Session : Genetic Algorithm

Course Title: Computational Intelligence
Course Code: 19CSE422A

Course Leader:
Dr. Vaishali R. Kulkarni
Assistant Professor, Department of Computer Science and Engineering
Faculty of Engineering and Technology
Ramaiah University of Applied Sciences, Bengaluru
Email: vaishali.cs.et@msruas.ac.in
Tel: +91-804-906-5555 Ext:2325
Website: www.msruas.ac.in/staff/fet_cse#Vaishali

Objectives of this Session

I wish to provide a foundation to:

1. Generic GAs
2. GA operators (crossover and mutation)
3. Variants of GA (CGAs and SSGAs)
4. Engineering applications of GAs

Intended Outcomes of this Session

At the end of this session, the student will be able to:

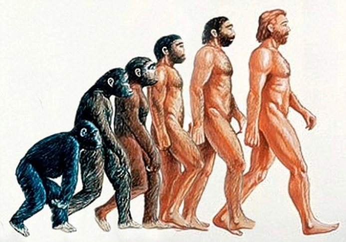
1. Implement a canonical GA to solve a given optimization problem
2. Classify GAs based on various figures of merit
3. Compare among the various types of GAs
4. Relate the GA operators with those in biological genetics
5. Perform one-point, two-point and uniform crossover on given chromosomes
6. Perform random and in-order mutation on given chromosomes
7. Illustrate the various GA operators

Recommended Resources for this Session

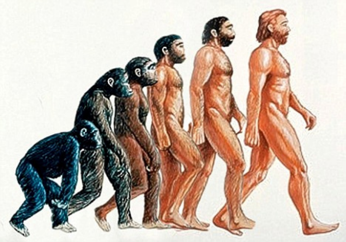
1. Engelbrecht, A. P. (2007). *Computational intelligence: An introduction*. Chichester, England, John Wiley & Sons.
2. De Jong, K. A. (2012). *Evolutionary Computation: A Unified Approach*. New York, USA, Bradford Books.
3. Konar, A. (2005). *Computational Intelligence: Principles, Techniques and Applications*. Secaucus, NJ, USA, Springer-Verlag New York, Inc.
4. Q. Wu, N. Rao, J. Barhen, S. Iyengar, V. Vaishnavi, H. Qi, and K. Chakrabarty. (2004). On computing mobile agent routes for data fusion in distributed sensor networks.*IEEE Transactions on Knowledge Data Eng.*, vol. 16, no. 6, pp. 740753.

Genetic Algorithms

- Popularized by John Holland, University of Michigan, in 1970

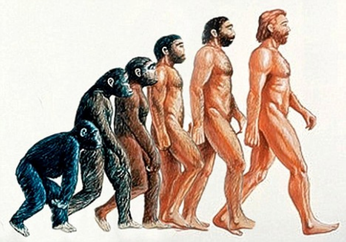


Genetic Algorithms



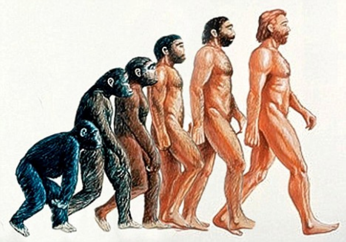
- Popularized by John Holland, University of Michigan, in 1970
- It models the mechanics of biological evolution

Genetic Algorithms



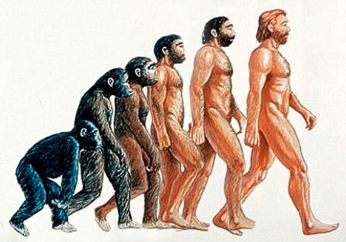
- Popularized by John Holland, University of Michigan, in 1970
- It models the mechanics of biological evolution
- Captures the biological genetic inheritance, mutation, selection and crossover

Genetic Algorithms



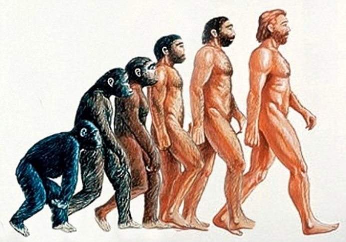
- Popularized by John Holland, University of Michigan, in 1970
- It models the mechanics of biological evolution
- Captures the biological genetic inheritance, mutation, selection and crossover
- Consists of a population of potential solutions called Chromosomes

Genetic Algorithms



- Popularized by John Holland, University of Michigan, in 1970
- It models the mechanics of biological evolution
- Captures the biological genetic inheritance, mutation, selection and crossover
- Consists of a population of potential solutions called Chromosomes
- A chromosome is made from N genes that represent the parameters to be optimized

Genetic Algorithms



- Popularized by John Holland, University of Michigan, in 1970
- It models the mechanics of biological evolution
- Captures the biological genetic inheritance, mutation, selection and crossover
- Consists of a population of potential solutions called Chromosomes
- A chromosome is made from N genes that represent the parameters to be optimized
- Represents a **species that evolves with time**

Genetic Algorithms

- Fraser (1957), Bremermann (1962), Reed (1967) and Holland (1975) are the early researchers in GAs. Due to Holland's exhaustive work, GA became popular and he is considered the father of GAs

Genetic Algorithms

- Fraser (1957), Bremermann (1962), Reed (1967) and Holland (1975) are the early researchers in GAs. Due to Holland's exhaustive work, GA became popular and he is considered the father of GAs
- GAs model genetic evolution, where the characteristics of individuals are expressed using genotypes

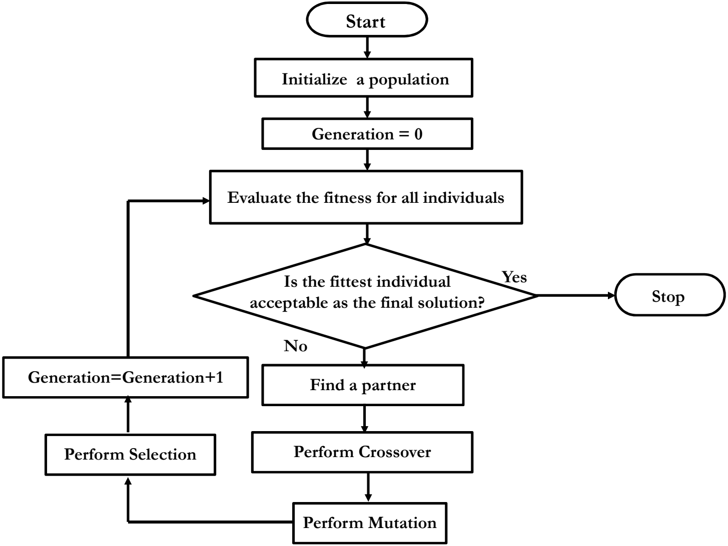
Genetic Algorithms

- Fraser (1957), Bremermann (1962), Reed (1967) and Holland (1975) are the early researchers in GAs. Due to Holland's exhaustive work, GA became popular and he is considered the father of GAs
- GAs model genetic evolution, where the characteristics of individuals are expressed using genotypes
- The driving operators of a GA are selection (to model survival of the fittest) and recombination through crossover operator (to model reproduction)

Genetic Algorithms

- Fraser (1957), Bremermann (1962), Reed (1967) and Holland (1975) are the early researchers in GAs. Due to Holland's exhaustive work, GA became popular and he is considered the father of GAs
- GAs model genetic evolution, where the characteristics of individuals are expressed using genotypes
- The driving operators of a GA are selection (to model survival of the fittest) and recombination through crossover operator (to model reproduction)
- In the canonical GA the following were used:
  - A bitstring chromosome representation
  - Proportional selection
  - One-point crossover
  - Uniform mutation (as a background operator with little importance)

Flowchart of a Generic GA



Crossover in GA

- Divided into three main categories based on the the number of parents used

Crossover in GA

- Divided into three main categories based on the the number of parents used
- **One Parent:** An offspring is generated from one parent

Crossover in GA

- Divided into three main categories based on the the number of parents used
- **One Parent:** An offspring is generated from one parent
- **Two Parent:** Two parents are used to produce one or two offspring

Crossover in GA

- Divided into three main categories based on the the number of parents used
- **One Parent:** An offspring is generated from one parent
- **Two Parent:** Two parents are used to produce one or two offspring
- **Multi-recombination:** More than two parents are used to produce one or more offspring

Crossover in GA

- Divided into three main categories based on the the number of parents used
- **One Parent:** An offspring is generated from one parent
- **Two Parent:** Two parents are used to produce one or two offspring
- **Multi-recombination:** More than two parents are used to produce one or more offspring
- Crossover operators are further categorized based on the representation scheme used (binary-specific, floating point, etc)

Crossover in GA

- Divided into three main categories based on the the number of parents used
- **One Parent:** An offspring is generated from one parent
- **Two Parent:** Two parents are used to produce one or two offspring
- **Multi-recombination:** More than two parents are used to produce one or more offspring
- Crossover operators are further categorized based on the representation scheme used (binary-specific, floating point, etc)
- Recombination is applied probabilistically. Each pair (or group) of parents have a probability p_c of producing offspring. Usually, a high crossover probability is used

Crossover in GA

- Divided into three main categories based on the the number of parents used
- **One Parent:** An offspring is generated from one parent
- **Two Parent:** Two parents are used to produce one or two offspring
- **Multi-recombination:** More than two parents are used to produce one or more offspring
- Crossover operators are further categorized based on the representation scheme used (binary-specific, floating point, etc)
- Recombination is applied probabilistically. Each pair (or group) of parents have a probability p_c of producing offspring. Usually, a high crossover probability is used
- In selection of parents, the following should be considered:
  - The same individual should not be used as both parents
  - The same individual should not take part in more than o

Crossover in GA

- An offspring produced may replace the worst parent (replacement)

Crossover in GA

- An offspring produced may replace the worst parent (replacement)
- It should be restricted that the offspring must be more fit than the worst parent

Crossover in GA

- An offspring produced may replace the worst parent (replacement)
- It should be restricted that the offspring must be more fit than the worst parent
- Crossover operators have been implemented where the offspring replaces the worst individual of the population

Crossover in GA

- An offspring produced may replace the worst parent (replacement)
- It should be restricted that the offspring must be more fit than the worst parent
- Crossover operators have been implemented where the offspring replaces the worst individual of the population
- In the case of two offspring, similar replacement strategies can be used

Crossover for Binary Gene Representation

- **One-point crossover:** The operator randomly selects a crossover point, and the bitstrings after that point are swapped between the two parents

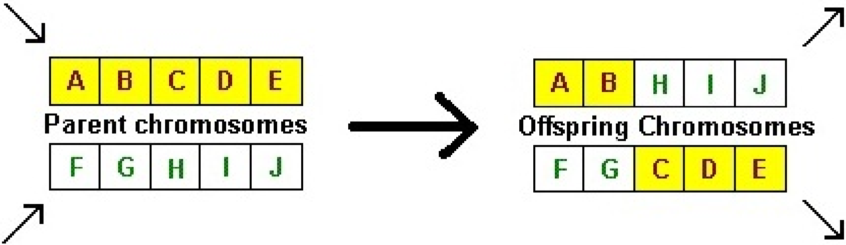
Crossover for Binary Gene Representation

- **One-point crossover:** The operator randomly selects a crossover point, and the bitstrings after that point are swapped between the two parents
- **Two-point crossover:** Two bit positions are randomly selected, and the bitstrings between these points are swapped

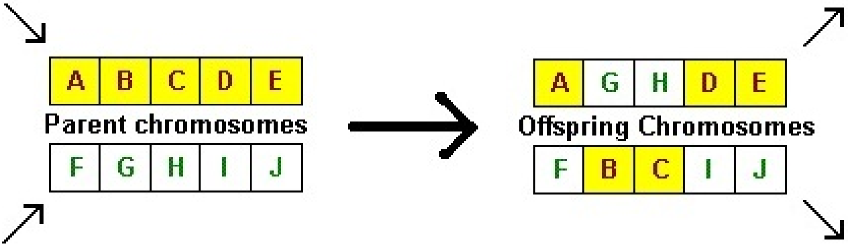
Crossover for Binary Gene Representation

- **One-point crossover:** The operator randomly selects a crossover point, and the bitstrings after that point are swapped between the two parents
- **Two-point crossover:** Two bit positions are randomly selected, and the bitstrings between these points are swapped
- **Uniform crossover:** An n_x-dimensional mask is created randomly. Here, p_x is the bit-swapping probability. If $p_x = 0.5$, then each bit has an equal chance to be swapped

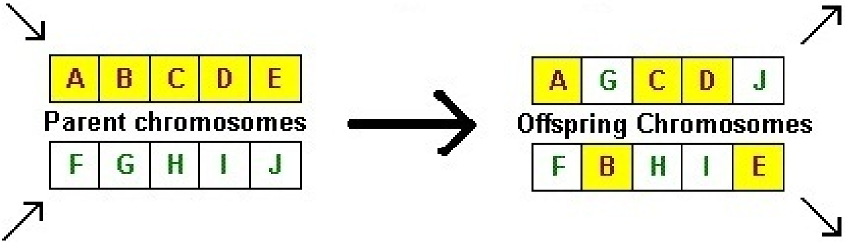
One Point Crossover



Two Point Crossover



Uniform Crossover



Control Parameters p_m and p_c

- In addition to the population size, the performance of a GA is influenced by the mutation rate p_m and the crossover rate p_c

Control Parameters p_m and p_c

- In addition to the population size, the performance of a GA is influenced by the mutation rate p_m and the crossover rate p_c
- In early GAss very low values for p_m and relatively high values for p_c are propagated

Control Parameters p_m and p_c

- In addition to the population size, the performance of a GA is influenced by the mutation rate p_m and the crossover rate p_c
- In early GAss very low values for p_m and relatively high values for p_c are propagated
- Optimal settings for p_m and p_c can significantly improve performance. To obtain such optimal settings through empirical parameter tuning is a time consuming process

Control Parameters p_m and p_c

- In addition to the population size, the performance of a GA is influenced by the mutation rate p_m and the crossover rate p_c
- In early GAss very low values for p_m and relatively high values for p_c are propagated
- Optimal settings for p_m and p_c can significantly improve performance. To obtain such optimal settings through empirical parameter tuning is a time consuming process
- A solution to the problem of finding best values for these control parameters is to use dynamically changing parameters

Control Parameters p_m and p_c

- In addition to the population size, the performance of a GA is influenced by the mutation rate p_m and the crossover rate p_c
- In early GAss very low values for p_m and relatively high values for p_c are propagated
- Optimal settings for p_m and p_c can significantly improve performance. To obtain such optimal settings through empirical parameter tuning is a time consuming process
- A solution to the problem of finding best values for these control parameters is to use dynamically changing parameters
- In addition to p_m and p_c, the choice of the best evolutionary operators to use is also problem dependent

Variants of GA

- Different implementations of a GA can be obtained by using different combinations of selection, crossover, and mutation operators

Variants of GA

- Different implementations of a GA can be obtained by using different combinations of selection, crossover, and mutation operators
- Although different operator combinations result in different behaviors, the same algorithmic flow discussed above followed

Variants of GA

- Different implementations of a GA can be obtained by using different combinations of selection, crossover, and mutation operators
- Although different operator combinations result in different behaviors, the same algorithmic flow discussed above followed
- The GA discussed so far differs from the biological evolution since population sizes in GAs are are fixed

Variants of GA

- Different implementations of a GA can be obtained by using different combinations of selection, crossover, and mutation operators
- Although different operator combinations result in different behaviors, the same algorithmic flow discussed above followed
- The GA discussed so far differs from the biological evolution since population sizes in GAs are are fixed
- Selection process is described by how a parent is selected, and how it decides if offspring will replace parents, and which parents to replace

Variants of GA

- Different implementations of a GA can be obtained by using different combinations of selection, crossover, and mutation operators
- Although different operator combinations result in different behaviors, the same algorithmic flow discussed above followed
- The GA discussed so far differs from the biological evolution since population sizes in GAs are are fixed
- Selection process is described by how a parent is selected, and how it decides if offspring will replace parents, and which parents to replace
- Two classes based on the replacement strategy: **Generational genetic algorithms (GGA)** and **steady state genetic algorithms (SSGA)**

GGAs and SSGAs

- The replacement strategy replaces all parents with their offspring after all offpsring have been created and mutated

GGAs and SSGAs

- The replacement strategy replaces all parents with their offspring after all offpsring have been created and mutated
- This results in no overlap between the current population and the new population

GGAs and SSGAs

- The replacement strategy replaces all parents with their offspring after all offpsring have been created and mutated
- This results in no overlap between the current population and the new population
- For SSGAs, a decision is made immediately after an offspring is created and mutated as to whether the parent or the offspring survives to the next generation

GGAs and SSGAs

- The replacement strategy replaces all parents with their offspring after all offpsring have been created and mutated
- This results in no overlap between the current population and the new population
- For SSGAs, a decision is made immediately after an offspring is created and mutated as to whether the parent or the offspring survives to the next generation
- Thus, there exists an overlap between the current and new populations

GGAs and SSGAs

- The replacement strategy replaces all parents with their offspring after all offpsring have been created and mutated
- This results in no overlap between the current population and the new population
- For SSGAs, a decision is made immediately after an offspring is created and mutated as to whether the parent or the offspring survives to the next generation
- Thus, there exists an overlap between the current and new populations
- The amount of overlap between the current and new populations is referred to as the generation gap

GGAs and SSGAs

- The replacement strategy replaces all parents with their offspring after all offpsring have been created and mutated
- This results in no overlap between the current population and the new population
- For SSGAs, a decision is made immediately after an offspring is created and mutated as to whether the parent or the offspring survives to the next generation
- Thus, there exists an overlap between the current and new populations
- The amount of overlap between the current and new populations is referred to as the generation gap
- GGAs have a zero generation gap, while SSGAs generally have large generation gaps

Replacement Strategies for SSGAs

- Replace worst: The offspring replaces the worst individual of the current population

Replacement Strategies for SSGAs

- Replace worst: The offspring replaces the worst individual of the current population
- Replace random: The offspring replaces a randomly selected individual of the current population

Replacement Strategies for SSGAs

- Replace worst: The offspring replaces the worst individual of the current population
- Replace random: The offspring replaces a randomly selected individual of the current population
- Kill tournament: A group of individuals is randomly selected, and the worst individual of this group is replaced with the offspring

Replacement Strategies for SSGAs

- Replace worst: The offspring replaces the worst individual of the current population
- Replace random: The offspring replaces a randomly selected individual of the current population
- Kill tournament: A group of individuals is randomly selected, and the worst individual of this group is replaced with the offspring
- Replace oldest: A first-in-first-out strategy is followed by replacing the oldest individual of the current population

Replacement Strategies for SSGAs

- Conservative selection: A tournament size of two individuals is used of which one is the oldest individual of the current population. The worst of the two is replaced by the offspring

Replacement Strategies for SSGAs

- Conservative selection: A tournament size of two individuals is used of which one is the oldest individual of the current population. The worst of the two is replaced by the offspring

- Elitist strategies: The best individual is excluded from selection

Replacement Strategies for SSGAs

- Conservative selection: A tournament size of two individuals is used of which one is the oldest individual of the current population. The worst of the two is replaced by the offspring
- Elitist strategies: The best individual is excluded from selection
- Parent-offspring competition: A selection strategy decides if an offspring replaces one of its own parents

Genetic Operators

- **Selection Operator:** A child that has better fitness than the population minimum is added to the population, and the parent having minimum fitness is removed

Genetic Operators

- **Selection Operator**: A child that has better fitness than the population minimum is added to the population, and the parent having minimum fitness is removed

- **Crossover Operator:** Two point crossover is used with a large crossover probability ($p_c \geq 0.9$)

Genetic Operators

- **Mutation:** Two mutation points are selected randomly and the values of these two points are exchanged with a small crossover probability ($p_m \leq 0.1$)

Before Mutation:

0	1	2	3	7	5	6	8	9	11	10	**Level 1**
1	1	1	0	1	1	0	1	0	1	0	**Level 2**

After Mutation:

0	1	2	3	9	5	6	8	7	11	10	**Level 1**
1	1	1	0	0	1	0	1	1	1	0	**Level 2**

Genetic Operators

- **Inversion:** Two mutation points are selected randomly and the order of nodes between these two points is reversed

Before Inversion:

0	1	2	3	9	5	6	8	7	11	10	Level 1
1	1	1	0	0	1	0	1	1	1	0	Level 2

After Inversion:

0	1	2	3	9	8	6	5	7	11	10	Level 1
1	1	1	0	0	1	0	1	1	1	0	Level 2

Engineering Applications of GAs

- Design
  - Design Optimization (turbojet engine)
  - Compiler Code Optimization

Engineering Applications of GAs

- Design
  - Design Optimization (turbojet engine)
  - Compiler Code Optimization
- Control Engineering
  - PID Parameter Tuning
  - Neural Network Training

Engineering Applications of GAs

- Design
  - Design Optimization (turbojet engine)
  - Compiler Code Optimization
- Control Engineering
  - PID Parameter Tuning
  - Neural Network Training
- Power Systems
  - Maintainable Scheduling
  - Economic Dispatch

Engineering Applications of GAs

- Design
  - Design Optimization (turbojet engine)
  - Compiler Code Optimization
- Control Engineering
  - PID Parameter Tuning
  - Neural Network Training
- Power Systems
  - Maintainable Scheduling
  - Economic Dispatch
- Distributed Networks
  - Sensor Localization
  - Sensor Placement
  - Optimal Routing

Engineering Applications of GAs

- Design
  - Design Optimization (turbojet engine)
  - Compiler Code Optimization
- Control Engineering
  - PID Parameter Tuning
  - Neural Network Training
- Power Systems
  - Maintainable Scheduling
  - Economic Dispatch
- Distributed Networks
  - Sensor Localization
  - Sensor Placement
  - Optimal Routing
- Robotics
  - Optimal Path Planning
  - Swarm Robotics

GA: Summary

✓ Many variants are available for all sorts of optimization problems

GA: Summary

- ✓ Many variants are available for all sorts of optimization problems
- ✓ Has a long history; so reliable

GA: Summary

- ✓ Many variants are available for all sorts of optimization problems
- ✓ Has a long history; so reliable
- ✗ Reported to have a tendency to converge to sub-optimal solution

Session Summary

- GA operators are: Crossover and Mutation

Session Summary

- GA operators are: Crossover and Mutation
- Popular crossover operators: One-point, Two-point and uniform

Session Summary

- GA operators are: Crossover and Mutation
- Popular crossover operators: One-point, Two-point and uniform
- Popular mutation operators: Random and In-order

Session Summary

- GA operators are: Crossover and Mutation
- Popular crossover operators: One-point, Two-point and uniform
- Popular mutation operators: Random and In-order
- Interactive evolution involves a human user on-line into the selection and variation processes

Session Summary

- GA operators are: Crossover and Mutation
- Popular crossover operators: One-point, Two-point and uniform
- Popular mutation operators: Random and In-order
- Interactive evolution involves a human user on-line into the selection and variation processes
- Application of GA include: Design, Control Engineering, Power Systems, Distributed Networks, Robotics and many others

Any Questions?

Thank You