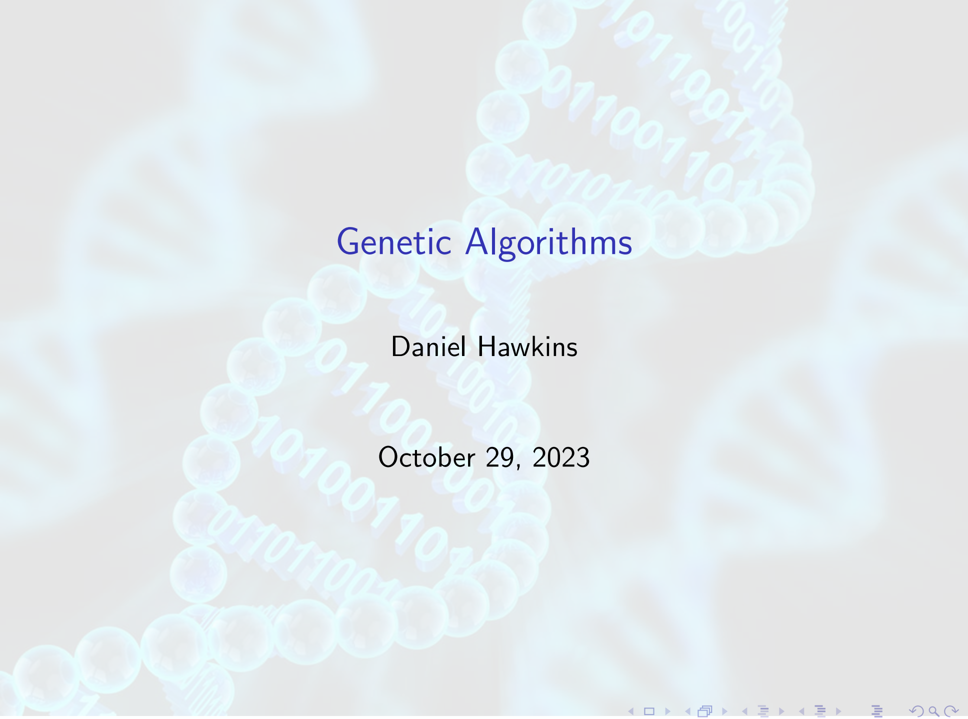


Genetic Algorithms

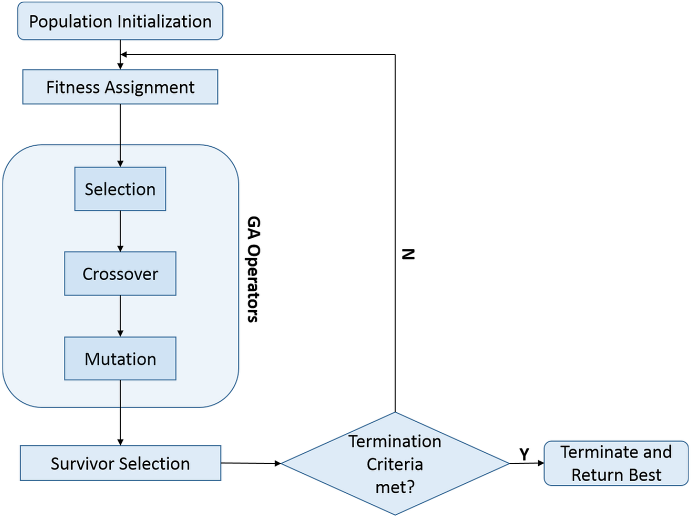
Daniel Hawkins

October 29, 2023

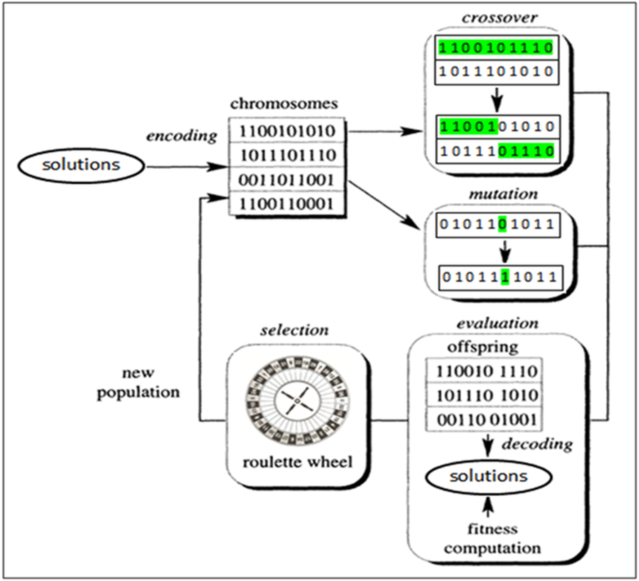
Genetic Algorithm: Overview

- Formalized and popularized in the 1960s-70s by John Holland, inspired by natural selection.
- Combines Darwin's "survival of the fittest" with selection, crossover, and mutation to solve optimization and search problems.
- Used in diverse fields like finance, game design, robotics, and drug discovery.
- Highly adaptable to different types of problems and data.
- Starts with a population of solutions, iterating and improving over generations.
- Ability to search large solution spaces, avoiding local optima.

Genetic Algorithm Workflow



Genetic Algorithm Workflow



Initialize Population

```
[0,1,1,0,0,0,1,1,0,1]

[1,1,1,1,0,1,1,0,1,1]

[1,0,1,1,1,1,1,0,1,0]

[0,0,0,0,0,1,0,0,1,0]
```

```
function initPopulation(popSize, chromLen)
    population = []
    for i = 1 to popSize
        chrom = []
        for j = 1 to chromLen
            bit = randomChoice([0, 1])
            chrom = chrom + [bit]
        population = population + [chrom]
    return population
```

Runtime: O(popSize * chromLen)

Calculate Fitness

```
fitness([0,1,1,0,0,0,1,1,0,1]) = 5

fitness([1,1,1,1,0,1,1,0,1,1]) = 8
```

```
function fitness(chrom)
    return sum(chrom)
```

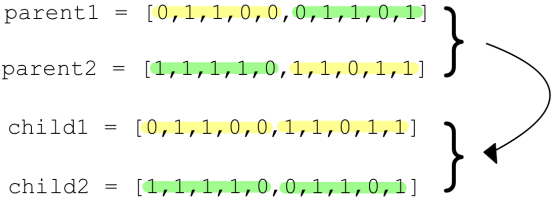
Run Time: O(chromLen)

Parent Selection

```
function selectParents(population)
    if length(population) == 1
        return population[0], population[0]
    elif length(population) == 2
        return population[0], population[1]
    parent1 = randomChoice(population[0:⌊length(population)/2⌋])
    parent2 = randomChoice(population[0:⌊length(population)/2⌋])
    return parent1, parent2
```

Run Time: O(1)

Crossover

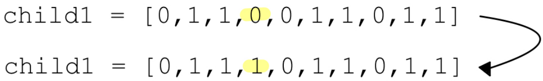


```
parent1 = [0,1,1,0,0,0,1,1,0,1]
parent2 = [1,1,1,1,0,1,1,0,1,1]
child1  = [0,1,1,0,0,1,1,0,1,1]
child2  = [1,1,1,1,0,0,1,1,0,1]
```

```
function crossover(parent1, parent2)
    point = randomInt(1, length(parent1) - 1)
    child1 = sublist(parent1, 0, point) +
             sublist(parent2, point, length(parent2))
    child2 = sublist(parent2, 0, point) +
             sublist(parent1, point, length(parent1))
    return child1, child2
```

Run Time: O(chromLen)

Mutation



```
child1 = [0,1,1,0,0,1,1,0,1,1]

child1 = [0,1,1,1,0,1,1,0,1,1]
```

```
function mutate(chrom, mutRate)
    newChrom = []
    for i = 1 to length(chrom)
        if randomFloat(0, 1) > mutRate
            newChrom = newChrom + [chrom[i]]
        else
            if chrom[i] == 0
                newChrom = newChrom + [1]
            else
                newChrom = newChrom + [0]
```

Run Time: O(chromLen)

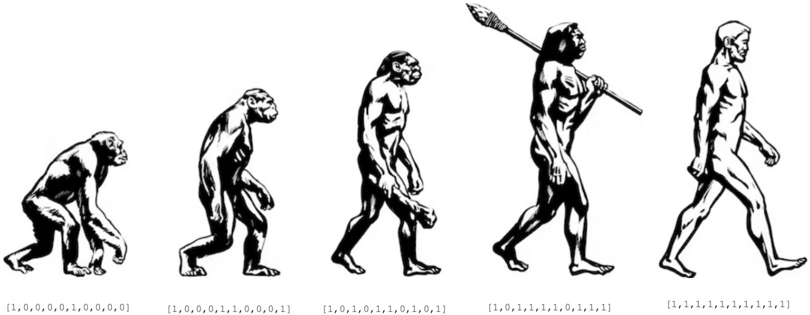
Main Algorithm

```
function geneticAlgo(numGen, popSize, chromLen, mutRate)
    population = initPopulation(popSize, chromLen)
    for i = 1 to numGen
        population = sort(population, key=fitness)
        if fitness(population[0]) == chromLen
          return population[0], i + 1
        newPop = []
        while length(newPop) < popSize
            parent1, parent2 = selectParents(population)
            child1, child2 = crossover(parent1, parent2)
            child1 = mutate(child1, mutRate)
            child2 = mutate(child2, mutRate)
            newPop = newPop + [child1, child2]
        population = newPop
    return max(population, key=fitness)
```

Run Time:

O(numGen*popSize*chromLen+numGen*popSize*log(popSize))

Conclusion



What would you like to optimize through evolution?