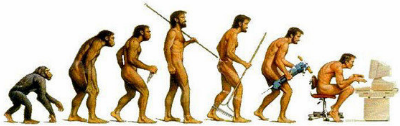
Evolutionary Computation

Introduction

Introduction

- Evolutionary computing represents another tool of soft computing techniques based on the concepts of artificial evolution.
- Generally speaking, evolution is the process by which life adapts to the changing environments.



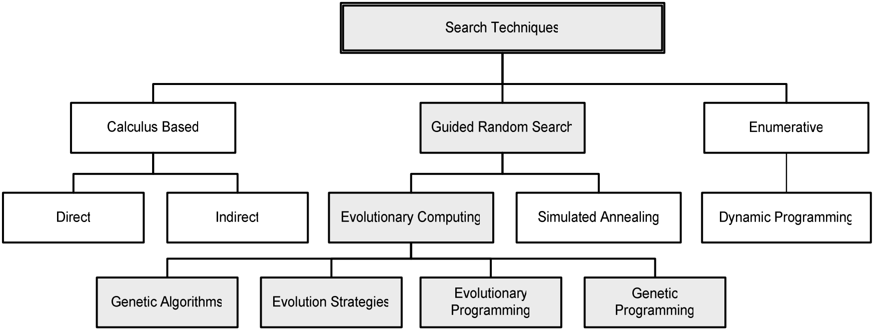
Introduction (cont.)

- A creature's survival depends, to a large extent, on its fitness within its environment, which is in turn determined by its genetic makeup.
- Chance, however, is always a factor, and even the fittest can die in unlucky circumstances.

Introduction (cont.)

- Evolution relies on having entire populations of slightly different organisms so that if some fail, others may yet succeed.
- Researchers have sought to formalize the mechanisms of evolution in order to apply it artificially to very different problems.
- The pursuit of artificial evolution using computers has lead to the development of an area commonly known as evolutionary computation or evolutionary algorithms.

Search Techniques



Search Techniques (cont.)

- EAs are different from the conventional single-point based optimization techniques such as gradient search and directed search methods:
    - They start the search from a population of points not from a single point.
    - They work with a coded version of the parameters.
    - They use stochastic reproduction instead deterministic rules.

Advantages of Evolutionary Algorithms

- EAs need no assumptions about the objective function of the given optimization problem.
  - such as continuity, unimodality, differentiability.
- EAs are robust to the problem type and generally yield optimal or suboptimal solutions.
- EAs can be run interactively (online parameter tuning).

Disadvantages of Evolutionary Algorithms

- No complete theoretical basis, but much progress has been made.
- No guarantee for finding the optimal solution, but guaranteed for some special cases.

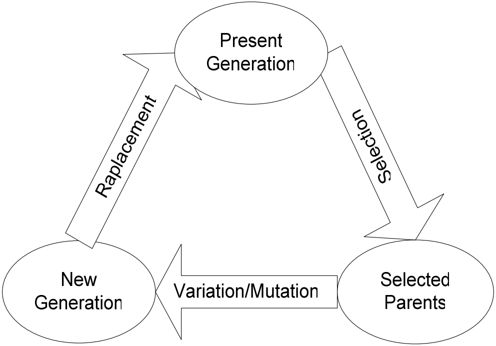
Where to Use EAs?

- We use an EA to solve the optimization problem, where random or enumerative search is impractical due to huge search space.
- We use an EA to solve the optimization problem, where a single-point based optimization technique gives poor results due to the existence of many local optima.

Common Components in EAs

- Representation of individuals: Coding
- Evaluation method for individuals: Fitness function
- Population initialization
- Parent selection mechanism
- Variation operators (crossover and mutation)
- Survivor Selection mechanism

Basic Evolution Cycle



- Different evolutionary computing techniques incorporate variation of the basic evolution cycle in terms of presentation models and specific combinations of evolutionary operations.

Basic Evolution Cycle (cont.)

Two Opposite Operations

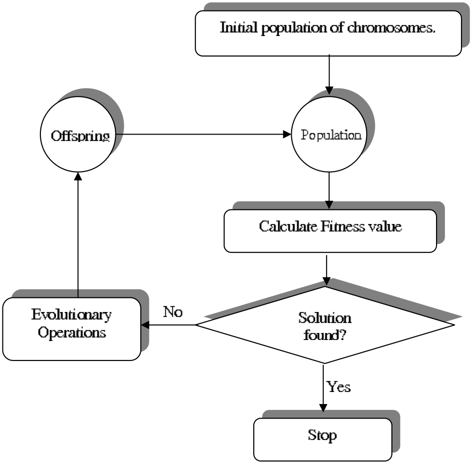
1. In one hand the selection operation intends to reduce diversity of the sample population.
2. On the other hand the crossover and the mutation operators try to increase the diversity of the population.

- These two opposite evolutionary operations enable the algorithm to improve the quality of sample solutions while exploring the whole search space.

EA Algorithm Outline

- $t := 0;$
- Initialize $P(t)$
- Evaluate $P(t);$
- While not terminate do
  - $P'(t) :=$ select $P(t);$
  - $P''(t) :=$ variation $P'(t);$
  - Evaluate $P''(t);$
  - $P(t+1) :=$ select $P(t)\ U\ P''(t);$
  - $t := t+1;$
- End while

Basic Structure of an Evolutionary Algorithm



Evolutionary Computation

- Evolutionary Computation is a broad term that covers a family of adaptive search population-based techniques that can be applied to the optimization of both discrete and continuous mappings.

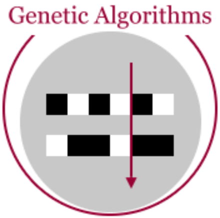
Evolutionary Computation

- Evolutionary programming
- Evolutionary strategies
- Genetic programming
- Genetic algorithms

Dialects of EAs

Genetic Algorithm (GA)

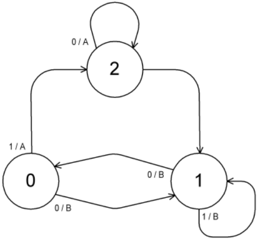
GA represents solution candidates by binary strings and applies recombination, mutation, and selection operators over them.



Dialects of EAs (cont.)

Evolutionary Programming (EP)

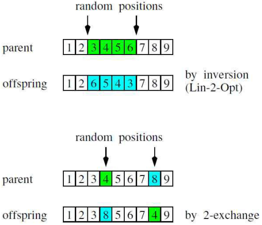
EP evolves the numerical parameters of a program.



Dialects of EAs (cont.)

Evolution Strategy (ES)

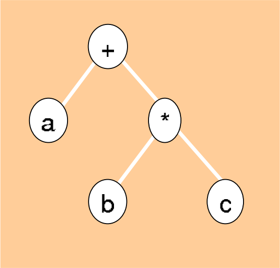
ES uses mutation, recombination, and selection applied to a population of individuals containing candidate solutions in order to evolve iteratively better and better solutions.



Dialects of EAs (cont.)

Genetic Programming (GP)

GP works with computer programs whose fitness are determined by their ability to solve a computational problem.



Genetic Algorithms

Genetic Algorithms: History

- GA's represent important class of evolutionary computing techniques.
- The origin of GA dates back to the early 50's when a group of computer scientists and biologists teamed up to simulate the behavior of a class of biological processes.
- But it was only later and in the early seventies that Holland and his associates introduced the methodology in a more formal and tractable way.

Genetic Algorithms: History

- GA's represent important class of evolutionary computing techniques.
- The origin of GA dates back to the early 50's when a group of computer scientists and biologists teamed up to simulate the behavior of a class of biological processes.
- But it was only later and in the early seventies that Holland and his associates introduced the methodology in a more formal and tractable way.

Genetic Algorithms

- Genetic algorithms are able of accurately solving a wide range of optimization problems.
- This is done through a procedure inspired from the biological process of evolution and the survival of the fittest concept.
- The search procedure of GA is stochastic in nature and doesn't usually provide for the exact location of the optima as some other gradient-based optimization techniques do.

Genetic Algorithms: Two Attractive Features

Advantages of GA to Derivative Based Approaches

- Given their discrete search nature, they could be easily applied to continuous as well as to discontinuous functions.
- Moreover, they usually outperform gradient based techniques in getting close to the global optima and hence avoid being trapped in local ones.

GA and Optimization

- Let suppose the goal is to maximize a function f of m variables given

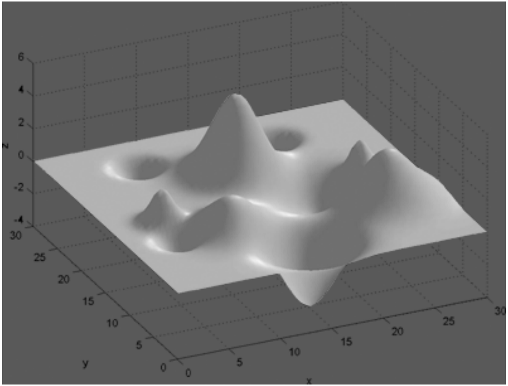
$$f(x_1, x_2, \cdots, x_m) : \; R^m \rightarrow \; R$$

- At the end of the optimization process one gets

$$f(x_1^*, x_2^*, \cdots, x_m^*) \geq f(x_1, x_2, \cdots, x_m)$$

  - where $(x_1^*, x_2^*, \cdots, x_m^*)$ represents the vector solution belonging to the search space(s) $R^m = (R \times R \times \times \cdots R)$

A 2D Function



Objective Function: Minimization Case

- In the case of minimization, one might pose the problem of optimization as that of maximization of a function h, which is the negation of f all over the search space.
- In other words, maximize the function h such as :

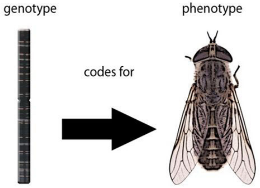
$$min\ f(x_1, \cdots, x_m) = max\ h(x_1, \cdots, x_m) = max(-f(x_1, \cdots, x_m))$$

GA and Optimization (cont.)

- The basic idea of GAs is to choose a random population in the range of optimization, with a fixed size n.
- Using the so-called binary encoding procedure, each variable is represented as a string of q binary digits.
- This leads to a population of elements represented by matrix of n rows and qm rows.
- A set of genetic operators is then applied to this matrix to create a new population at which the function f should attain increasingly larger values.

Genotype

- Each individual in nature has a form determined by its DNA.
- Its collection of genetic traits is commonly known as a genotype (or chromosome).
- In genetic algorithms, the term genotype is used to describe the encoding of a problem solution represented by an individual.



GAs: Terminology

Population of Potential Solutions

Population of Potential Solutions is the collection of solutions points (also called individuals).

Chromosome

Chromosome is an individual solution represented as an array of sequence of strings (also called genotype).

GAs: Terminology (cont.)

Gene

Gene is a string in the chromosome.

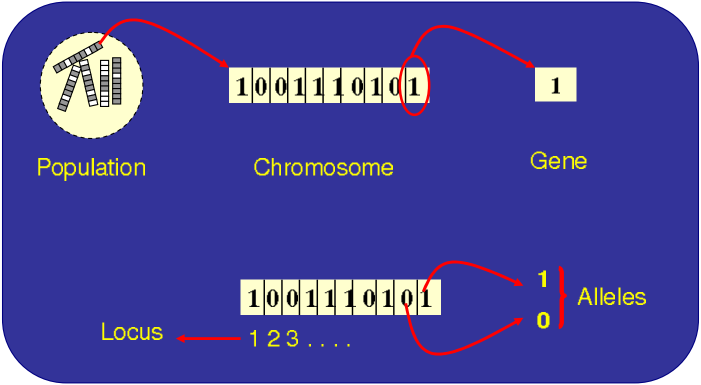
Alleles

Alleles is the values, or the states, that genes might have (0 or 1 in the case of binary encoding).

Locus

Locus is the position of a gene in a chromosome.

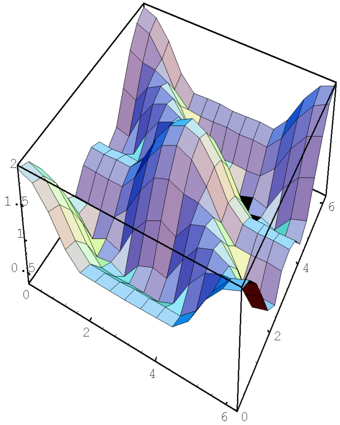
GAs Terminology: Graphical Illustration

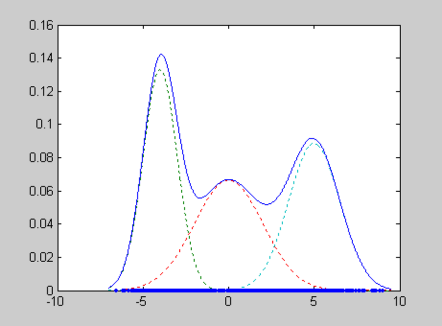


Fitness Function

- To use genetic algorithms, it is necessary to provide a means for evaluating the value or goodness of a particular solution.
- In nature, this is frequently thought to be the fitness of a creature (as in the popular concept of survival of the fittest), referring to its relative ability to survive in its environment.
- A fitness function, also called an objective function, measures the fitness or the goodness of a particular solution.

Fitness Function Examples





Fitness Function (cont.)

- In a number of cases, the objective function is quite obvious.
- In the equation minimization problem described above, the genotype is a set of parameters for the function, and the objective function is simply the value of the equation being minimized, given the genotype.
- In this case, a lower result from the objective function represents a better solution to problem.
- In many cases, however, a good objective function is more difficult to construct and heuristic approaches must be taken at times.

Fitness Function (cont.)

- In a number of cases, the objective function is quite obvious.
- In the equation minimization problem described above, the genotype is a set of parameters for the function, and the objective function is simply the value of the equation being minimized, given the genotype.
- In this case, a lower result from the objective function represents a better solution to problem.
- In many cases, however, a good objective function is more difficult to construct and heuristic approaches must be taken at times.

Schema Theorem

Schema Theorem: The Foundation Of GA

- Schema forms the theoretical basis of genetic algorithms.
- Schemata are templates that partially represent a solution point in the search space.
- Thus, a schema is a set of chromosomes that share certain values.
- These certain values are introduced in the chromosomes as undefined positions (*). i.e, they can take any value in the solution space.

Schema Theorem (cont.)

Schema Theorem

- If chromosomes are coded using symbols from an alphabet A, schemata are chromosomes whose symbols belong to $A \ U \ \{*\}$
- The * symbol is interpreted as a wildcard (or don't care), and its loci is called undefined.
- The size of the generated population depends directly on the number of the don't care symbols in the schema.
- A schema contains n don't care symbols match 2^n chromosomes.

Schema Theorem (cont.)

Example 1

- Given the schema 10100111*.
- The following chromosomes belong to this schema:
  - 101001111
  - 101001110
  - (Note that the * can be replaced with 0 or 1)
- Since we are using the binary encoding the alphabet size is 2 (0 and 1), and there is one (*) the maximum chromosomes can be obtained $= 2^1 = 2$

Schema Theorem (cont.)

Example 2

- Given the schema 101*0111*.
- From this schema we can derive the following chromosomes:
  - 101001110
  - 101001111
  - 101101110
  - 101101111
- Maximum number of chromosomes $= 2^2 = 4$

Schema Theorem: Definitions

Schema Order

- Schema order describes the number of *none* don't care symbols in the schema.
- The smaller order a schema has, the more general it is, i.e. the more chromosomes belong to it.

Schema Length

- Schema length is the distance between the furthest two *none* don't care symbols.
- The length determines the likelihood of a member of the schema being disrupted by crossover.
- The closer two symbols are, the most likely to stay together after the crossover operation.

Schema Theorem: Definitions

Schema Order

- Schema order describes the number of *none* don't care symbols in the schema.
- The smaller order a schema has, the more general it is, i.e. the more chromosomes belong to it.

Schema Length

- Schema length is the distance between the furthest two *none* don't care symbols.
- The length determines the likelihood of a member of the schema being disrupted by crossover.
- The closer two symbols are, the most likely to stay together after the crossover operation.

Schema Theorem: Definitions

Example

Schema	Order	Length
1*******	1	0
101*	3	2
*0*10***	3	3

Schema Theorem (cont.)

Schemata Fitness

- The fitness of a schema is the averages fitness of the chromosomes belong to the schema.
- In general, the fitness estimate is more accurate for low order schema since more chromosomes belong to the schema.
- It is possible to obtain the fitness of a schema contained in a population based on the fitness values of the chromosomes in that population.

Schemata Fitness

Example

- Suppose the following chromosomes with their fitness values:

Chromosome	Fitness
101	4
110	3
011	2

- The fitness for the following schemata can be calculated:

Schema	Fitness of schema
***	$(4+3+2)/3 = 3$
**0	$3/1 = 3$
1	$(3+2)/2 = 2.5$

Schemata Fitness

- The schema theorem can be interpreted using the following equation, which relates the schema fitness of the current generation to the expected chromosomes in new generation:

$$M(H, t+1) \geq M(H, t)\frac{f(H)}{F_{tot}}(1 - P_c - P_m)$$

  - $M(H, t)$ is the number of the chromosomes in population t that matches the schema H,
  - $f(H)$ is the average fitness of chromosomes with schema H,
  - F_{tot} is the average fitness of the whole population,
  - P_c, P_m are the probabilities of crossover and mutation.

Genetic Operators

- They are processes by which evolution changes the composition of a population.
- In natural evolution there are three major forces: natural selection, mating, and mutation.

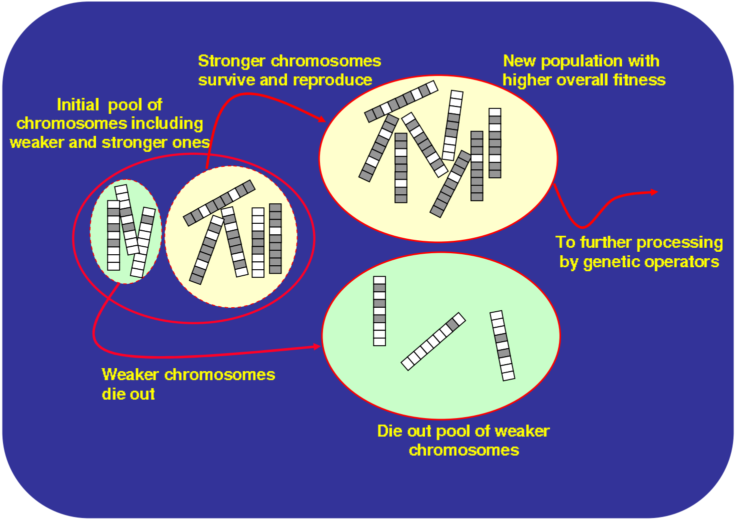
Three Basic Genetic Operators in Artificial Evolution

- Selection
- Crossover
- Mutation

Selection

- This procedure is applied to select the individuals that participate in the reproduction process to give birth to the next generation.
- In general, selection operators are stochastic, probabilistically selecting good solutions and removing bad ones based on the evaluation given them by the objective function.

Process of Selection in GA



Selection: Several Heuristics

Several Selection Operators

- Elitist model, where the top 10 to 20 individuals of the population are chosen for further processing,
- Ranking model, where each member of the population is ranked based on its fitness value,
- Roulette wheel procedure, where each individual i is assigned a probability p to be chosen for reproduction.

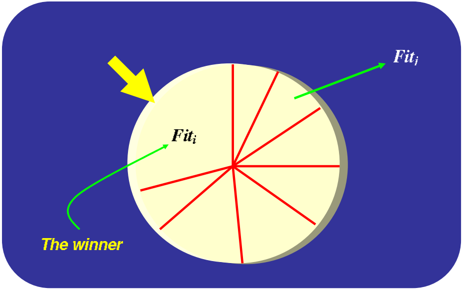
Selection: Roulette wheel

- Individuals with higher performance have higher chance to be selected.

The Performance of an Individual

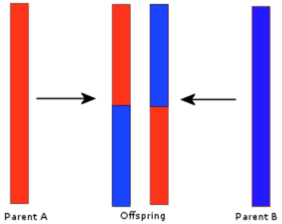
$$P_i = \frac{fit(v_i)}{F_{\text{Total}}}$$

- The sum of the fitness values characterizes a total fitness of the population.



Crossover

- Crossover is derived from the natural phenomenon of mating, but refers most specifically to genetic recombination.
- It creates two new individuals by combining two older ones.
- Most crossover operators randomly select a set of genes from each parent to form a child's genotype.



Crossover Process

- The GA selects two strings at random from the mating pool.
- It calculates whether crossover should take place using a parameter called the crossover probability.
- If no crossover is performed, the two selected strings are simply copied to the new population.
- If crossover does take place, then a crossover operator is applied to recombine the two genotypes.
- The objective is to produce new offspring with higher cumulative fitness.

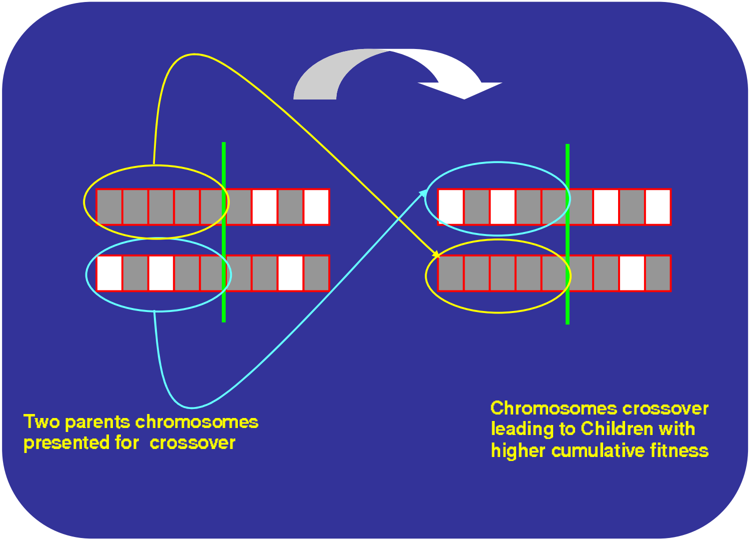
Crossover Operators

Several Crossover Operators

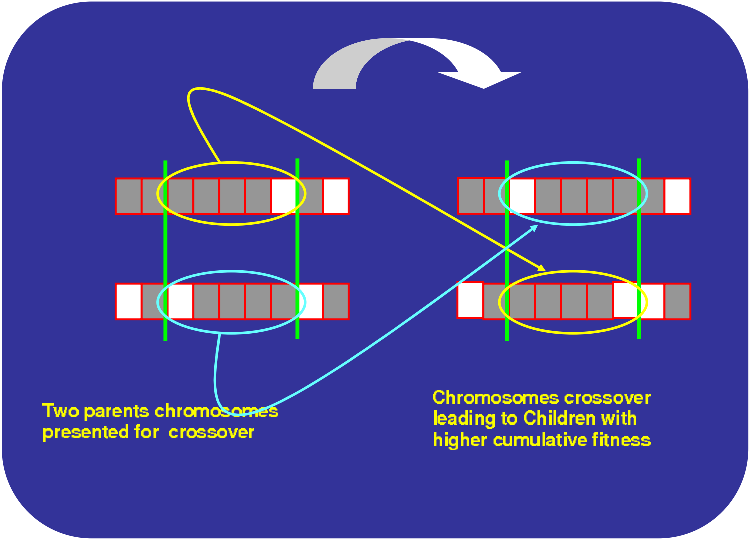
- Single-point crossover
- Uniform crossover
- Multi-point crossover
- Heuristic crossover

- The simplest one is the single-point crossover.
- It works by randomly selects a cut-off point.
- The parts that are cut off are then switched over and recombined to make two new chromosomes.

Single-Point Crossover



Multi-Point Crossover



Mutation

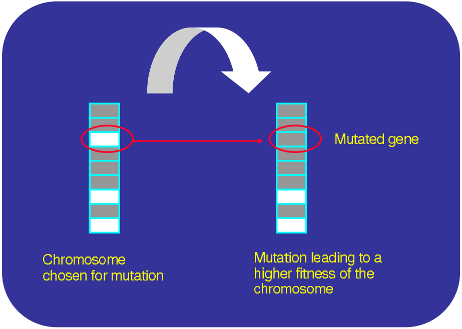
- Sometime, and depending on the initial population, there may not be enough variety of strings to ensure the GA exploits the entire problem space.
- The mutation operator introduces random changes into the chromosomes.
- It selects genes in an individual at random and changes the allele.
- It ensures the diversity in the population.
- Mutation operator provides a good mechanism to escape from the local minima.

Mutation (cont.)

Several Types of Mutation Operators

- Binary mutation
- Non-uniform mutation
- Uniform mutation
- Boundary mutation

- In the binary mutation, the GA selects, randomly, a gene in a chromosome and changes its value to a new one.



Steps for Implementing GA

Steps 1-3

- Step 1: Encode the variables of the algorithm as binary chromosomes where v_i $(i = 1, 2, \cdots, p)$ and p is the population size of the possible solutions.
- Step 2: Initialize population of chromosomes.
- Step 3: Perform the following steps until the predefined condition is achieved:

Steps for Implementing GA (cont.)

Step 3

- Step 3 (cont.):
  - 3.1: Evaluate the fitness function values, $fit(v_i)$, for each chromosome $v_i (i = 1, 2, \cdots, p)$
  - 3.2: Select a new generation
  - 3.3: Calculate the total fitness of the population:

$$F = \sum_{i=1}^{p} fit(v_i)$$

  - 3.4: Calculate the probability of selection, P_i, for each chromosome v_i $(i = 1, 2, \cdots, p)$

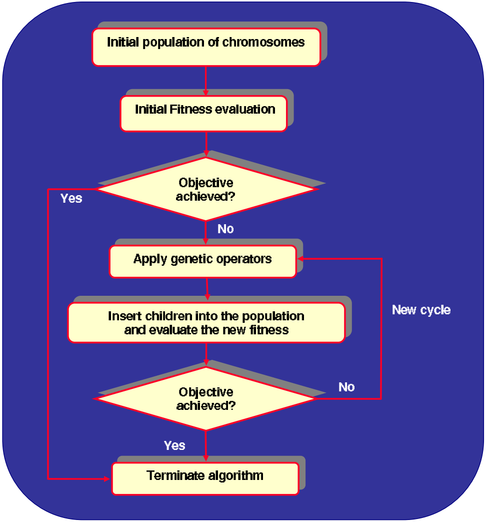
$$P_i = \frac{fit(v_i)}{F}$$

Steps for Implementing GA (cont.)

Step 3-4

- Step 3 (cont.):
  - 3.5: Generate a random float number r belonging to the interval $[0, 1]$
  - 3.6: If $r \leq q_1$ then select v_1; otherwise select $v_i (2 \leq r \leq q_i)$ such that $q_{i-1} \leq r \leq q_i$.
  - 3.7: Apply genetic operators:
    - Crossover with probability P_c
    - Mutation with probability P_m
  - 3.8: Choose the new offspring as the current population.
- Step 4: Go back to step 3 if optimization requirement is not attained.

Schematic Representation of GA



Representation of Chromosomes (Coding)

- Determines how to encode the independent parameter into a chromosome.
- Enables the chromosome to carry all the essential information about that parameter.

Three Main Encoding Mechanisms

1. Binary coding
2. Floating point coding
3. Gray coding

Encoding

Binary Coding

- A binary vector of length n bits is used as a chromosome to represent a real value of each parameter x_i.
- Different parameters may have different length (n).
- The length of the chromosome depends on the precision required.

Encoding

Floating-Point Coding

- The content of chromosomes comes in the form of real numbers.
- Chromosomes are created by putting all the real numbers one after another.
- This means that the encoding and decoding steps do not have to take place.
- A simple crossover operator can change a number (instead of a bit).

Encoding

Gray Coding

- Map a decimal number to a string of binary digits.
- Only one digit has to be changed when increasing and decreasing the number by one.
- Minimized the Hamming Distances between Adjacent numbers.
- Based on the fact that any two consequent numbers n and $n+1$ differ by only a single bit flip.

Example

- Find x that maximizes the function $f(x)$:

$$f(x) = 20 + 100 * xcos(4\pi x) * exp(-2x)$$

Parameters

- The range of x is $[0, 1.5]$
- The required precision is given by 3 decimals.
- Population size $p = 10$
- Crossover probability $P_c = 0.25$
- Mutation probability $P_m = 0.1$

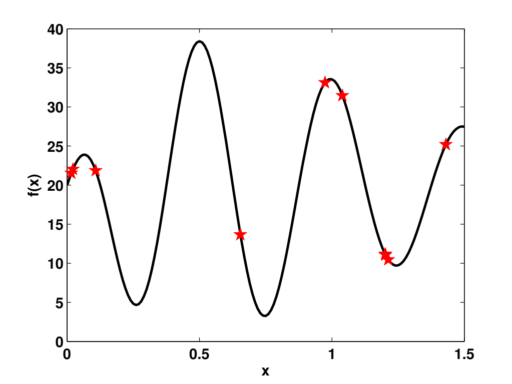
Example

Initial Population

Chromosome	Binary coding	Decimal value	Fitness	Selection probability
x_1	00010010100	0.1085	21.8025	0.1803
x_2	11001100111	1.2010	11.1218	0.0552
x_3	11001101001	1.2025	11.0231	0.0547
x_4	10100110001	0.9739	33.1448	0.1646
x_5	11001110111	1.2128	10.4288	0.0518
x_6	01101111101	0.65447	13.6220	0.0677
x_7	00000010110	0.0161	21.5290	0.1069
x_8	11110100000	1.4304	25.2480	0.1254
x_9	10110001011	1.0398	31.4024	0.1560
x_{10}	00000011110	0.0220	22.0240	0.1094

Initial Population

Graphical Illustration



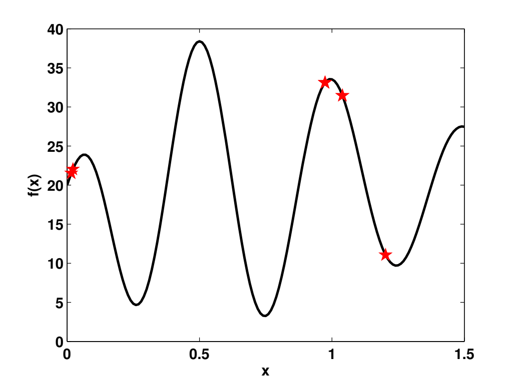
Example (cont.)

New Population after Applying Selection Operator

Chromosome	Binary coding	Decimal value
x_1	00000011110	0.0220
x_2	00000011110	0.0220
x_3	11001101001	1.2025
x_4	10100110001	0.9739
x_5	10110001011	1.0398
x_6	10110001011	1.0398
x_7	10100110001	0.9739
x_8	10110001011	1.0398
x_9	10100110001	0.9739
x_{10}	00000010110	0.0161

New Population after Applying Selection Operator

Graphical Illustration



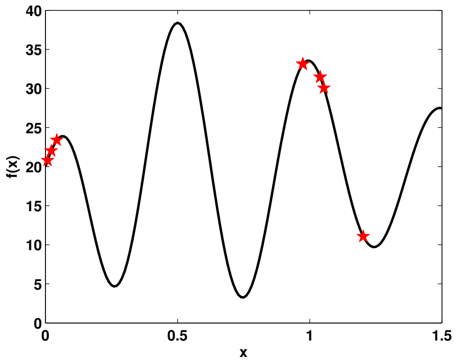
Example (cont.)

New Population after Crossover and Mutation

Chromosome	Binary coding	Decimal value	Fitness
x_1	00000001011 CO*	0.0081	20.7891
x_2	00000011110	0.0220	22.0240
x_3	11001101001	1.2025	11.0231
x_4	10100110001	0.9739	33.1448
x_5	10110011110 CO*	1.0537	29.9967
x_6	10110001011	1.0398	31.4024
x_7	10100110001	0.9739	33.1448
x_8	10110001011	1.0398	31.4024
x_9	10100110001	0.9739	33.1448
x_{10}	10110011110 M*	0.043	23.2514

New Population after Crossover and Mutation

Graphical Illustration



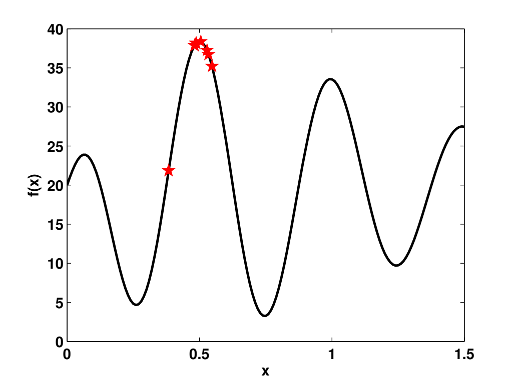
Example (cont.)

Population after 16 Generations

Chromosome	Binary coding	Decimal value	Fitness
x_1	01010010001	0.4814	37.8831
x_2	01011101011	0.5474	35.1638
x_3	01010010001	0.4814	37.8831
x_4	01010010001	0.4814	37.8831
x_5	01011011001	0.5342	36.6842
x_6	01010011001	0.4873	38.1542
x_7	01010101011	0.505	38.3936
x_8	01011010001	0.5283	37.2136
x_9	01000011001	0.3835	24.1274
x_{10}	01010011001	0.4873	38.1542

Population after 16 Generations

Graphical Illustration



Known Issues in GA

- Local minima and premature convergence
- Mutation interference
- Deception

Local Minima and Premature Convergence

- GA's are less prone to being trapped because of the effects of mutation and crossover.
- However, if a GA explores such a region extensively, it may be almost completely dominated by solutions within that region.
- A GA dominated by a set of identical or very similar solutions is often said to have converged.
- If a GA converges to a set of solutions within a local minimum then the GA is said to have converged prematurely.

Local Minima and Premature Convergence

- GA's are less prone to being trapped because of the effects of mutation and crossover.
- However, if a GA explores such a region extensively, it may be almost completely dominated by solutions within that region.
- A GA dominated by a set of identical or very similar solutions is often said to have converged.
- If a GA converges to a set of solutions within a local minimum then the GA is said to have converged prematurely.

Mutation Interference

- Mutation interference occurs when mutation rates in a GA are so high that solutions are so frequently mutated that the algorithm never manages to explore any region of the space thoroughly.
- Even if it finds good solutions, they tend to be rapidly destroyed.
- A GA experiencing mutation interference will probably never converge since its population is too unstable.

Deception

- In some cases, a problem space may lead a GA to converge naturally to a sub-optimal solution, which appears good.
- This is often the case when there are many decent solutions of similar form but a much better solution that has a radically different form.
- It may also be that the region of attraction around the global minima is very small, and there exist other local minima with much larger regions of attraction.
- Thus, these minima mislead the GA as to the form of the best solution in a manner known as deception.

Deception

- In some cases, a problem space may lead a GA to converge naturally to a sub-optimal solution, which appears good.
- This is often the case when there are many decent solutions of similar form but a much better solution that has a radically different form.
- It may also be that the region of attraction around the global minima is very small, and there exist other local minima with much larger regions of attraction.
- Thus, these minima mislead the GA as to the form of the best solution in a manner known as deception.

Multi-Objective GA

Multi-Objective Optimization Problem

- Many real-world optimization problems have two or more competing objectives.
- For example, a vehicle maker wishes to maximize crash resistance for safety and minimize vehicle weight for high fuel efficiency.

Two Conflicting Goals

- A vehicle design A may achieve high crash resistance at low fuel efficiency.
- Another vehicle design B may achieve high fuel efficiency at low crash resistance.

Multi-Objective Optimization Problem (cont.)

- None of these designs can be considered as a superior one if we have no preference in safety or economy criteria.
- For such multi-objective optimization problems, we have a set of optimal solutions (called Pareto-optimal set) instead of a single optimal solution.
- Given the optimal solution set, a human decision maker can choose one solution according to his or her preference.

Multi-Objective GA

- A GA is well-suited for the multi-objective optimization problems since it processes a set of solutions in parallel.

Several Multi-Objective GAs

- Nondominated sorting GA (Srinivas '94)
- Niched Pareto GA (Horn '94)
- Strength Pareto GA (Zitzler '99)
- Fast nondominated sorting GA (Deb '02)

Constraint Handling

Constraint Handling Using GA

- Most real-world optimization problems involve constraints.

Constrained Optimization Problem

A constrained optimization problem can be formulated as:

- Minimize $f(X)$ for $X = (x_1, \cdots, x_n)$
- subject to $\ell_i \leq i \leq u_i$, $1 \leq i \leq n$
- q inequality constraints $g_j(x) \leq 0$ $j = 1, \cdots, q$
- $m - q$ equality constraints $h_j(x) \leq 0$ $j = q + 1, \cdots, m$

Constraint Handling Using GA

Three Popular Methods for Handling Constrained Problems

Three popular methods handling constraints:

- Penalty Function Based Method
- Method based on preference feasible solutions over infeasible ones
- Method based on multi-objective optimization

Constraint Handling Using GA: Penalty Function Based Method

Penalty Function Based Method

- An infeasible individual is penalized based on its constraint violation.
- Penalty function combines objective function value and the constraint violation value to decide the fitness of each individual.
- Easy to implement.
- Requires some degree of parameter tuning to tailor for each problem.

Penalty Function Based Method

Different Types of Penalty Function

Several different types of penalty functions are available:

- Death penalty method: Infeasible individuals are not considered for selection.
- Static penalty method: Penalty is the weighted sum of constraint violation.
- Dynamic penalty method: Penalty also depends on the generation index.

Penalty Function Based Method

Death Penalty Method

- The greatest penalty that can be imposed on an infeasible solution.
- Doesn't exploit any information from the infeasible individuals to guide the search.
- May work well for the problems where feasible space is convex and covers a large portion of search space.

Penalty Function Based Method

Static Penalty Method

- Static penalty method modifies the objective function as:

$$obj(X) = f(X) + \sum_{j=1}^{m} r_j c_j(X)$$

  - where $f(X)$ is the original objective function,
  - r_i is penalty coefficient for constraint j,
  - $c_j(X)$ is degree of violation of constraint j corresponding to X.
- The performance highly depends on the proper penalty coefficient chosen for given constraints.

GA Applications

- Scheduling problems (e.g, job-shop scheduling, traveling salesman problem)
- Optimization of network topologies
- Resource allocation over a distributed system
- Electronic circuit design
- Aircraft design
- Game playing
- Training of fuzzy systems or artificial neural networks