Stochastic Local Search Variants

Computer Science cpsc322, Lecture 16

(Textbook Chpt 4.8)

February, 9, 2009

Lecture Overview

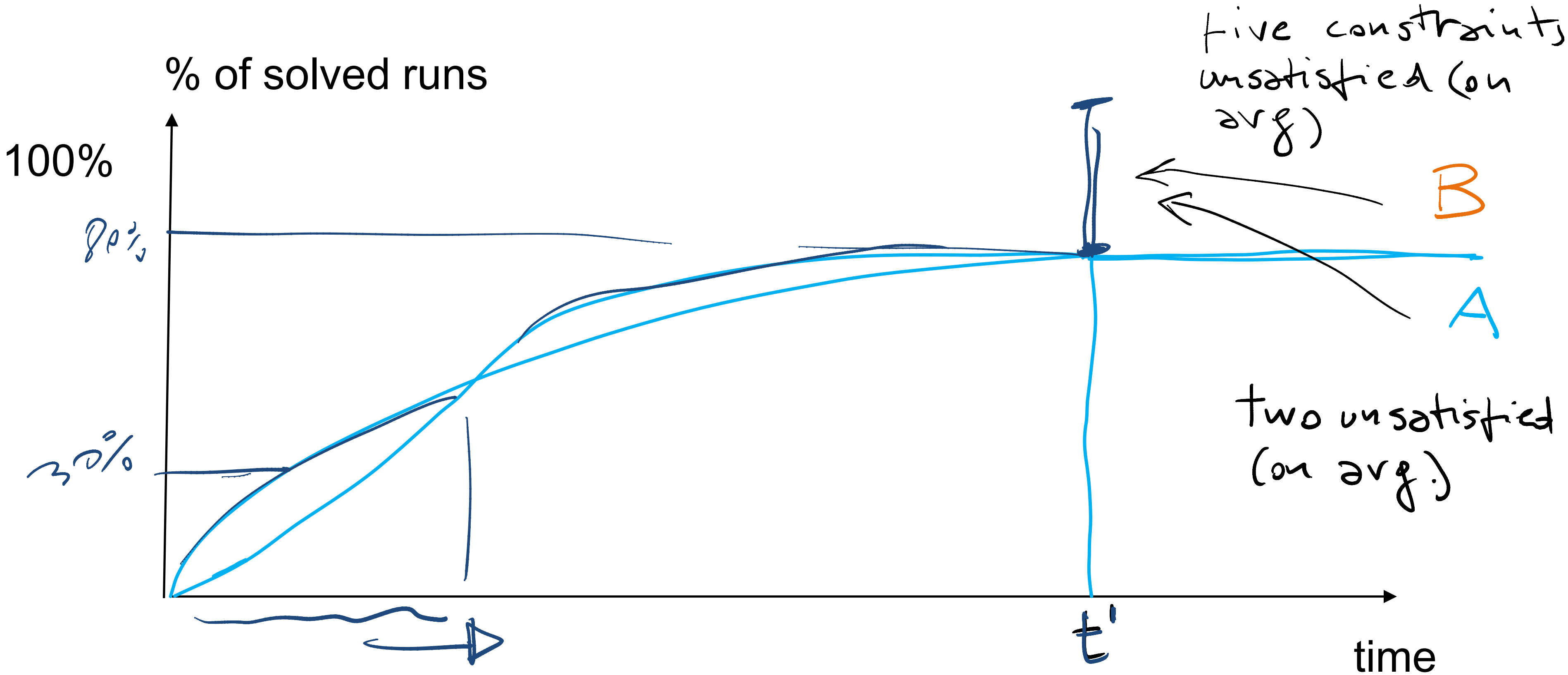
- **Recap SLS**
- SLS variants

Stochastic Local Search

min max

- **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. 9%
    - Random restart: reassign random values to all variables. 1%
  - Always keep best solution found so far
  - Stop when
    - Solution is found (in vanilla CSP ….satisfying all C……….)
    - Run out of time (return best solution so far)

Runtime Distributions



Which one would you use if you could wait t = t' ?

you should look at the quality of the answers on the unsolved problems. In pure CSP # of unsatisfied constraints

so use A

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…..

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., $h(n') > h(n)$, adopt it.
- If it isn't an improvement, adopt it probabilistically depending on the difference and a temperature parameter, *T*. $h(n) > h(n') \Rightarrow h(n') - h(n) < 0$
  - we move to *n'* with probability $e^{(h(n')-h(n))/T}$

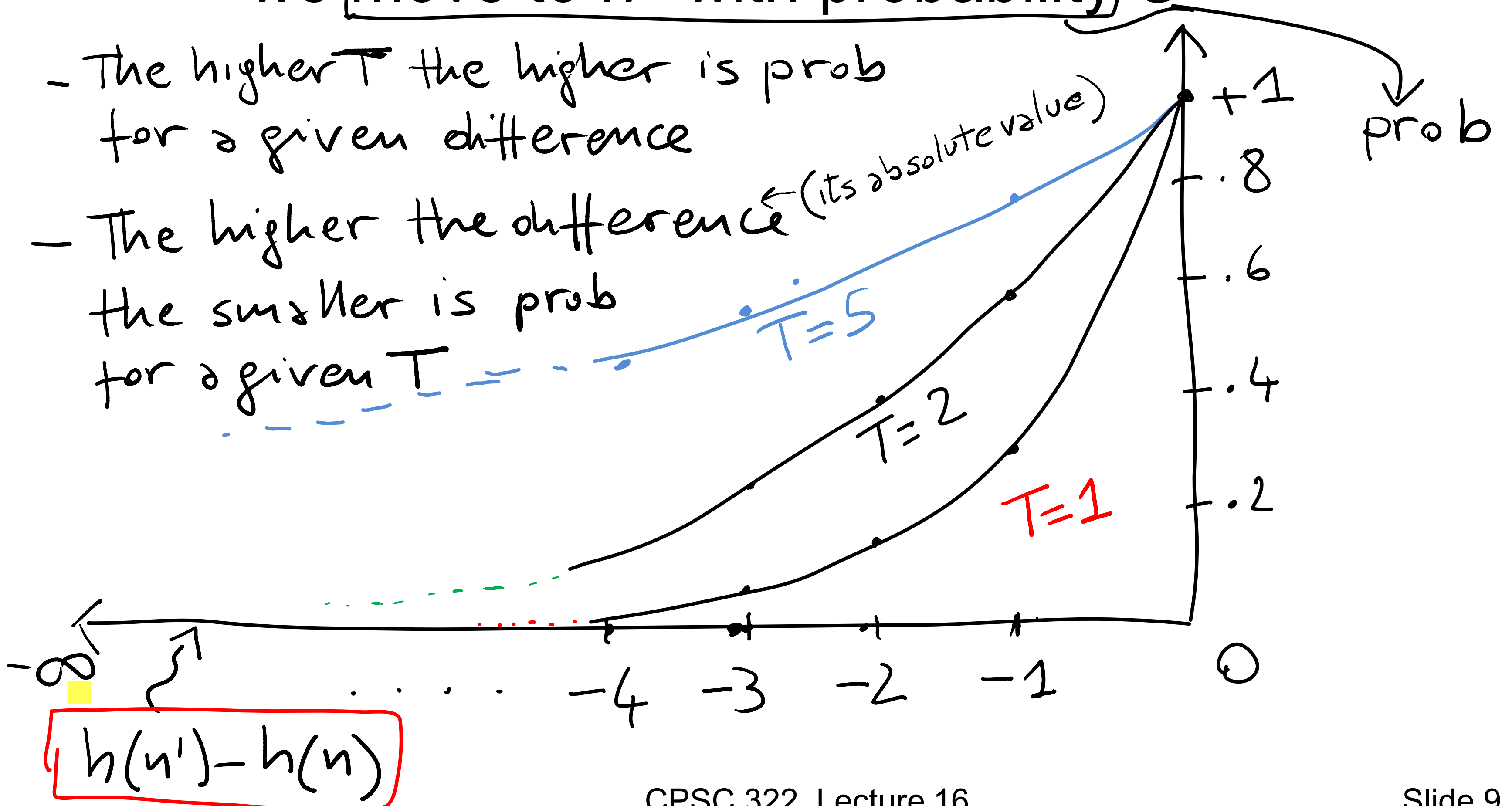
see next slide

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

$h(n) > h(n') \Rightarrow h(n') - h(n) < 0$

  - 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.

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

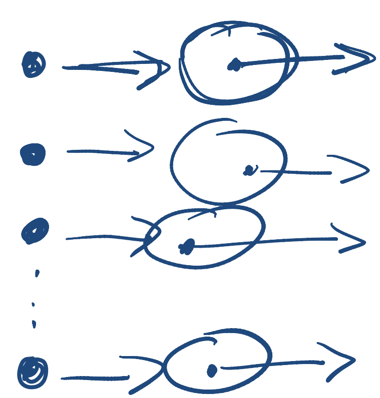
- At every stage, update your population.
- Whenever one individual is a solution, report it.

Simplest strategy: Parallel Search

- All searches are independent
- Like *k* restarts

but more memory :(

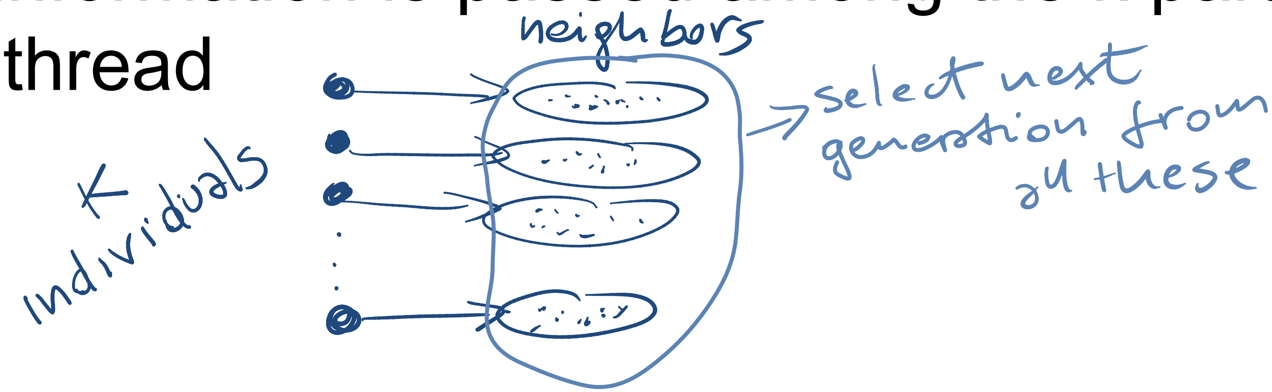
No reason to use it



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

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

h: scoring function

$$\text{Probability of selecting}(n_j) = \frac{h(n_j)}{\sum_{i=1}^{m} h(n_i)}$$

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)

Lecture Overview

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

Population Based SLS: Genetic Algorithms

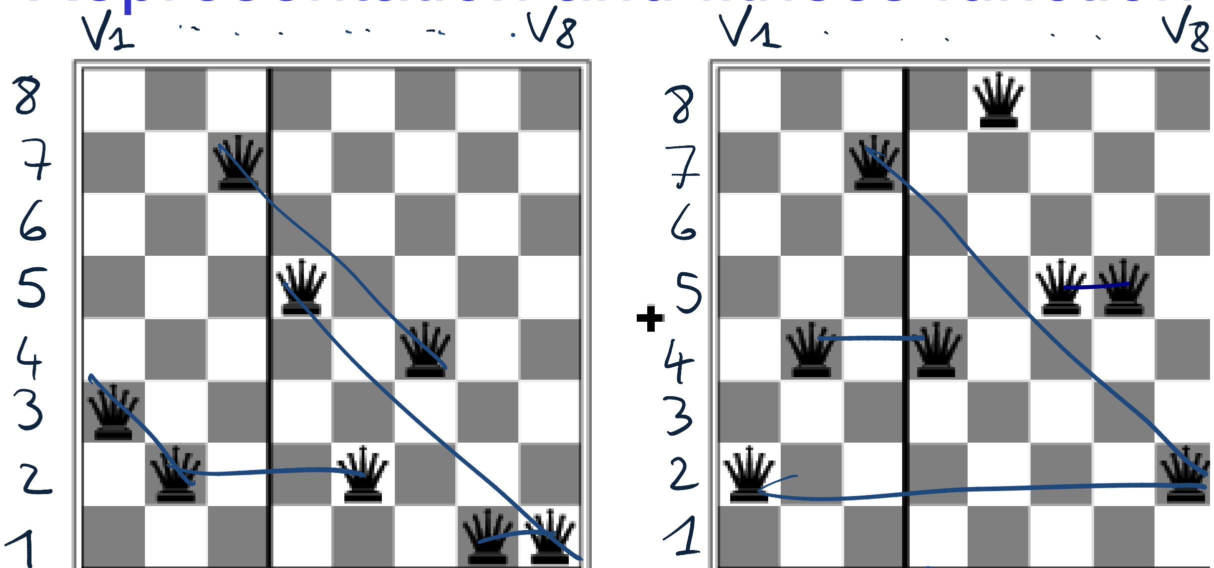
- Start with *k* randomly generated individuals (population)
- An individual is represented as a string over a finite alphabet (often a string of 0s and 1s)
- A successor is generated by combining two parent individuals (loosely analogous to how DNA is spliced in sexual reproduction)
- Evaluation/Scoring 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

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



(28-4)

24748552	24
32752411	23

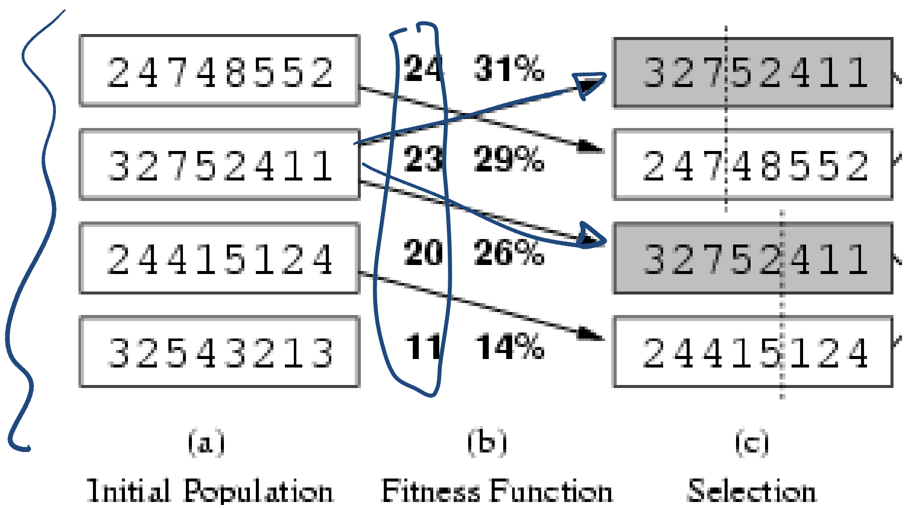
(28-5)

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



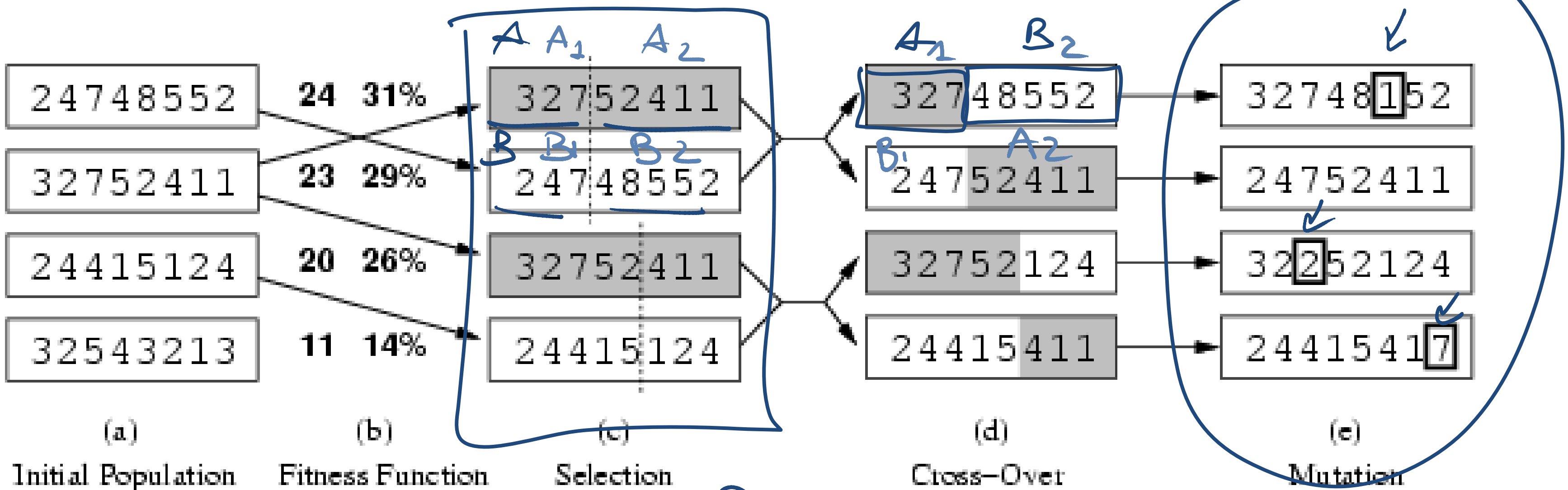
24/(24+23+20+11) = 31%

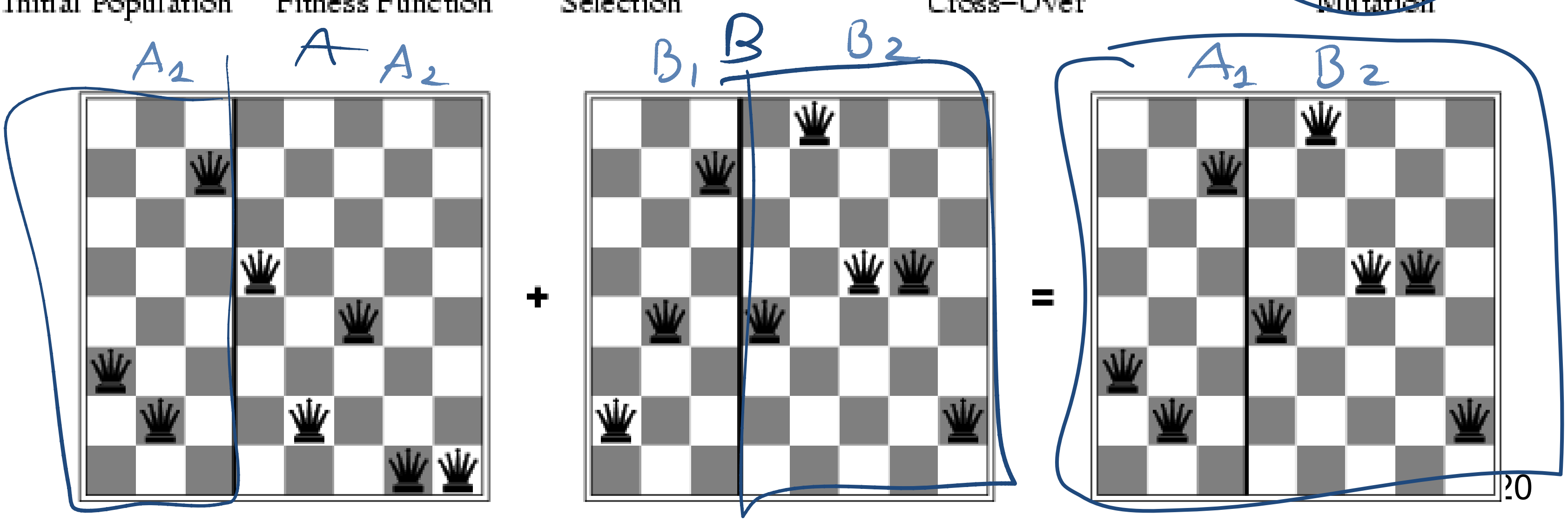
23/(24+23+20+11) = 29% etc

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

Modules we'll cover in this course: R&Rsys

Environment

Problem		Deterministic	Stochastic
Static	Constraint Satisfaction	*Vars + Constraints* Arc Consistency Search → for CSP SLS	
	Inference	*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

Next class

Start Planning (Chp 11)

Feedback summary

	☺	😐	☹	
• Assignments *(prog. , unclear)*	7	1	7	(0)
• TAs	0	0	1	(-1)
• Textbook	6	2	2	(+4)
• Lectures *(more interactive)*	5	5	1	(+4)
• Practice Exercises *(one per lecture)*	6	-	1	(+5)
• Course Topics	6	1	-	(+6)
• Learning Goals	6	-	-	(+6)
• Slides (*hard to read*)	10	1	3	(+7)
• AIspace	13	1	1	(+12)
• Exams… lots of sample questions				

What is coming next?

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

……………….

Systematically solving CSPs: Summary

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

    may or may not be a solution

- Apply Depth-First Search with Pruning
- Split the problem in a number of disjoint cases
  - Apply Arc Consistency to each case

CSPs summary

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

- Systematic Search approach (search space …..?)
  - Constraint network support n^2d^3
    - ✓ inference e.g., Arc Consistency (can tell you if solution does not exist)
    - ✓ Decomposition
  - 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