Artificial Intelligence

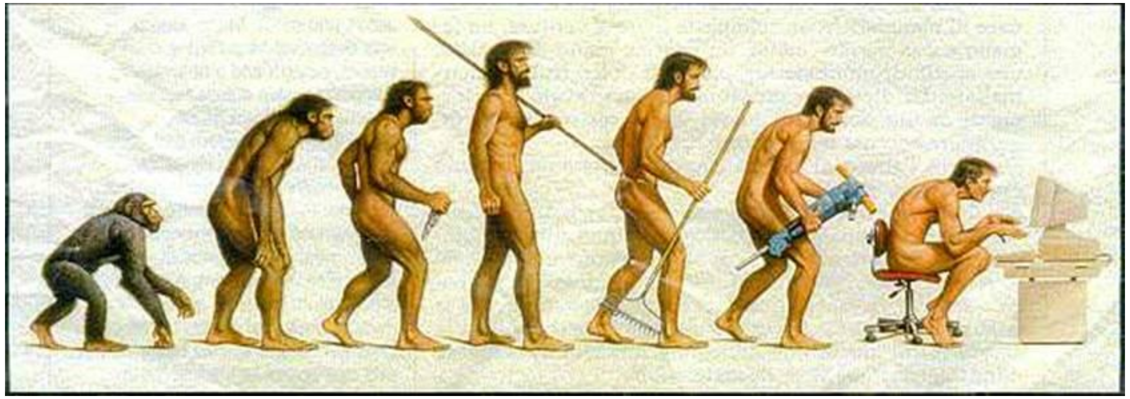
CS4365 --- Fall 2022

Local Search

Instructor: Yunhui Guo

Genetic Algorithms

- Approach mimics evolution



- Usually presented using a rich (and different) vocabulary:
  - fitness, populations, individuals, genes, crossover, mutations, etc.

- Still, can be viewed quite directly in terms of standard local search

Features of Evolution

- High degree of parallelism

- New individuals ("next state / neighboring states"):
  derived from "parents" ("crossover operation")
  genetic mutation

- Selection of next generation
  based on survival of the fittest

General Idea of Genetic Search

- Maintain a population of individuals (states / strings / candidate solutions)
  - Similar to local beam search

- Each individual is evaluated using a fitness function, i.e., an evaluation function. The fitness scores force individuals to compete for the privilege of survival and reproduction.
  - Similar to hill climbing

General Idea of Genetic Search

- Generate a sequence of generations:
  - From the current generation, select pairs of individuals (based on fitness) to generate new individuals, using crossover
- Introduce some noise through random mutations
- Hope that average and maximum fitness (i.e., value to be optimized) increases over time

Genetic Algorithms as Search

- Genetic algorithms are local heuristic search algorithms.

- Especially good for problems that have large and poorly understood search spaces

- Genetic algorithms use a randomized parallel beam search to explore the state space

- You must be able to define a good fitness function, and of course, a good state representation

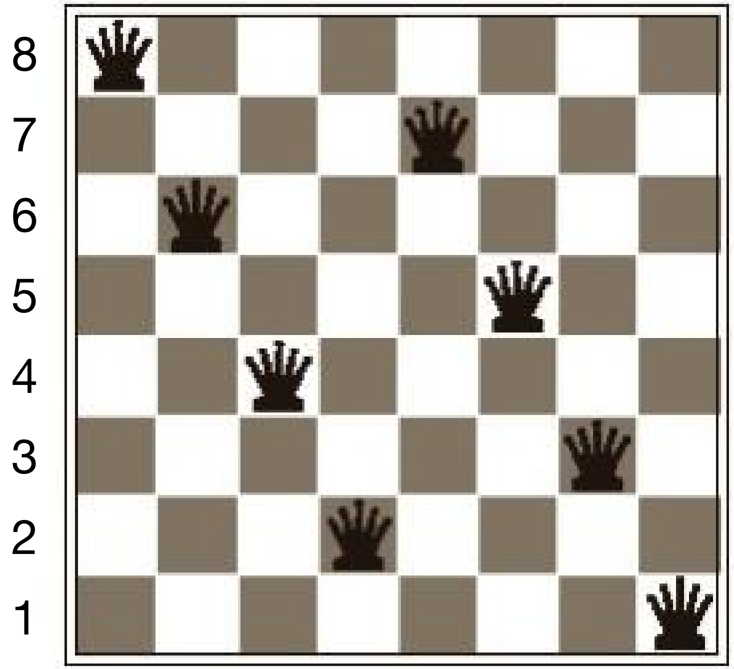
Binary String Representation

- Individuals are usually represented as a string over a finite alphabet, usually bit strings.

- Individuals represented can be arbitrarily complex.
  - E.g. each component of the state description is allocated a specific portion of the string, which encodes the values that are acceptable.

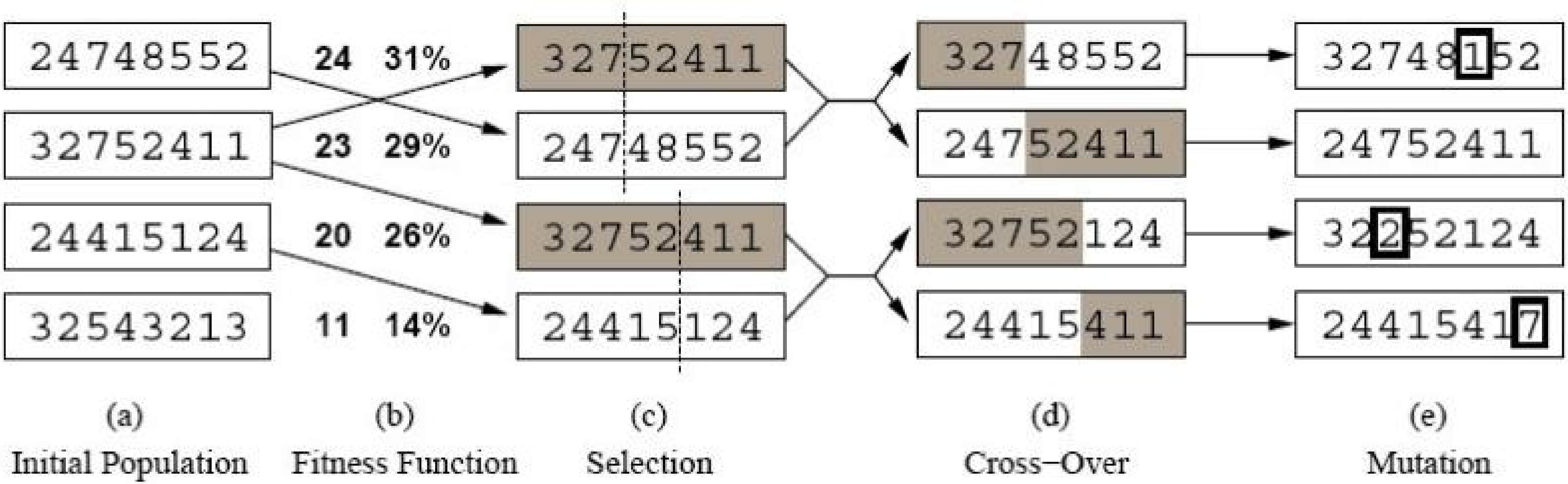
- Bit string representation allows crossover operation to change multiple values in the state description. Crossover and mutation can also produce previously unseen values.

8-queens State Representation



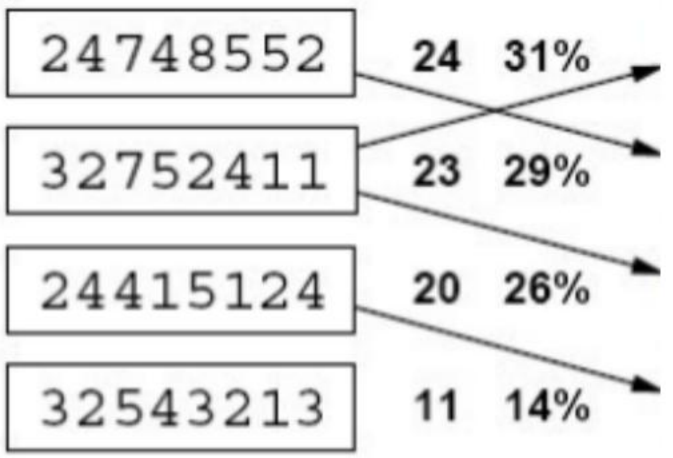
- Option 1: 86427531
- Option 2: 111 101 011 001 110 100 010 000

GA: High-level Algorithm



Selecting the Most Fit Individuals

- Individuals are chosen probabilistically for survival and crossover based on fitness proportionate selection:



(a) Initial Population (b) Fitness Function

$$\Pr(i) = \frac{Fitness(i)}{\sum_{j=1}^{p} Fitness(i_j)}$$

Other Selection Methods

- Tournament Selection:
  - Two individuals selected at random. With probability p, the more fit of the two is selected. With probability (1 – p), the less fit is selected.
- Rank Selection:
  - The individuals are sorted by fitness and the probability of selecting an individual is proportional to its rank in the list.

Crossover Operators

- Single-point crossover:

Parent A:	1	0	0	1	0	1	1	1	0	1
Parent B:	0	1	0	1	1	1	0	1	1	0

Crossover Operators

- Single-point crossover:

Parent A:	1	0	0	1	0	1	1	1	0	1
Parent B:	0	1	0	1	1	1	0	1	1	0

Crossover Operators

- Single-point crossover:

Parent A:	1	0	0	1	0	1	1	1	0	1
Parent B:	0	1	0	1	1	1	0	1	1	0

Crossover Operators

- Single-point crossover:

Parent A:	1	0	0	1	0	1	1	1	0	1
Parent B:	0	1	0	1	1	1	0	1	1	0
Child AB:	1	0	0	1	0	1	0	1	1	0

Crossover Operators

- Single-point crossover:

Parent A:	1	0	0	1	0	1	1	1	0	1
Parent B:	0	1	0	1	1	1	0	1	1	0

Child AB:	1	0	0	1	0	1	0	1	1	0
Child BA:	0	1	0	1	1	1	1	1	0	1

Crossover Operators

- Two-point crossover:

Parent A:	1	0	0	1	0	1	1	1	0	1
Parent B:	0	1	0	1	1	1	0	1	1	0

Crossover Operators

- Two-point crossover:

Parent A:	1	0	0	1	0	1	1	1	0	1
Parent B:	0	1	0	1	1	1	0	1	1	0

Crossover Operators

- Two-point crossover:

Parent A:	1	0	0	1	0	1	1	1	0	1
Parent B:	0	1	0	1	1	1	0	1	1	0

Crossover Operators

- Two-point crossover:

Parent A:	1	0	0	1	0	1	1	1	0	1
Parent B:	0	1	0	1	1	1	0	1	1	0
Child AB:	1	0	0	1	1	1	0	1	0	1

Crossover Operators

- Two-point crossover:

Parent A:	1	0	0	1	0	1	1	1	0	1
Parent B:	0	1	0	1	1	1	0	1	1	0
Child AB:	1	0	0	1	1	1	0	1	0	1
Child BA:	0	1	0	1	0	1	1	1	1	0

Crossover Operators

- Uniform crossover:

Parent A:	1	0	0	1	0	1	1	1	0	1
Parent B:	0	1	0	1	1	1	0	1	1	0

Crossover Operators

- Uniform crossover:

Parent A:	1	0	0	1	0	1	1	1	0	1
Parent B:	0	1	0	1	1	1	0	1	1	0

Crossover Operators

- Uniform crossover:

Parent A:	1	0	0	1	0	1	1	1	0	1
Parent B:	0	1	0	1	1	1	0	1	1	0

Crossover Operators

- Uniform crossover:

Parent A:	1	0	0	1	0	1	1	1	0	1
Parent B:	0	1	0	1	1	1	0	1	1	0
Child AB:	1	1	0	1	1	1	1	1	0	1

Crossover Operators

- Uniform crossover:

Parent A:	1	0	0	1	0	1	1	1	0	1
Parent B:	0	1	0	1	1	1	0	1	1	0
Child AB:	1	1	0	1	1	1	1	1	0	1
Child BA:	0	0	0	1	0	1	0	1	1	0

Mutation

- Mutation: randomly flip one bit

Individual A: 1 0 0 1 0 1 1 1 0 1

Mutation

- Mutation: randomly flip one bit

Individual A: 1 0 0 1 0 1 1 1 0 1

Mutation

- Mutation: randomly flip one bit

Individual A: 1 0 0 1 0 1 1 1 0 1

Individual A': 1 1 0 0 0 1 1 1 0 1

Mutation

- The mutation operator introduces random variations, allowing solutions to jump to different parts of the search space.

- What happens if the mutation rate is too low?
- What happens if the mutation rate is too high?

- A common strategy is to use a high mutation rate when search begins
- But to decrease the mutation rate as the search progresses.

GA: High-level Algorithm

GA(Fitness, Fitness theshold, p, r, m)

P ← randomly generate p individuals

For each i in P, compute Fitness(i)

While [max Fitness(i) < Fitness theshold]

1. Probabilistically select $(1 - r)p$ members of P to add to Ps
2. Probabilistically choose $r \cdot p / 2$ pairs of individuals from P. For each pair, $\langle i_1, i_2 \rangle$, apply crossover and add the offspring to Ps
3. Mutate $m \cdot p$ random members of Ps
4. $P \leftarrow P_S$
5. For each i in P, compute Fitness(i)

Return the individual in P with the highest fitness.

Example

- Maximize the function $f(x) = x^3$, where $x \in [0, 15]$

Step 1: Choose an encoding

$$0 \rightarrow 0000 \qquad 15 \rightarrow 1111$$

Example

- Maximize the function $f(x) = x^3$, where $x \in [0, 15]$

Step 2: Randomly choose an initial population

4, 7, 9, 12

Example

- Maximize the function $f(x) = x^2$, where $x \in [0, 15]$

Step 3: Select parents

		Fitness	Pr(i)	Selection
4	0100	16	16/290	0111
7	0111	49	49/290	1001
9	1001	81	81/290	1001
12	1100	144	144/290	1100
		290		

Example

- Maximize the function $f(x) = x^2$, where $x \in [0, 15]$

Step 4: Crossover and Mutation

0111	$0101 \rightarrow 01\underline{1}1$
1001	$1011 \rightarrow 101\underline{0}$
1001	$1000 \rightarrow 1\underline{1}00$
1100	$1101 \rightarrow 110\underline{0}$

Example

- Maximize the function $f(x) = x^2$, where $x \in [0, 15]$

Step 5: Evalute the offspring

Before	Current
4	0111 : 7
7	1010 : 10
9	1100 : 12
12	1100 : 12

How Does Genetic Algorithm Work?

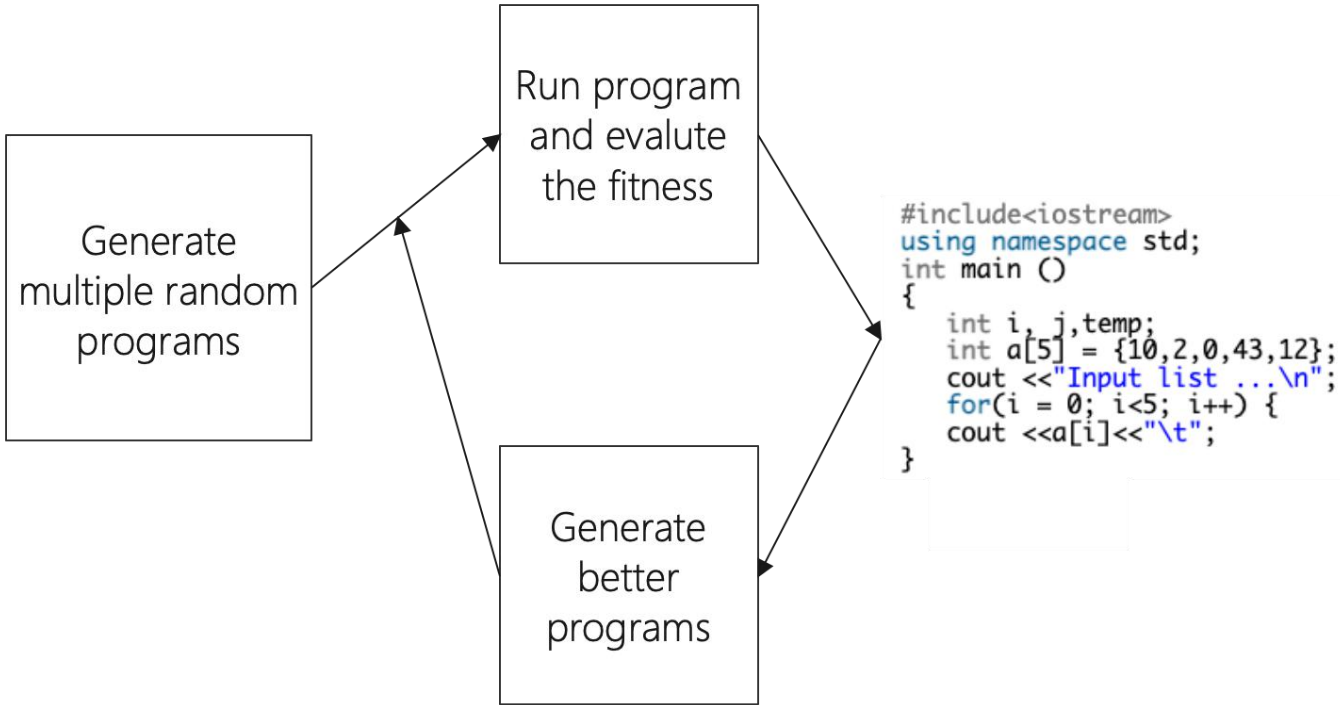
- Based on the idea of schema
  - For example, 246***** is a schema for 8-queen problem
  - 11** is a schema for the optimization problem
- Each schema can be thought as a building block of the final solution
- If the average fitness of the instances of a schema is above the mean, then the number of instances of the schema within the population will grow over time

Limitations of GA

- It is expensive to evaluate the fitness function
- The stopping criterion is not clear
- Can get stuck in local minimum

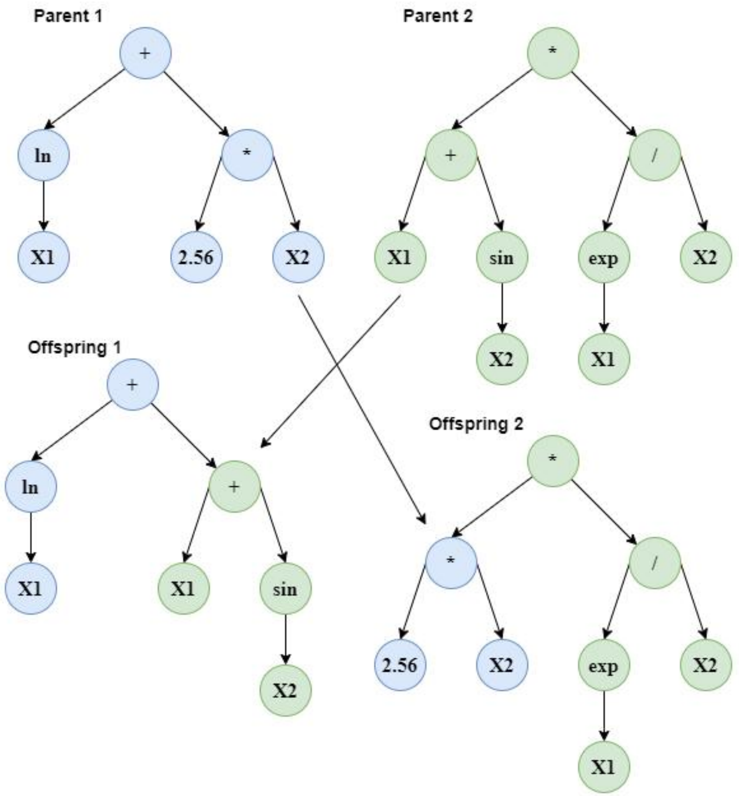
Genetic Programming

- In Genetic Programming, programs are evolved instead of bit strings.
- We want to find the program that completes the task we have.



Genetic Programming

- In Genetic Programming, programs are evolved instead of bit strings.



https://towardsdatascience.com/unit-4-genetic-programming-d80cd12c454f

Remarks on GA's

- In practice, several 100 to 1000's of strings. Value of crossover difficult to determine (so far).

- Crowding can occur when an individual that is much more fit than others reproduces like crazy, which reduces diversity in the population.

- In general, GA's are highly sensitive to the representation

Local Search --- Summary

- Surprising efficient search method.
- Wide range of applications
  - any type of optimziation / search task
- Handles search spaces that are too large
  - (e.g., 10^1000) for systematic search
- Often best available algorithm when lack of global information.
- Formal properties remain largely elusive. Research area will most likely continus to thrive