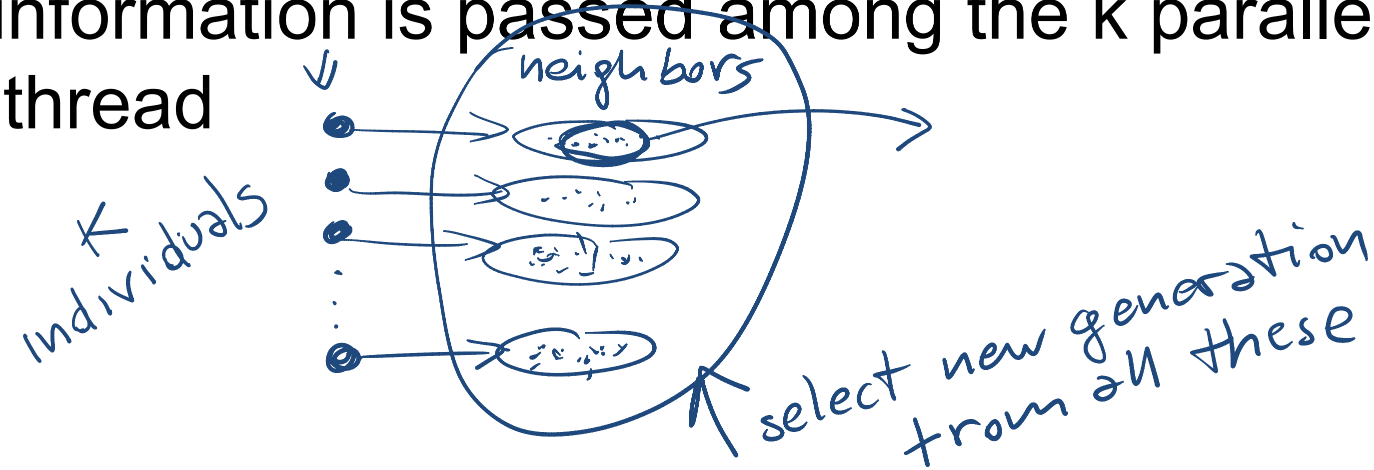
K poss. worlds



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)

iclicker.

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

h: scoring function

$$Prob\ of\ selecting\ n_j\ ?=$$

A $\dfrac{n_j}{\sum_i n_i}$

B $\dfrac{h(n_j)}{\sum_i h(n_i)}$

C $\dfrac{\sum_i h(n_i)}{h(n_j)}$

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

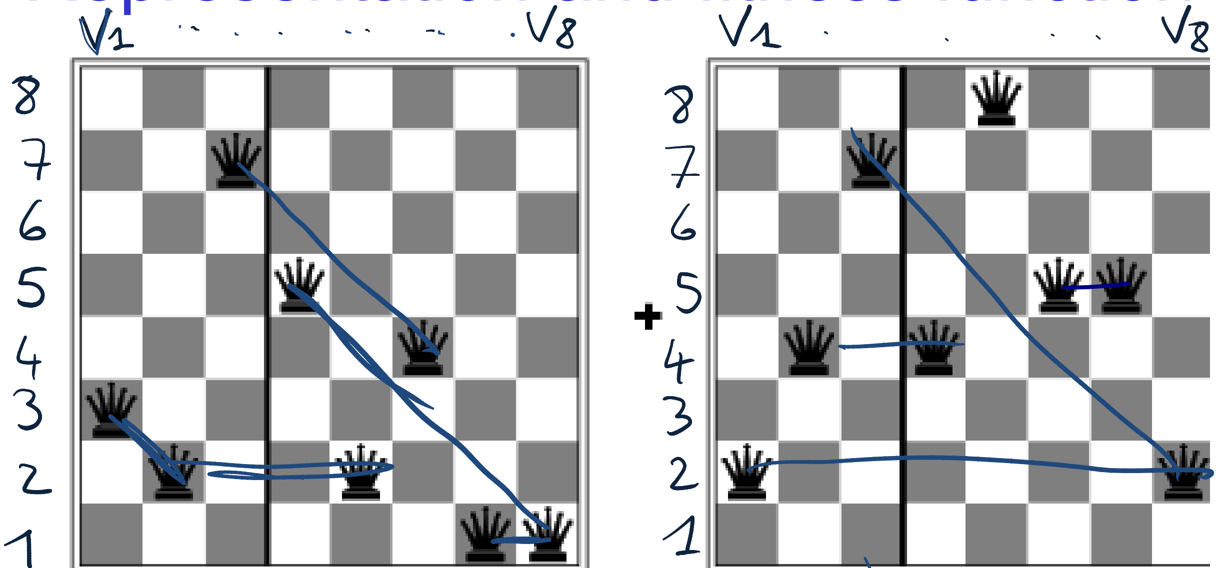
- Start with *k* randomly generated individuals (population)
- An individual is represented as a string over a finite alphabet (often a string of 0s and 1s)
- A successor is generated by combining two parent individuals (loosely analogous to how DNA is spliced in sexual reproduction)
- Evaluation/Scoring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

Genetic algorithms: Example 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

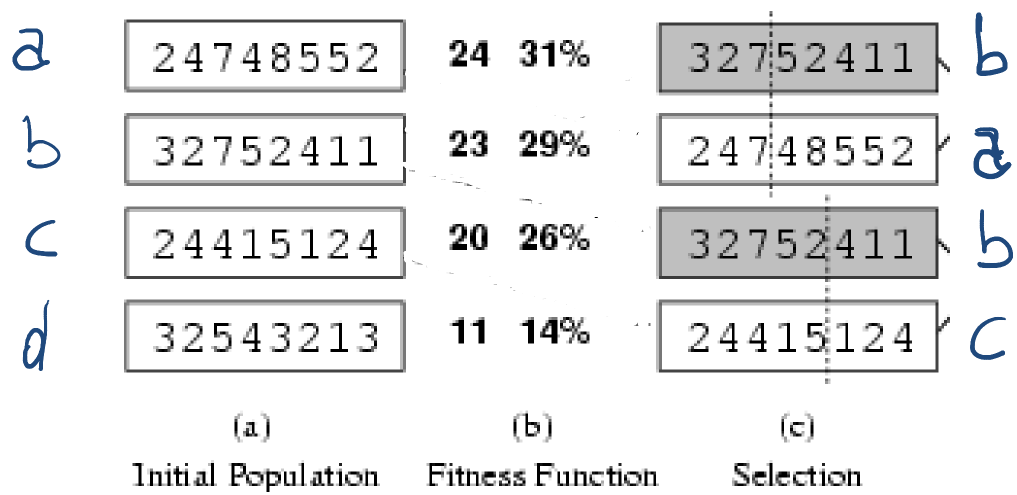
(28 − 5)

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



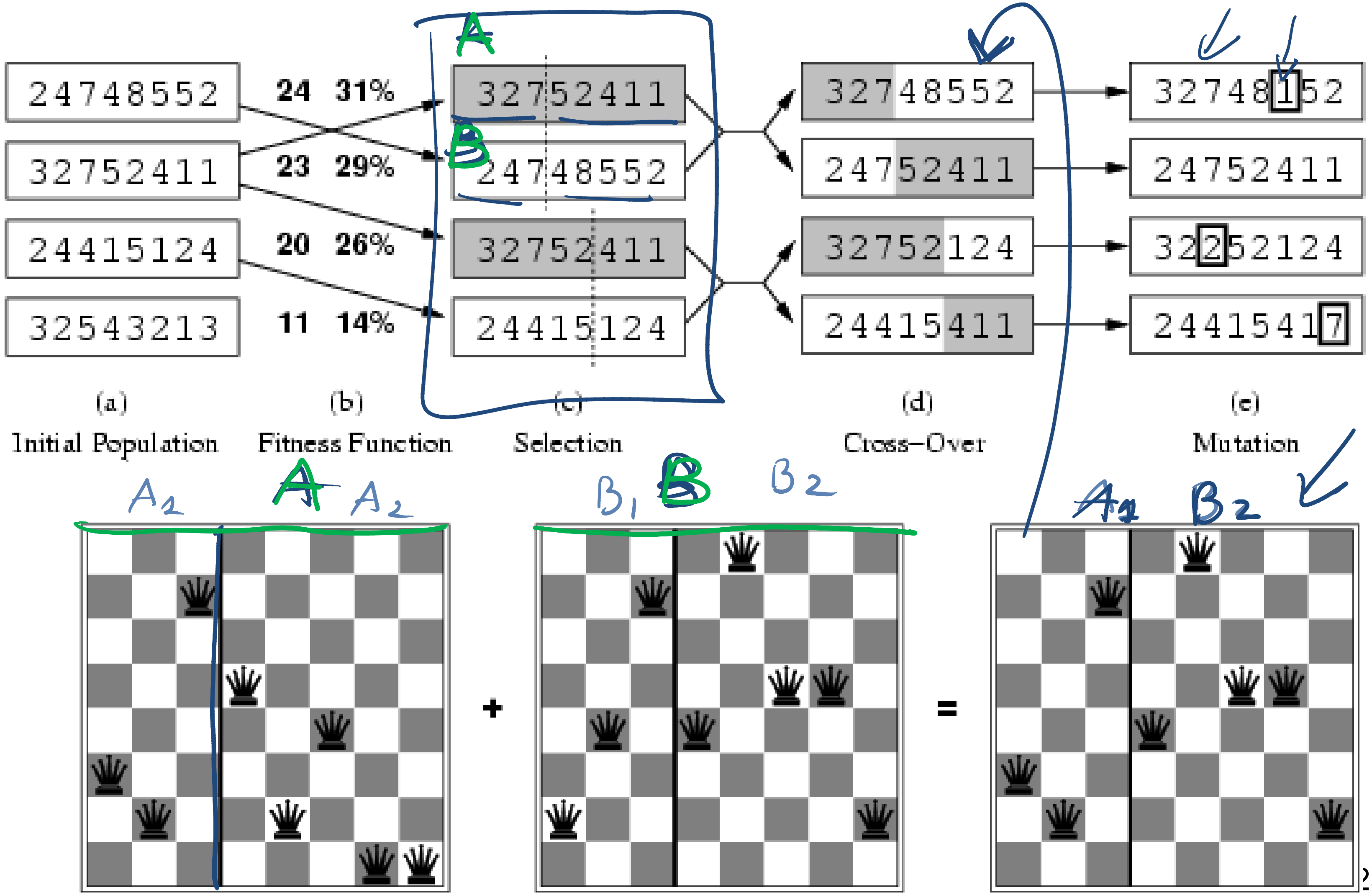
24/(24+23+20+11) = 31%

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

same as Beam Search 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!)

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		Environment: Deterministic	Environment: Stochastic
Static	Constraint Satisfaction	*Vars + Constraints* Arc Consistency Search → for CSP SLS	
Static	Query	*Logics* Search → for Inference	*Belief Nets* Var. Elimination
Sequential	Planning	*STRIPS* actions Search → for complex planning	*Decision Nets* Var. Elimination *Markov Processes* Value Iteration

Representation

Reasoning Technique

Assignment-2 on CSP will be out this evening (programming!)

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***)

322 Feedback ☺ or ☹

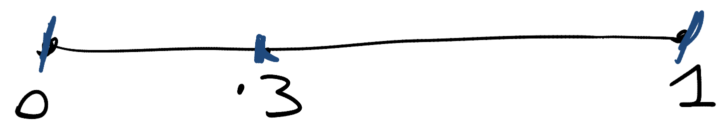
- Lectures
- Slides
- Practice Exercises
- Assignments
- AIspace
- ……
- Textbook
- Course Topics / Objectives
- TAs
- Learning Goals
- ……

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

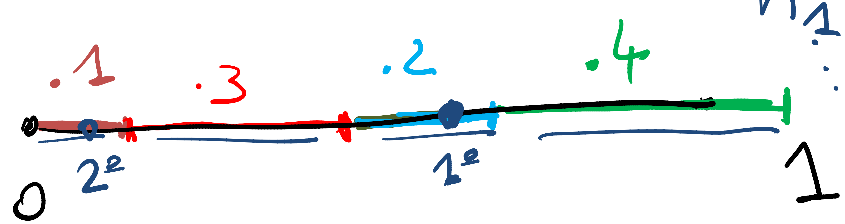
SAME HERE

→ n_1 $p_1 = .1$

→ n_2 $p_2 = .3$

→ n_3 $p_3 = .2$

→ n_4 $p_4 = .4$



n_3 first sample

n_1 second sample