Evolutionary Algorithms

CSCI 3100

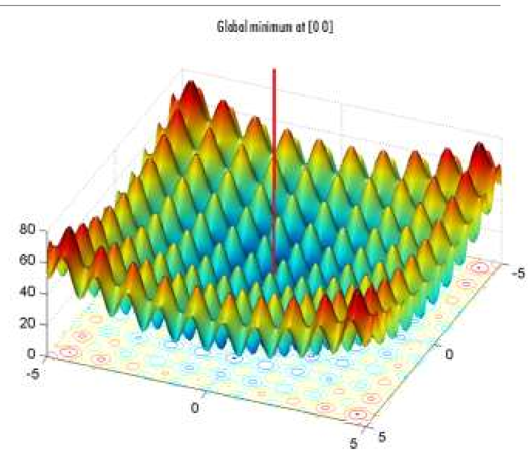
Motivation

Some problems do not have an efficient algorithm

Some problems do not have an approximation algorithm

What if we still need to solve them?

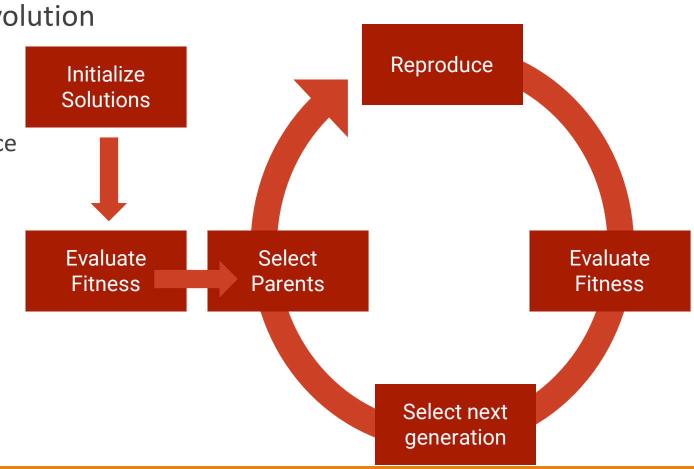
Solution space can be very complex



Evolutionary Algorithm Overview

Motivated by Darwin’s Theory of Evolution

Key idea:
- Generate candidate solution(s)
- Test solutions(s) on the problem instance
- Improve solutions and re-test
- Quit when
  - You are satisfied with solution quality
  - No improvements are made for a while
  - Run out of time



Initializing candidate solutions

Need to select how to represent each solution

Representation depends on the problem

Example problems:
- Minimize/maximize a function $f(x_1, x_2, \ldots, x_n)$, where $x_i \in \mathbb{R}$
- TSP: Minimize the weight of the cycle that visits every vertex in a graph
- 3-SAT: maximize the number of clauses that are satisfied

In either case, a candidate solution is a vector of values (real numbers, vertex indices, boolean values).

Initialize a “population” of candidate solutions using
- Random number generator
- Non-random strategy

How many candidate solutions should we generate?
- Parameter of the algorithm.

Suppose we are solving the following 3-SAT problem:
$(x_1+x_2+x_3')\wedge(x_1'+x_2+x_4)\wedge(x_1+x_3'+x_4')\wedge(x_2+x_3'+x_4)$

We can represent a solution as a vector of 4 booleans.
What does the following candidate solution [0, 1, 1, 0] mean?

A. x_1=false, x_2=true, x_3=true, x_4=false

B. x_1=true, x_2=false, x_3=false, x_4=true

C. x_1=false, x_2=true, x_3'=true, x_4 = false

D. $(x_1+x_2+x_3')$ is false
$(x_1'+x_2+x_4)$ is true
$(x_1+x_3'+x_4')$ is true
$(x_2+x_3'+x_4)$ is false

Evaluate Fitness

Fitness – a measure of the solution quality

High fitness – good solution, low fitness – bad solution

We don't know what the optimal fitness is

How can we tell if the fitness of a candidate solution is "high" or "low"?

Compare the fitness of all candidate solutions

Fitness is a relative measure

Fitness examples

Minimize/maximize a function $f(x_1, x_2, \ldots, x_n)$, where $x_i \in \mathbb{R}$

- Fitness: $f(x_1, x_2, \ldots, x_n)$ if maximization problem
- Fitness: $-f(x_1, x_2, \ldots, x_n)$ if minimization problem

TSP: Minimize the weight of the cycle that visits every vertex in a graph

- Given a cycle that visits every vertex, fitness is 1-(total weight of the cycle).

3-SAT: Maximize the number of clauses that are satisfied

- Given assignment of variables, fitness is the number of clauses this assignment satisfies

Suppose we are solving the following 3-SAT problem: $(x_1+x_2+x_3')$^$(x_1'+x_2+x_4)$^$(x_1+x_3'+x_4')$^$(x_2+x_3'+x_4)$

Fitness of the candidate solution S is the number of clauses S satisfies. What is the fitness of S=[1,0,1,0]?

A. 1

B. 2

C. 3

D. 4

E. 5

Creating new solutions

Random solutions are not very good

Evolutionary concept – survival of the fittest

Evolutionary pressure
- Solutions with higher fitness are used as the starting point of new solutions
- Solutions with higher fitness are more likely to survive

Select Parents

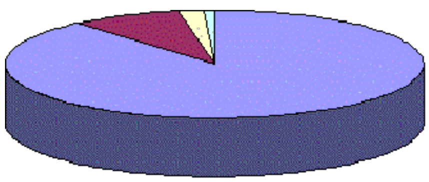
Use fitness to determine which candidate solutions will be the starting point for new solutions

Fitness proportional selection
- Probability of S_i getting selected is $\frac{f(S_i)}{\sum_{j=1}^{N} f(S_j)}$

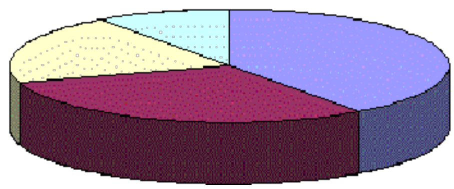
Tournament selection
- Select k individuals from the population at random
- The best individual in this group “wins” the tournament with probability p
- The second best individual “wins” the tournament with probability p*(1-p)
- The third best individual “wins” the tournament with probability $p*(1-p)^2$
- p is a parameter of an algorithm

Rank selection
- Rank all individuals according to their fitness
- Probability of an individual getting selected is proportional to its rank

We have 4 solutions with fitness values: 90, 5, 3, 2. Charts below represent probability of each individual being selected using two different selection methods. Which chart represents rank selection?



A



B

Reproduction: Recombination + Mutation

Recombination

Crossover – take one part of the solution from parent 1 and another part from parent 2

One point crossover:

- Select an index into a solution vector: k
- Copy values 1 through k from parent 1, copy the rest of the values from parent 2
- Copy values 1 through k from parent 2, copy the rest of the values from parent 1

Example:

- Maximize function f(x1, x2, x3, x4)
- Parent 1 = [3.2, 1, 4, 2]
- Parent 2 = [5, 1.2, 2.3, 1]
- Select k = 3
- Offspring 1 = [3.2, 1, 4, 1] Offspring 2 = [5, 1.2, 2.3, 2]

Suppose we have the following two solutions participating in crossover: [0, 1, 1, 1] and [1, 0, 1, 0]. Our crossover point k is 2. What are the two offspring produced by this crossover?

A. [0, 1, 1, 1] and [1, 0, 1, 0]

B. [0, 0, 0, 0] and [1, 1, 1, 1]

C. [0, 1, 1, 0] and [1, 0, 1, 1]

D. [0, 1, 1, 0] and [1, 1, 1, 0]

E. Need to know what problem this is solving to determine the answer.

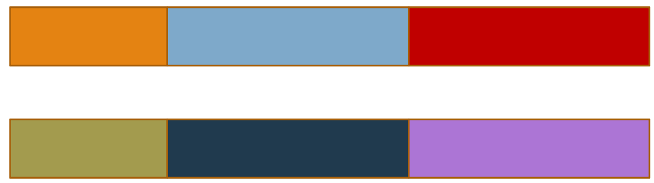
Two-point crossover

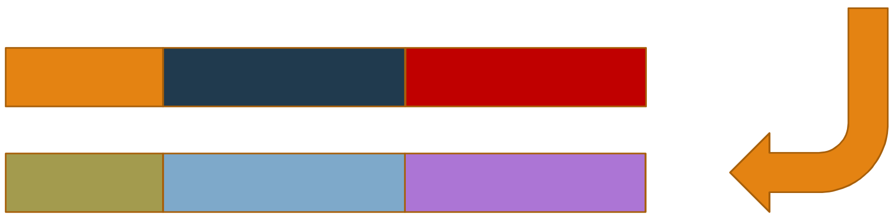
Select two indices: k and j

Pick values 1 through k from parent 1

Pick values k+1 through j from parent 2

Pick the rest from parent 1





Mutation

Each new solution can be randomly "mutated" with probability p_m
- Parameter of the algorithm

Mutation – randomly changing a value in the solution vector

Examples:
- 3-SAT: randomly select a value to "flip"(if value was 0, make it 1; if value was 1, make it 0)
- TSP: randomly swap a pair of vertices in a solution vector.

Generic Evolutionary Algorithm

```
BEGIN
  INITIALISE population with random candidate solutions;
  EVALUATE each candidate;
  REPEAT UNTIL ( TERMINATION CONDITION is satisfied ) DO
    1 SELECT parents;
    2 RECOMBINE pairs of parents;
    3 MUTATE the resulting offspring;
    4 EVALUATE new candidates;
    5 SELECT individuals for the next generation;
  OD
END
```

What do you think is the most time consuming step of the loop?

A. 1

B. 2

C. 3

D. 4

E. 5