Genetic Algorithms

Dr. Mahmoud Nabil Mahmoud
mnmahmoud@ncat.edu

North Carolina A & T State University

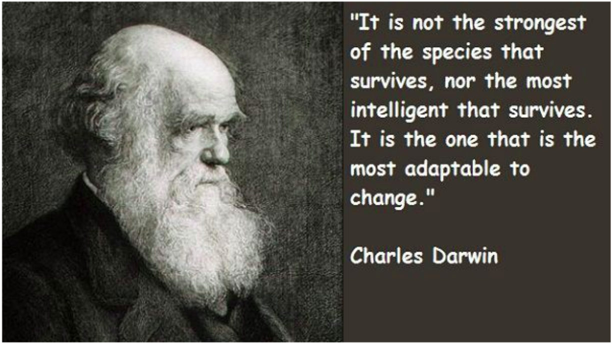
September 6, 2021

Agenda

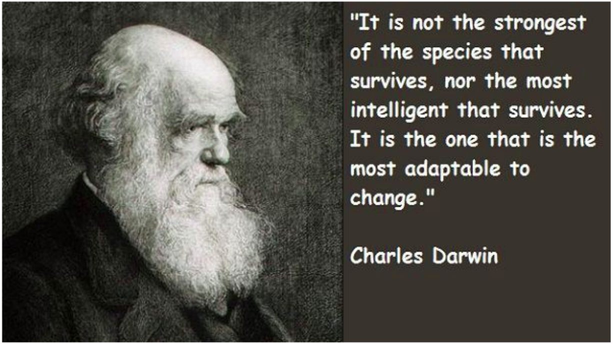
1. What are Genetic Algorithms
2. Mechanics of Genetic Algorithm
3. GA Walkthrough example
4. GA behavior

Outline

Charles Darwin



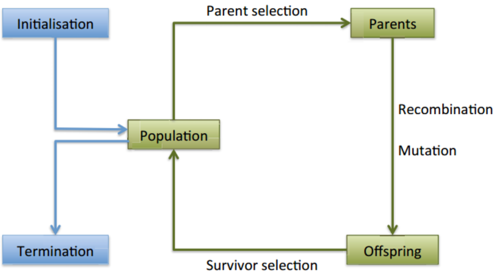
Charles Darwin



Survival of the fittest

GA Metaphor

Evolution		Problem solving
Environment	$\longleftrightarrow$	Problem
Individual	$\longleftrightarrow$	Candidate solution
Fitness	$\longleftrightarrow$	Quality



What are Genetic Algorithms?

Genetic Algorithms

GA are search algorithms based on the mechanics of natural selection and natural genetics.

- Developed by John Holland in 1970s at the University of Michigan.
- Holland Goal was to
  - Design Artificial systems that imitate mechanisms in natural systems.
  - Explain the internal processes of natural systems.

Central theme of GA is robustness, and the balance between efficiency and efficacy.

GA Design Principles

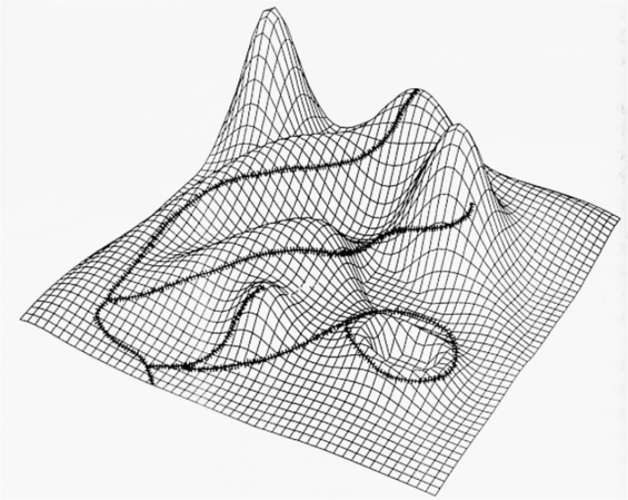
- **Robustness.** Ability to adapt to different conditions and problems seamlessly.
  - Costly redesigns can be reduced.
  - Systems can perform their functions for longer.
  - GA are not restrictive to assumptions concerning: continuity, existence of derivative, uni-modality, etc.
- **Efficacy.** Getting the task done.
- **Efficiency.** Getting the task done with good performance.

GA are theoretically and empirically proven to provide robust search in complex spaces

GA Applications

- Criminal- Likeness Reconstruction
- Trip, Traffic and Shipment Routing
- Evolvable Hardware
- Automotive Design
- Encryption and Code Breaking
- Finance and Investment Strategies
- ....

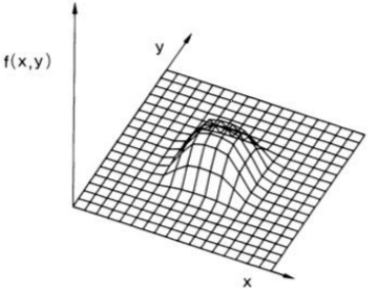
Searching for good solutions on a rough landscape



Robustness of Traditional Optimization Methods

Existing main optimization methods are: calculus, enumerative, and random search.

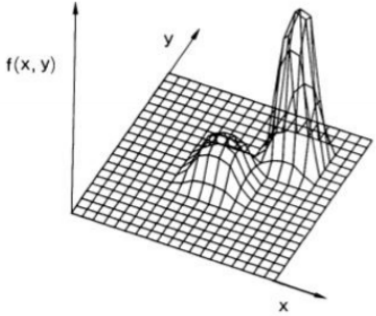
- **Calculus-based methods.**
    - **Indirect.** Get the gradient and set it equal to zero.
    - **Direct.** Also known as hill climbing where you start from random position and move in the direction related to the local gradient.



Robustness of Traditional Optimization Methods

Problem with calculus-based methods

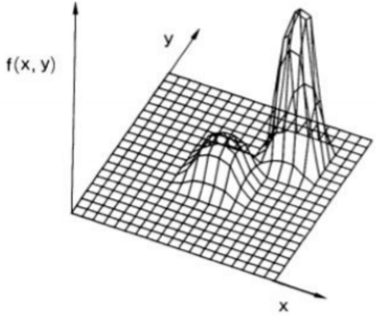
- Locality in scope.
- Rely on the existence of the gradient (well defined slope values).



Robustness of Traditional Optimization Methods

Problem with calculus-based methods

- Locality in scope.
- Rely on the existence of the gradient (well defined slope values).



Not robust due to the assumption of continuity, uni-modality, the gradient existence

Robustness of Tradiitional Optimization Methods

Existing main optimization methods

- **Enumerative schemes.** simply scan every point in the search space by evaluating the objective function.

Robustness of Tradiitional Optimization Methods

Existing main optimization methods

- **Enumerative schemes.** simply scan every point in the search space by evaluating the objective function.
    - Not practical for infinite search spaces.

Robustness of Tradiitional Optimization Methods

Existing main optimization methods

- **Enumerative schemes.** simply scan every point in the search space by evaluating the objective function.
  - Not practical for infinite search spaces.
  - Lack the efficiency.

Robustness of Tradiitional Optimization Methods

Existing main optimization methods

- **Enumerative schemes.** simply scan every point in the search space by evaluating the objective function.
  - Not practical for infinite search spaces.
  - Lack the efficiency.
- **Dynamic Programming.** is enumerative method.

Robustness of Tradiitional Optimization Methods

Existing main optimization methods

- **Enumerative schemes.** simply scan every point in the search space by evaluating the objective function.
  - Not practical for infinite search spaces.
  - Lack the efficiency.
- **Dynamic Programming.** is enumerative method.
  - Suffer from curse of dimensionality.

Robustness of Tradiitional Optimization Methods

Existing main optimization methods

- **Enumerative schemes.** simply scan every point in the search space by evaluating the objective function.
    - Not practical for infinite search spaces.
    - Lack the efficiency.
- **Dynamic Programming.** is enumerative method.
    - Suffer from curse of dimensionality.

Robustness of Tradiitional Optimization Methods

Existing main optimization methods

- **Enumerative schemes.** simply scan every point in the search space by evaluating the objective function.
  - Not practical for infinite search spaces.
  - Lack the efficiency.
- **Dynamic Programming.** is enumerative method.
  - Suffer from curse of dimensionality.

Not robust due to inefficiency

Robustness of Tradiitional Optimization Methods

Existing main optimization methods

- **Random search.** simply randomly search and save the best.

Robustness of Tradiitional Optimization Methods

Existing main optimization methods

- **Random search.** simply randomly search and save the best.
  - Not practical for infinite search spaces.

Robustness of Tradiitional Optimization Methods

Existing main optimization methods

- **Random search.** simply randomly search and save the best.
    - Not practical for infinite search spaces.
    - Lack the efficiency.

Robustness of Tradiitional Optimization Methods

Existing main optimization methods

- **Random search.** simply randomly search and save the best.
  - Not practical for infinite search spaces.
  - Lack the efficiency.

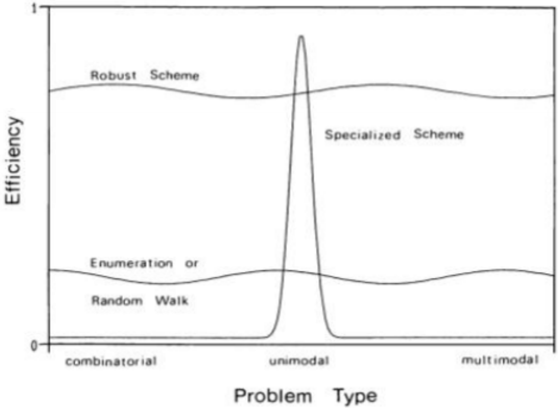
Robustness of Tradiitional Optimization Methods

Existing main optimization methods

- **Random search.** simply randomly search and save the best.
  - Not practical for infinite search spaces.
  - Lack the efficiency.

Not robust due to inefficiency

Efficiency and Robustness



A robust scheme work well across a broad spectrum of problem types

The goals of optimization

We should distinguish between

1. Process of optimization. How to improve?
2. Optimal point. Destination

The goals of optimization

We should distinguish between

1. Process of optimization. How to improve?
2. Optimal point. Destination

In practice and most complex problems it would be nice to be perfect but it is good if we can only improve.

Outline

Simple Genetic Algorithm

1. `P(0) ← Generate-Random-Population()`
2. **while** `Not-Terminated?` **do**
    1. `P(t) ← Evaluate-Population(P(t))`
    2. `P(t) ← Reproduction(P(t))`
    3. `P(t+1) ← Generate-Offspring(P(t))`
    4. `t ← t + 1`
3. **return** `P(t)`

How GA are different from other methods

GA is different if four ways:

1. GA works with coding the variable (i.e., parameter set).
2. GA search from population of points
3. GA use a pay-off (fitness) function.
4. GA use probabilistic transition rules.

How GA are different from other methods

GA is different if four ways:

1. GA works with coding the variable (i.e., parameter set).
2. GA search from population of points
3. GA use a pay-off (fitness) function.
4. GA use probabilistic transition rules.

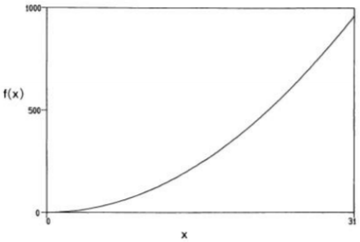
Three main operations in GA:

- Reproduction
- Cross over
- Mutation

How GA are different from other methods 1&2

For a function $f(x) = x^2$ where $x \in [0, 31]$ a population of five bit strings will evolve through time **Ex.** Initial population of size n = 4

- 01101
- 11000
- 01000
- 10011



Working with a population decrease the opprtuinity to stuck in local minima (multi-modal problems)

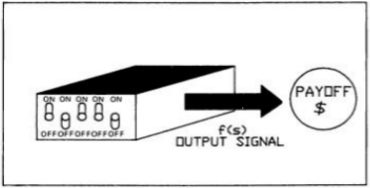
Remember base-x transformation:

$(10011)_2 = 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 19$

$(53596)_{10} = 5 \times 10^4 + 3 \times 10^3 + 5 \times 10^2 + 9 \times 10^1 + 6 \times 10^0 = 53596$

How GA are different from other methods 3

For a function $f(x) = x^2$ where $x \in [0, 31]$ single instance from the population have a fitness value.



Ex.

- 00001 is less fit than 01101
- 11000 is more fit than 01101

Working with fitness eliminate the need for an auxiliary information such as gradient or tables

How GA are different from other methods 4

GA uses probabilistic transition rule to move from population t to population t+1.

Can this be considered as a random search?

How GA are different from other methods 4

GA uses probabilistic transition rule to move from population t to population t+1.

Can this be considered as a random search?

Ans.

- No, GA use random choice as a tool to guide the search towards region of search space with **likely improvement**
- GAs are not classified as randomized search schemes.

Walk through a simple genetic algorithm

For a function $f(x) = x^2$ where $x \in [0, 31]$ a population of five bit strings will evolve through time

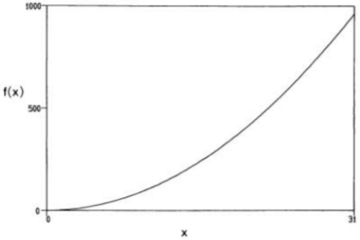
Ex.

Initial population of size n $= 4$

- 01101
- 11000
- 01000
- 10011

Operators:

- Reproduction
- Crossover
- Mutation



Reproduction Operator

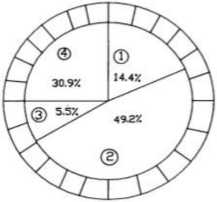
Reproduction

Is the process in which individual strings are copied according to their objective function values (fitness).

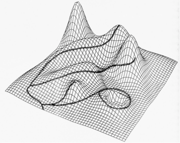
- f measures profit we want to maximize.
- string x with high $f(x)$ value has higher probability to have more offsprings.
- To get a reproduction candidate simply spin a roulette wheel

Ex.

No.	String	Fitness	% of Total
1	01101	169	14.4
2	11000	576	49.2
3	01000	64	5.5
4	10011	361	30.9
Total		1170	100.0



Reproduction Operator



In selecting potential parents we usually have a trade off between exploration and exploitation.

Exploitation

Concentrate on individuals on a certain region of the search space.

- Restricting search space.
- Less diversity.
- Reach local optimum.

Exploration

Concentrate on diverse individuals over different regions of the search space.

Solution: Trade-off, give finite probability to worse individuals to become parents to maintain diversity!

Crossover Operator

Crossover

is a genetic operator used to combine the genetic information of two parents to generate new offspring. Also known as mating operator.

Crossover Operator

Crossover

is a genetic operator used to combine the genetic information of two parents to generate new offspring. Also known as mating operator.

- Members of newly produced strings are mated at random
- An integer position k is selected at random from $[1, \ell - 1]$
- The two parents are cut at position k and the resulting substrings are swapped

Crossover Operator

Crossover

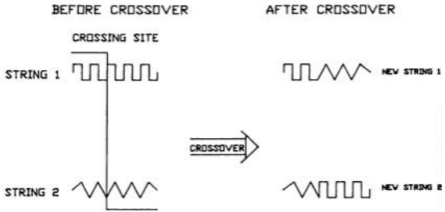
is a genetic operator used to combine the genetic information of two parents to generate new offspring. Also known as mating operator.

- Members of newly produced strings are mated at random
- An integer position k is selected at random from $[1, \ell - 1]$
- The two parents are cut at position k and the resulting substrings are swapped

Ex. Let $k = 4$

$$A_1 = 0\ 1\ 1\ 0 \mid 1$$
$$A_2 = 1\ 1\ 0\ 0 \mid 0$$

$$A'_1 = 0\ 1\ 1\ 0\ 0$$
$$A'_2 = 1\ 1\ 0\ 0\ 1$$



Mutation Operator

Mutation

Mutation is a genetic operator used to maintain genetic diversity from one generation of a population to the next.

Mutation Operator

Mutation

Mutation is a genetic operator used to maintain genetic diversity from one generation of a population to the next.

- Has very small probability.
- Every bit in a string of length ℓ can flip (mutate) with probability 0.001
- Aim to avoid local minima by preventing the population from becoming too similar to each other

Mutation Operator

Mutation

Mutation is a genetic operator used to maintain genetic diversity from one generation of a population to the next.

- Has very small probability.
- Every bit in a string of length ℓ can flip (mutate) with probability 0.001
- Aim to avoid local minima by preventing the population from becoming too similar to each other

Ex.

1 0 1 0 0 1 0

↓

1 0 1 0 1 1 0

GA Initialization

- Initialization should be kept simple in most GA applications
- The first population is seeded by randomly generated individuals.
- Problem-specific heuristics can be used in this step, to create an initial population with higher fitness.

GA Termination

The following options are commonly used :

- The maximally allowed CPU time elapses.
- The total number of generations reaches a given limit.
- The fitness improvement remains under a threshold value for a given period of time (i.e., for a number of generations or fitness evaluations).
- The population diversity drops under a given threshold.

Notes

- The introduced operators use randomness, however it is a directed randomness.
- You efficiently build new solutions from the best partial solutions of previous trials.
- Think of it like a group of people attending academic conferences, who has greater chance to speak, how people can exchange ideas.

Outline

GA a simulation by hand

For a function $f(x) = x^2$ where $x \in [0, 31]$ a population of 4 strings.

String No.	Initial Population (Randomly Generated)	x Value (Unsigned Integer)	$f(x)$ x^2	pselect, $\frac{f_i}{\Sigma f}$	Expected count $\frac{f_i}{\bar{f}}$	Actual Count from Roulette Wheel
1	0 1 1 0 1	13	169	0.14	0.58	1
2	1 1 0 0 0	24	576	0.49	1.97	2
3	0 1 0 0 0	8	64	0.06	0.22	0
4	1 0 0 1 1	19	361	0.31	1.23	1
Sum			1170	1.00	4.00	4.0
Average			293	0.25	1.00	1.0
Max			576	0.49	1.97	2.0

GA a simulation by hand

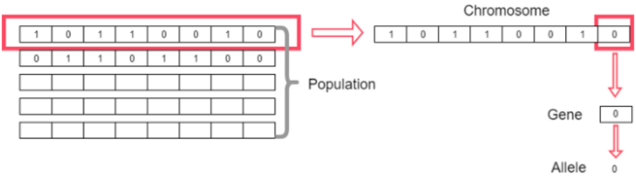
Reproduction, and Crossover with no mutation.

Mating Pool after Reproduction (Cross Site Shown)	Mate (Randomly Selected)	Crossover Site (Randomly Selected)	New Population	x Value	$f(x)$ x^2
0 1 1 0 | 1	2	4	0 1 1 0 0	12	144
1 1 0 0 | 0	1	4	1 1 0 0 1	25	625
1 1 | 0 0 0	4	2	1 1 0 1 1	27	729
1 0 | 0 1 1	3	2	1 0 0 0 0	16	256
					1754
					439
					729

- The population average fitness improved from 239 to 439.
- The maximum fitness also improved from 576 to 729

Terminology Comparsion

Natural	Genetic Algorithm
chromosome	string
gene	feature, character, or detector
allele	feature value
locus	string position
genotype	structure
phenotype	parameter set, alternative solution, a decoded structure
epistasis	nonlinearity



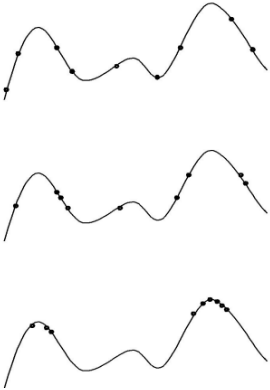
When to use GA

- Highly multimodal functions
- Discrete or discontinuous functions
- High-dimensionality functions, including many combinatorial ones
  Nonlinear dependencies on parameters (interactions among parameters)
- Often used for approximating solutions to NP complete combinatorial problems
- DON'T USE if a hill-climber, etc., will work well

Outline

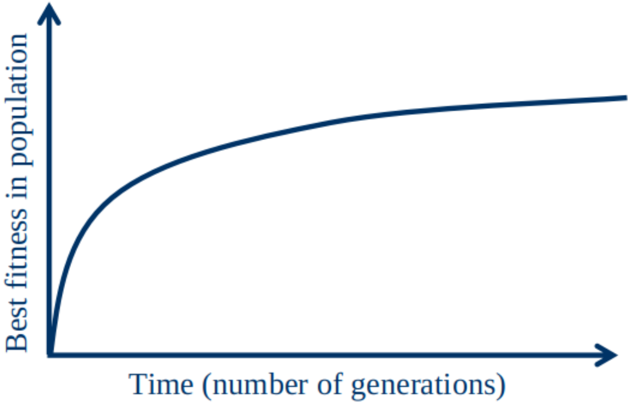
Typical behavior of an EA

Phases in optimizing on a 1-dimensional fitness landscape



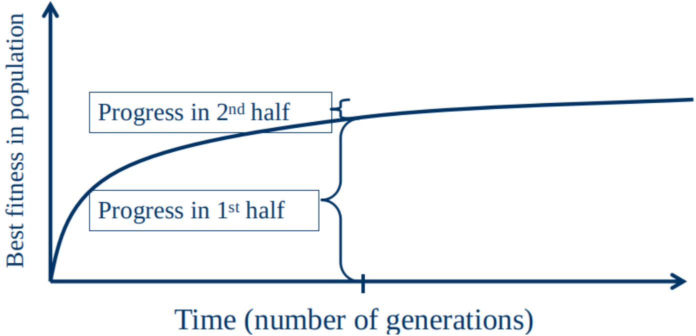
- Early phase: quasi-random population distribution
- Mid-phase: population arranged around/on hills
- Late phase: population concentrated on high hills

Typical run: progression of fitness

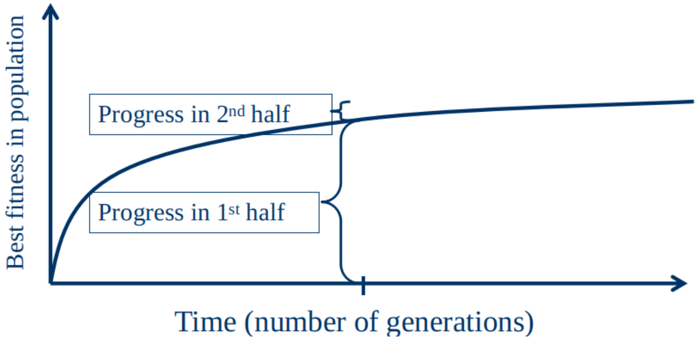


Typical run of an EA shows so-called anytime behavior

Are long runs beneficial?

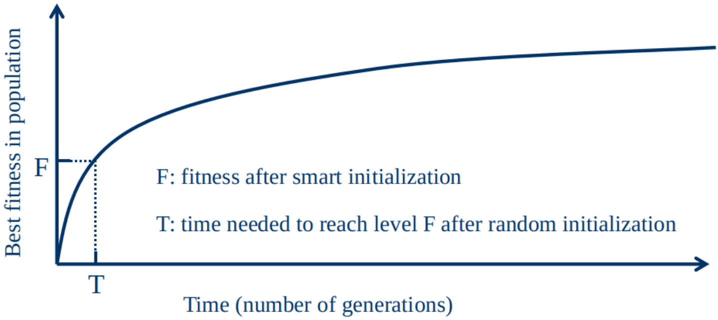


Are long runs beneficial?

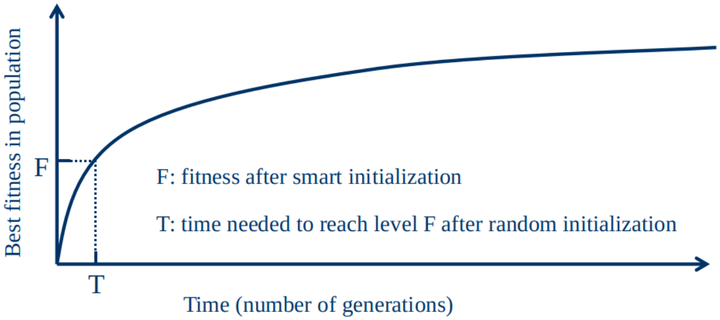


- it depends on how much you want the last bit of progress
- it may be better to do more shorter runs

Is it worth expending effort on smart initialization?



Is it worth expending effort on smart initialization?



- possibly, if good solutions/methods exist.
- care is needed, see chapter on hybridization

References

- Goldenberg, D.E., 1989. Genetic algorithms in search, optimization and machine learning.
- Michalewicz, Z., 2013. Genetic algorithms + data structures= evolution programs. Springer Science & Business Media.

Questions