SCS5107 Computational Intelligence

UNIT II

Evolutionary Algorithms

Theory of Evolution

The theory of evolution is the body of thought that examines evidence and uses it to deduce the consequences of the fact that evolution is going on all the time. You do not need to accept the theory of evolution in biology to do evolutionary computation. Evolutionary computation uses the ideas in the theory of evolution, asserting nothing about their validity in biology.

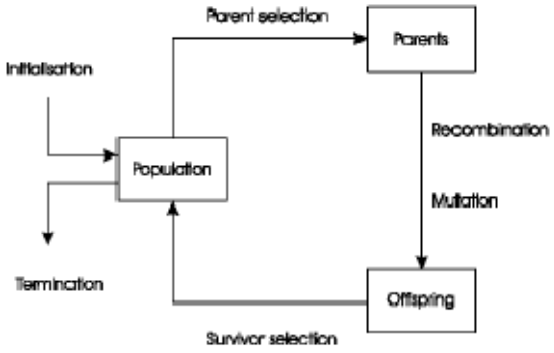
There are two opposing forces that drive evolution: variation and selection. Variation is the process that produces new alleles and, more slowly, genes. Variation can also change which genes are or are not expressed in a given individual. Selection is the process whereby some alleles survive and others do not. Variation builds up genetic diversity; selection reduces it.

Evolutionary computation operates on populations of data structures. It accomplishes variation by making random changes in these data structures and by blending parts of different structures. These two processes are called mutation and crossover, and together are referred to as variation operators. Selection is accomplished with any algorithm that favors data structures with a higher fitness score.
There are many different possible selection methods. “evolution is the result of survival of the fittest” is a pretty good description of many evolutionary computation systems. When we use evolutionary computation to solve a problem, we operate on a collection(population) of data structures (creatures).These creatures will have explicitly computed finesses used to decide which creatures will be partially or completely copied by the computer (have offspring)

Mutations of data structures can be “good” or “bad.” A good mutation is one that increases the fitness of a data structure. A bad mutation is one that reduces the fitness of a data structure. The representation used in a given example of Evolutionary algorithm is the data structure used together with the choice of variation operators. The data structure by itself is the chromosome

(or gene) used in the evolutionary computation. The fitness function is the method of assigning a heuristic numerical estimate of quality to members of the evolving population. In some cases, it decides only which of two structures is better without assigning an actual numerical quality.

General Scheme of EAs



A Typical Evolutionary Algorithm Cycle

Step 1: Initialize the population randomly or with potentially good ***solutions***.
Step 2: Compute the ***fitness*** of each individual in the population.
Step 3: Select parents using a ***selection procedure***.
Step 4: Create offspring by ***crossover*** and ***mutation*** operators.
Step 5: Compute the ***fitness*** of the new offspring.
Step 6: Select members of population to die using a ***selectionprocedure***.
Step 7: Go to Step 2 until termination criteria are met.

Biology

A gene is a sequence of DNA bases that code for a trait, e.g., eye color or ability to metabolize alcohol. An allele is a value of a trait. The eye color gene could have a blue allele or a hazel allele in different people. Evolution is the variation of allele frequencies in populations over time.

Global Optimization

Global optimization is the branch of applied mathematics and numerical analysis that deals with the optimization of single or maybe even
multiple, possible conflicting, criteria. These criteria are expressed as a set of mathematical functions F = {f1, f2, . . . , fn}, the so-called objective functions. The result of the optimization process is the set of inputs for which these objective functions return optimal values. The difference between optimization algorithms and search algorithms is that when performing a search algorithm.

we know the element xi we are looking for and just want to find its position in a structured set. In global optimization algorithms on the other hand we do not even know the characteristics of the xi beforehand and are only given some criteria which describe if a given configuration is good or not.

Local Maximum and Minimum

Definition 2 (Local Maximum). A (local) maximum $\hat{x}_l \in X$ of an objective function $f : X \mapsto \mathbb{R}$ is an input element with $f(\hat{x}_l) \geq f(x)$ for all x neighboring $\hat{x}_l$.

If $X \subseteq \mathbb{R}$, we can write:

$$\hat{x}_l : \exists \varepsilon > 0 : f(\hat{x}_l) \geq f(x) \; \forall x \in X, |x - \hat{x}_l| < \varepsilon \tag{1.1}$$

Definition 3 (Local Minimum). A (local) minimum $\check{x}_l \in X$ of an objective function $f : X \mapsto \mathbb{R}$ is an input element with $f(\check{x}_l) \leq f(x)$ for all x neighboring $\check{x}_l$.

If $X \subseteq \mathbb{R}$, we can write:

$$\check{x}_l : \exists \varepsilon > 0 : f(\check{x}_l) \leq f(x) \; \forall x \in X, |x - \check{x}_l| < \varepsilon \tag{1.2}$$

Definition 4 (Local Optimum). An (local) optimum $x_l^\star \in X$ of an objective function $f : X \mapsto \mathbb{R}$ is either a local maximum or a local minimum (or both).

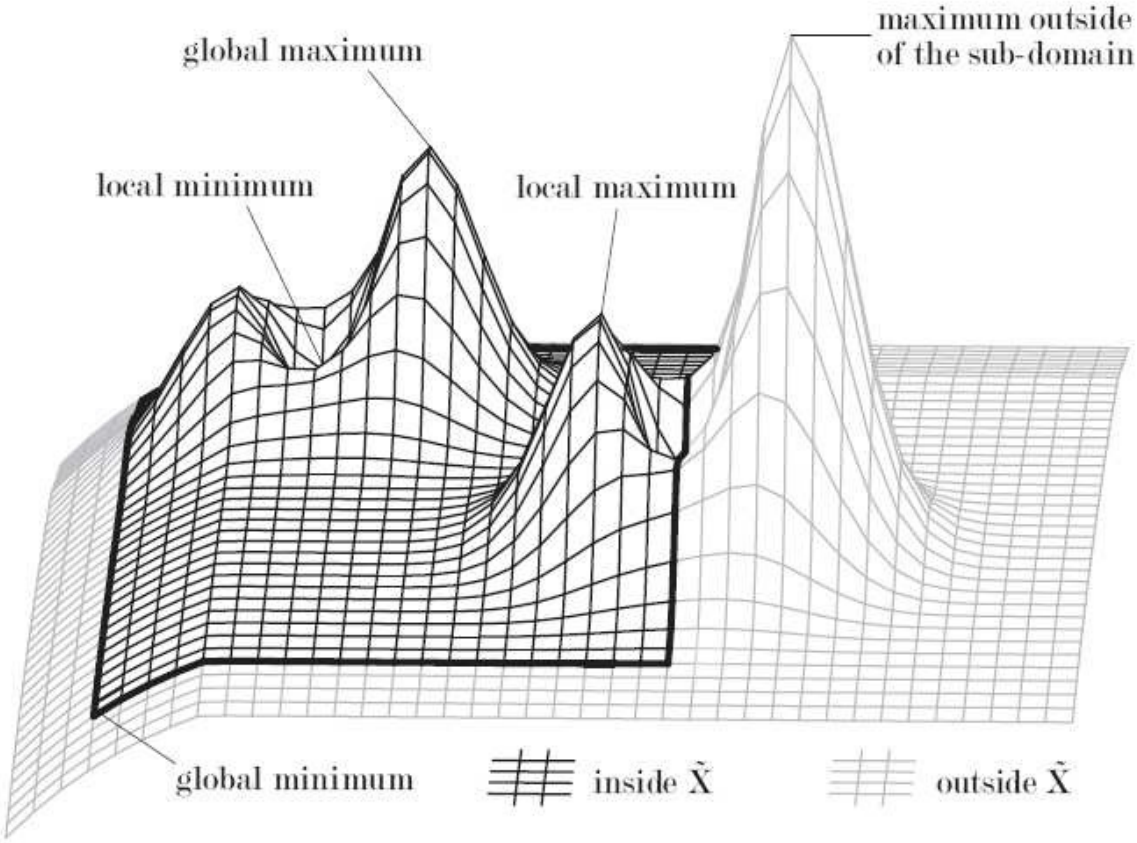
Global Maximum and Minimum

Definition 5 (Global Maximum). A global maximum $\hat{x} \in X$ of an objective function $f : X \mapsto \mathbb{R}$ is an input element with $f(\hat{x}) \geq f(x) \forall x \in X$.

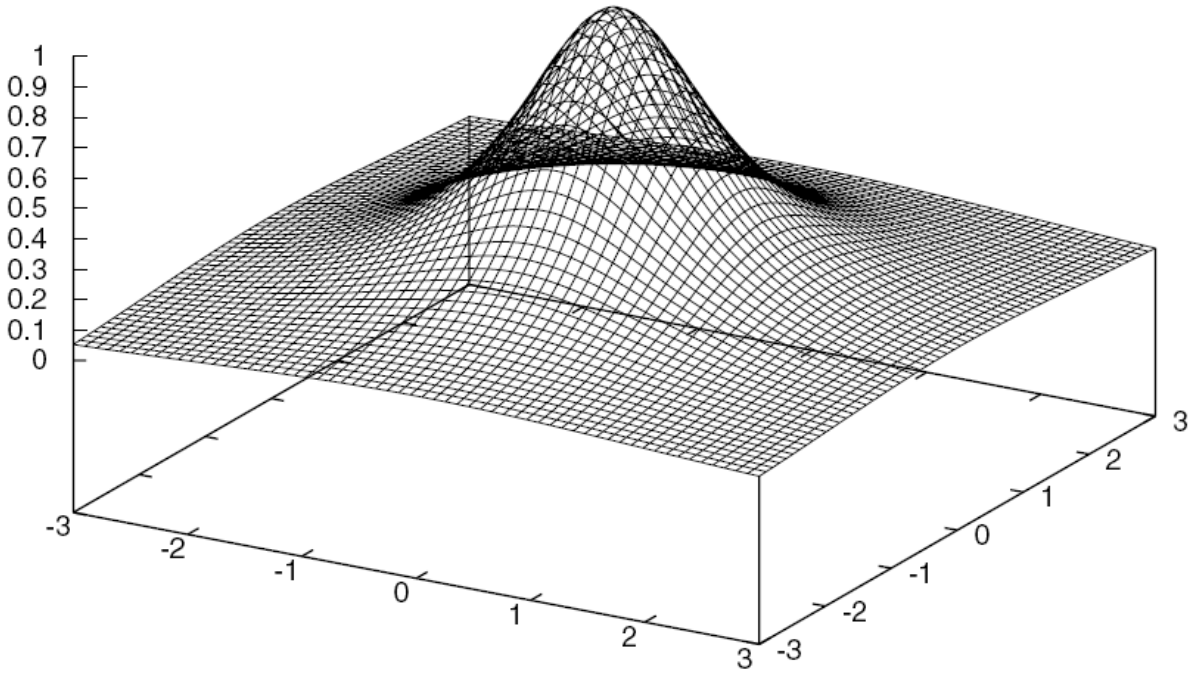
Definition 6 (Global Minimum). A global minimum $\check{x} \in X$ of an objective function $f : X \mapsto \mathbb{R}$ is an input element with $f(\check{x}) \leq f(x) \forall x \in X$.

Definition 7 (Global Optimum). A global optimum $x^\star \in X$ of an objective function $f : X \mapsto \mathbb{R}$ is either a global maximum or a global minimum (or both).

Global and Local Optima



Simple Functions



History of Evolutionary Algorithms

Several efforts were started in parallel during 1960s.Evolutionary Strategies (Berlin Technical University). Genetic Algorithms (University of Michigan).Evolutionary Programming (UCLA). During 1990s the above communities agreed to the term "Evolutionary Computation

Evolutionary Strategies

At the Technical University Berlin, Rechenberg and Schwefel (1965 paper) began formulating ideas about how evolutionary processes could be used to solve difficult real-valued parameter optimization problems. From these early ideas emerged a family of algorithms called "*evolution strategies*" which today represent some of the most powerful evolutionary algorithms for function optimization.

Evolutionary Programming

At UCLA during the same period Fogel (1966 paper) saw the potential of achieving the goals of artificial intelligence via evolutionary techniques.
These ideas were initially explored in a context in which intelligent agents were represented as finite state machines, and an evolutionary framework called "*evolutionary programming*" was developed which was quite effective in evolving better finite state machines (agents) over time.

Genetic Algorithms

At the University of Michigan, Holland (1962 paper) saw evolutionary processes as a key element in the design and implementation of robust adaptive systems, capable of dealing with an uncertain and changing environment.
His view emphasized the need for systems which self adapt over time as a function of feedback obtained from interacting with the environment in which they operate.
This lead to an initial family of "reproductive plans" which formed the basis for what we call "*simple genetic algorithms*" today.

Components of Evolutionary Algorithms

- Representation (definition of individuals)
- Evaluation function (or fitness function)
- Population
- Parent selection mechanism
- Variation operators, recombination and mutation
- Survivor selection mechanism (replacement)

Representation

The first step in defining an EA is to link the "real world" to the "EA world". Objects forming possible solution within the original problem context are referred to as **phenotypes**, while their encoding in the EA are called **genotypes**. Within the EC literature many synonyms can be found. **Candidate solution** and **individuals** denote points in the **phenotype space**. **Chromosome** and **individuals** denote points in the **genotype space**. A placeholder is commonly called a variable, a **locus**, or –in a biology-oriented terminology – a **gene**. An object in such a place can be called a value or an **allele**.

Representation Example

An individual is typically described as a fixed length vector of L features which are chosen presumably because of their (potential) relevance to estimating an individual's fitness.
For example,
– <hair color, eye color, skin color, height, weight>

Evaluation Function

The role of the evaluation function is to represent the requirements the population should adopt to. Technically, it is a function or procedure that assigns a quality measure to genotypes. The evaluation function is commonly called the **fitness function** in EC. This might cause a counterintuitive terminology if the original problem requires minimization, because the term fitness is usually associated with maximization.

Parent Selection Mechanism

The role of parent selection is to distinguish among individuals based on their quality, and, in particular, to allow the better individuals to become parents of the next generation. An individual is a **parent** if it has been selected to undergo variation in order to create offspring. In EC, parent selection is typically probabilistic.

Variation Operators

The role of **variation operators** is to create new individuals from old ones.
In the corresponding phenotype space this amounts to generating new candidate solutions. Variation operators in EC are divided into two types.

Mutation
Crossover

Mutation

Mutation is a unary variation operator. It is applied to one genotype and delivers a (slightly) modified mutant, the **child** or **offspring**. A mutation operator is

always stochastic: its output – the child – depends on the outcomes of a series of random choices.

Crossover

A binary variation operators is called **recombination** or **crossover**. This operator merges information from two parent genotypes into one or two offspring genotypes. Like mutation, recombination is a stochastic operator. The choices of what parts of each parent are combined and how this is done, depend on random drawings. The principle behind recombination is simple – by mating two individuals with different but desirable features, we can produce an offspring that combines both of those features.

Crossover OR Mutation?

Exploration: Discovering promising areas in the search space, i.e. gaining information on the problem
Exploitation: Optimizing within a promising area, i.e. using information

- There is co-operation AND competition between them
- Crossover is explorative, it makes a big jump to an area somewhere "in between" two (parent) areas
- Mutation is exploitative, it creates random small diversions, thereby staying near (in the area of) the parent
- Only crossover can combine information from two parents
- Only mutation can introduce new information (alleles)

Survival Selection Mechanism

The role of survival selection (**replacement**) is to distinguish among individuals based on their quality. It is similar to parent selection, but it is used I
a different stage of the evolutionary cycle. In contrast to parent selection, which is typically stochastic, survivor selection is often deterministic.

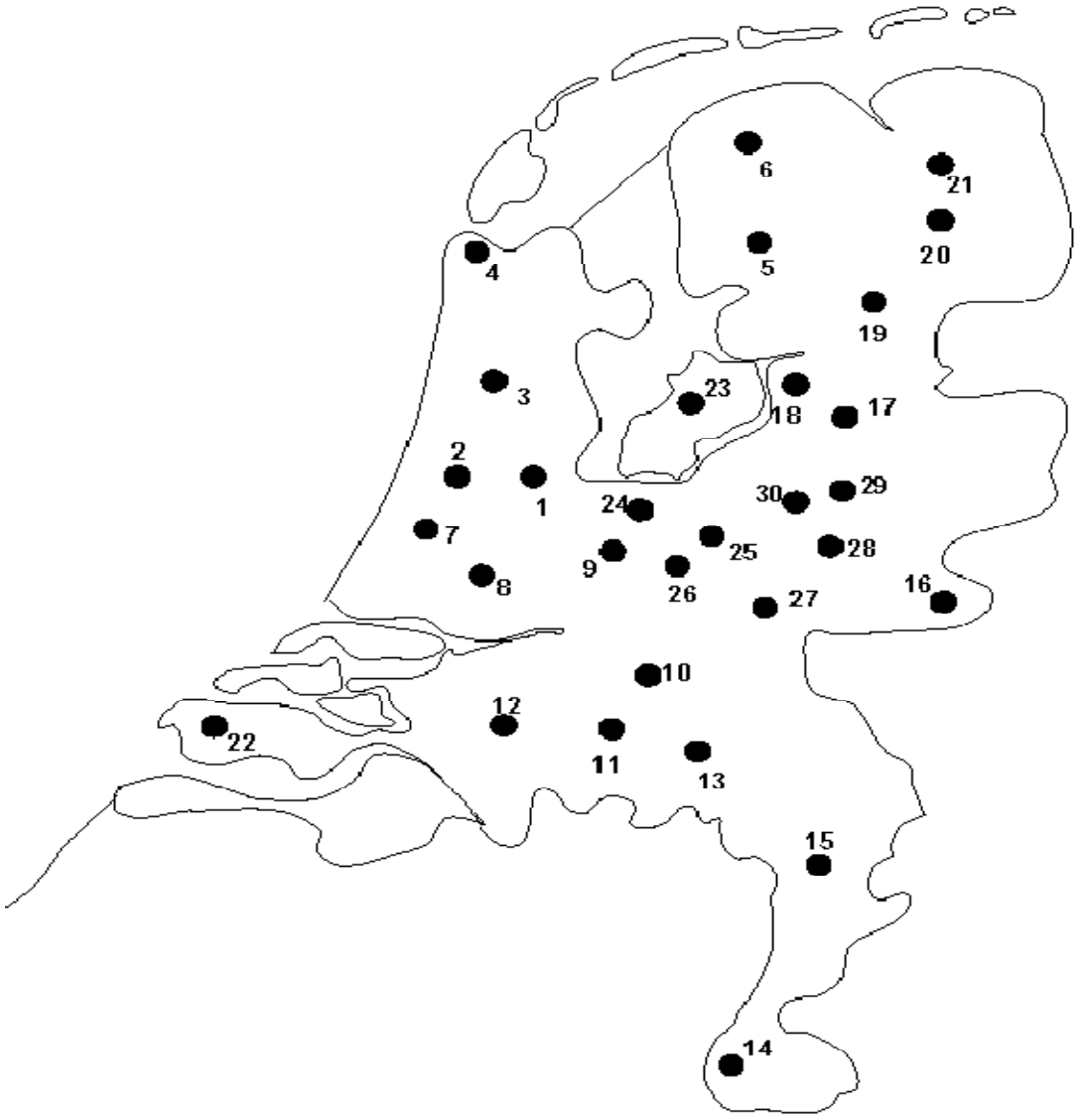
Initialization and Termination

Initialization is kept simple in most EA applications, the first population is seeded by randomly generated individuals. Problem specific heuristics can also be used in this step to create an initial population with higher fitness. The following options are used for termination. The maximally allowed CPU time elapses. The fitness

improvement remains under a threshold value for a given period of time. The population diversity drops under a given threshold.

Traveling Sales Person Problem

Given a number of cities and the costs of traveling from one city to any other city, what is the cheapest round-trip route that visits each city exactly once and then returns to the starting city?



Permutation Representation: TSP

Problem:

Given n cities. Find a complete tour with minimal length

Encoding:

Label the cities 1, 2, … , *n.* One complete tour is one permutation (e.g. for n =4 [1,2,3,4], [3,4,2,1] are OK).Search space is BIG for 30 cities there are 30! ≈ 1032 possible tours

TSP: Nearest Neighbor

A B C D E F G H
A 0 8 3 1 4 9 3 6
B 8 0 5 10 11 4 3 6
C 3 5 0 8 7 1 5 12
D 1 10 8 0 9 11 6 4
E 4 11 7 9 0 5 17 3
F 9 4 1 11 5 0 4 1
G 3 3 5 6 17 4 0 7
H 6 6 12 4 3 1 7 0

Start with A: A – D – H – F – C – B – G – E Cost?
Start with E: E – H – F – C – A – D – B – G Cost?
Start with G: G – B – F – H – E – A – D – C Cost?

Initialize the Population

Candidate Solutions
A C B F H D E G
H B G E A C D F
A H G C B D F E
E G B C D H F A
F H A D C B E G
C D B A H E G F

Candidate Solutions	Fitness
A C B F H D E G	43
H B G E A C D F	52
A H G C B D F E	49
E G B C D H F A	47
F H A D C B E G	49
C D B A H E G F	56

	A	B	C	D	E	F	G	H
A	0	8	3	1	4	9	3	6
B	8	0	5	10	11	4	3	6
C	3	5	0	8	7	1	5	12
D	1	10	8	0	9	11	6	4
E	4	11	7	9	0	5	17	3
F	9	4	1	11	5	0	4	1
G	3	3	5	6	17	4	0	7
H	6	6	12	4	3	1	7	0

Order 1 Crossover Example

Copy randomly selected set from first parent

Copy rest from second parent in order 1,9,3,8,2

Crossover Example

- We selected the parents in the previous slides

Parent 1	A	C	B	F	H	D	E	G	43
Parent 2	F	H	A	D	C	B	E	G	49

- Let's do crossover to produce two offspring. Suppose the crossover points are 3 and 5

Offspring 1	D	C	B	F	H	E	G	A
Offspring 2	F	H	A	D	C	E	G	B

Insert Mutation for Permutations

- Pick two allele values at random
- Move the second to follow the first, shifting the rest along to accommodate
- Note that this preserves most of the order and the adjacency information

Swap Mutation for Permutations

Pick two alleles at random and swap their positions

Preserves most of adjacency information (4 links broken), disrupts order more

Mutation Example

Perform swap mutation on the two offspring produced in the previous slide.

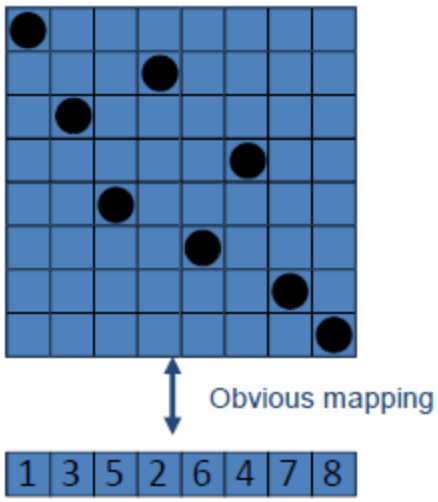
Offspring 1	D C B F H E G A
Offspring 2	F H A D C E G B

For Offspring 1, swap 2nd and 4th gene.

For Offspring 2, swap 2nd and 6th gene.

Offspring 1	D F B C H E G A	55
Offspring 2	F E A D C H G B	40

The 8-Queen Problem: Representation



Spring 2011

The 8-Queen Problem: Fitness

Penalty of one queen: the number of queens she can check.
Penalty of a configuration: the sum of the penalties of all queens.
Note: penalty is to be minimized
Fitness of a configuration:
inverse penalty to be maximized

The 8-Queen Problem: Mutation

- Small variation in one permutation, e.g.:
  - swapping values of two randomly chosen positions

The 8-Queens Problem: Crossover

Combining two permutations into two new permutations:

- choose random crossover point
- copy first parts into children
- create second part by inserting values from other parent:
  - in the order they appear there
  - beginning after crossover point
  - skipping values already in child

Evolutionary Algorithm Cycle

Step 1: Initialize the population randomly or with potentially good ***solutions***.
Step 2: Compute the ***fitness*** of each individual in the population.
Step 3: Select parents using a ***selection procedure***.
Step 4: Create offspring by ***crossover*** and ***mutation*** operators.
Step 5: Compute the ***fitness*** of the new offspring.
Step 6: Select members of population to die using a ***selection procedure***.
Step 7: Go to Step 2 until termination criteria are met.

Selection

Selection is one of the main operators in EAs, andrelates directly to the Darwinian concept of survival of the fittest. A new population of candidate solutions is selected at the end of each generation to serve as the population of the next generation. The selection operator should ensure that good individuals do survive to next generations.
Reproduction: Offspring are created through the application of crossover and/or mutation operators.

Selective Pressure

Selection operators are characterized by their selective pressure, also referred to as the takeover time, which relates to the time it requires to produce a uniform population. It is defined as the speed at which the best solution will occupy the entire population by repeated application of the selection operator alone. An operator with a high selective pressure decreases diversity in the population more rapidly than operators with a low selective pressure, which may lead to premature convergence to suboptimal solutions. A high selective pressure limits the exploration abilities of the population.

Selection Procedure

Selection in evolutionary algorithms is the process of choosing which individuals reproduce offspring and which individuals survive to the next generation. When selection is used to choose which individuals reproduce, the process is referred to as ***pre-selection***(parent(s) selection).When it is used to select the individuals that survive to the next generation it is called ***post-selection***(survival selection).

• ***Deterministic selection*** tends to behave more like greedy hill climbing algorithms and exploits the nearest areas with promising solutions.
• ***Probabilistic selection*** schemes are more exploratory and search the landscape. Schemes based on exploration are said to have a low selection pressure, while schemes based on exploitation are said to have greater selection pressure.In other words, selection pressure is a vague measure of how often more fit individuals are selected to reproduce and/or live to the next generation. Selection schemes can be further categorized into generational or steady-state schemes. A selection scheme is ***generational*** when the entire current population is replaced by its offspring to create the next generation A scheme is referred to as ***steady-state*** when a selected few offspring replace a few members of the current generation to form the next generation.

Selection Schemes

- Fitness Proportional
- Rank Selection
- Tournament Selection
- Truncation
- Elitist
- Uniform Stochastic

Parent Selection Mechanism

Assigns variable probabilities of individuals acting as parents depending on their fitness. Usually probabilistic high quality solutions more likely to become parents than low quality. Even worst in current population usually has nonzero probability of becoming a parent. This *stochastic* nature can aid escape from local optima.

Fitness Proportional Selection

Proportional selection, proposed by Holland, biases selection towards the most fit individuals. A probability distribution proportional to the fitness is created, and individuals are selected by sampling the distribution. Because selection is directly proportional to fitness, it is possible that strong individuals may dominate in producing offspring, thereby limiting the diversity of the new population. This is known as **premature convergence**. In other words, proportional selection has a high selective pressure. When fitness values are all very close together, there is almost no selection pressure. Therefore, later in a run, when some convergence has taken place and the worst individuals are gone, the performance only increases very slowly. Also known as Roulette Wheel Selection.

Rank Based Selection

Attempts to remove problems of FPS by basing selection probabilities on *relative* rather than *absolute* fitness. Rank population according to fitness and then base selection probabilities on rank where fittest has rank m and worst rank 1.This imposes a sorting overhead on the algorithm, but this is usually negligible compared to the fitness evaluation time. Selection is independent of actual fitness values, with the advantage that the best individual will not dominate in the selection process. It preserves a constant selection pressure by sorting the population on the basis of fitness, and then allocating selection probabilities to individuals according to their rank, rather than according to their actual fitness values.

Tournament Selection

FP and R selection methods and the algorithms used to sample from their probability distribution relied on a knowledge of the entire population.

In certain situations, if the population is very large, or if the population is distributed in some way (perhaps on a parallel system), obtaining this knowledge is either highly time consuming or at worst impossible.

In other cases, there might not be a universal fitness definition at all. For instance, think of an application evolving game playing strategies. In this case we might not be able to quantify the strength of a given strategy but we can compare any two of them by simulating a game played by these strategies as opponents. Pick *nts* members at random then select the best of these.

Repeat to select more individuals. Inherits the advantage of rank selection. Does not require global reordering. For crossover with two parents, tournament. selection is done twice, once for the selection of each parent. Provided that the tournament size, nts , is not too large, tournament selection prevents the best individual from dominating, thus having a lower selection pressure. On the other hand, if nts is too small, the chances that bad individuals are selected increase.

Thus, the the selective pressure is directly related to *nts .If nts = ns , the best individual will always* be selected, resulting in a very high selective pressure. On the other hand, if *nts = 1,* random selection is obtained. There also exists a non-deterministic variant of this selection where this is not necessarily the case. Therefore, a probability p is defined. The best individual in the tournament is selected with probability p, the second best with probability p(1 −p), the third best with probability p(1 − p)2 and so on.

Tournament selection is perhaps the most widely used selection operators in the modern applications of EAs, due to its extreme simplicity and the fact that the selection pressure is easy to control by varying the tournament size.

Random Selection

Random selection is the simplest selection operator, where each individual has the same probability *to be selected. No fitness* information is used, which means that the best and the worst individuals have exactly the same probability of surviving to the next generation. Random selection has the lowest selective pressure. Random selection returns elements by chance. A possible preceding fitness assignment process as well as the objective values of the individuals play no role at all. This hinders the optimization algorithm to follow any gradient in the fitness landscape – it is effectively turned into a random walk.

• Random selection is thus not applied exclusively, but can serve as mating selection scheme in conjunction with a separate environmental selection.
• It maximally preserves the diversity and can be a good choice if used to pick elements from an optimal set.

Survivor Selection

Most EAs use fixed population size so need a way of going from (parents + offspring) to next generation .Often deterministic Fitness based : e.g., rank parents+ offspring and take best Age based: make as many offspring as parents and delete all parents. Sometimes do combination (elitism)

Age-Based and Fitness-Based Replacement

The basis of these schemes is that the fitness of individuals is not taken into account during the selection of which individuals to replace in the population, rather they are designed so that each individual exists in the population for the same number of EA iterations. A wide number of strategies have been proposed for choosing which m of the m + l parents and offspring should go forward to the next EA iteration.

Replace Worst/Truncation

In this scheme the worst lambda members of the population are selected for replacement. Although this can lead to very rapid improvements in the mean population fitness, it can also lead to premature convergence as the population tends to rapidly focus on the fittest member currently present.

Elitism

Elitism refers to the process of ensuring that the best individuals of the current population survive to the next generation. The best individuals are copied to the new population without being mutated. The more individuals that survive to the next generation, the less the diversity of the new population.

Elitist Selection

In elitism at least one copy of the best individual in the population is always passed

onto the next generation. The main advantage is that convergence is guaranteed (i.e., if the global maximum is discovered, the EA converges to the maximum).
– By the same token, however, there is a risk of being trapped in a local maximum.

Hall of Fame

The hall of fame is a selection scheme similar to the list of best players of an arcade game. For each generation, the best individual is selected to be inserted into the hall of fame. The hall of fame will therefore contain an archive of the best individuals found from the first generation. The hall of fame can be used as a parent pool for the crossover operator, or, at the last generation, the best individual is selected as the best one in the hall of fame.

Elitism

This scheme is commonly used in conjunction with age-based and stochastic fitness-based replacement schemes, in an attempt to prevent the loss of the current fittest member of the population. In essence a trace is kept of the current fittest member, and it is always kept in the population. The main advantage is that convergence is guaranteed (i.e., if the global maximum is discovered, the EA converges to the maximum). By the same token, however, there is a risk of being trapped in a local maximum.

No Free Lunch Theorem

Wolpert and Macready published a paper with a very strong title: “No Free Lunch Theorems for Optimization”. The key contents of the paper can be quoted as follows: For both static and time dependent optimization problems, the average performance of any pair of algorithms across all possible problems.

Genetic Algorithms

In EP and ES schemes, individuals die off only when replaced by younger individuals with higher fitness. This results in a significant loss of diversity in the population and can increase the likelihood of becoming trapped on a false peak. One way to handle this problem is to allow new individuals to replace existing individuals with higher fitness. A more direct method is to use generational model

in which parent survive for exactly one generation and are completely replaced by their offspring. This is the form that standard GA take and also the form that (μ , λ) – ES model take.

GA also implement the biological notion of fitness in a somewhat different fashion than we have seen so far (fitness proportional scheme).The EP and ES systems all used objective fitness to determine which offspring survive to adulthood. However, once in the parent pool, there is no additional bias with respect to which individuals get to reproduce – all parents have an equal chance. Another important difference between Gas and the other models is the way in which offspring are produced. The basic idea is that offspring inherit gene values from more than one parent. This mixing of parental gene values along with an occasional mutation provides the potential for a much more aggressive exploration of the space.

Crossover provides an additional source of variation involving larger initial steps, improving the initial rate of convergence. Since crossover does not introduce new gene values, its influence diminishes as the population becomes more homogenous, and the behavior of GA with crossover becomes nearly identical to a GA with no crossover.

Universal Genetic Code

Holland emphasized the importance of a universal string-like "genetic" representation to be used internally by a GA to represent the genome of an individual. He initially suggested a binary string representation in which each bit internally is viewed as a gene, and the mapping to the external phenotype left unspecified and problem specific. Crossover and mutation operate as before at the gene level except at a much finer level of granularity. In the case of a binary representation, mutation simplifies to a "bit flipping" operator.

Example (Goldberg)

- Simple problem: max x2 over {0,1,…,31}
- GA approach:

– Representation: binary code, e.g. 01101 ↔13
– Population size: 4
– 1-point crossover, bitwise mutation
– Roulette wheel selection

– Random initialization
• We show one generational cycle done by hand

x^2 Example: Selection

String no.	Initial population	x Value	Fitness $f(x) = x^2$	$Prob_i$	Expected count	Actual count
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
Average			293	0.25	1.00	1
Max			576	0.49	1.97	2

x^2 Example: Crossover

String no.	Mating pool	Crossover point	Offspring after xover	x Value	Fitness $f(x) = x^2$
1	0 1 1 0 | 1	4	0 1 1 0 0	12	144
2	1 1 0 0 | 0	4	1 1 0 0 1	25	625
2	1 1 | 0 0 0	2	1 1 0 1 1	27	729
4	1 0 | 0 1 1	2	1 0 0 0 0	16	256
Sum					1754
Average					439
Max					729

x^2 Example: Mutation

String no.	Offspring after xover	Offspring after mutation	x Value	Fitness $f(x) = x^2$
1	0 1 1 0 0	1 1 1 0 0	26	676
2	1 1 0 0 1	1 1 0 0 1	25	625
2	1 1 0 1 1	1 1 0 1 1	27	729
4	1 0 0 0 0	1 0 1 0 0	18	324
Sum				2354
Average				588.5
Max				729

Evolutionary Programming

First developed by Lawrence Fogel in 1966 for use in pattern learning. Real-valued approach to problem encoding. Models only the behavioral linkage between parents and their offspring, rather than seeking to emulate specific genetic operators from nature such as the encoding of behavior in a genome and recombination by genetic crossover.

Evolutionary Programming Procedure

1. Initialize the population
2. Expose the population to the environment
3. Calculate the fitness for each member
4. Randomly mutate each “parent” population member
5. Evaluate parents and children
6. Select members of new population
7. Go to step 2 until some condition is met

Population Initialization

Component variables are usually real-valued .Dynamic range constraints usually exist and are observed. Random initial values within dynamic ranges are used. Population is often a few dozen to a few hundred.

EP Mutation Process

The process of mutation is illustrated in equation 3.2:

$$p_{i+k,j} = p_i + N(0, \beta_j \phi_{p_i} + z_j), \forall_j = 1..., n, \quad (3.2)$$

where

$p_{i,j}$ is the j^{th} element of the i^{th} organism

$N(\mu, \sigma^2)$ is a Gaussian random variable with mean μ and variance σ^2

ϕ_{p_i} is the fitness score for p_i

β_j is a constant of proportinality to scale ϕ_{p_i}

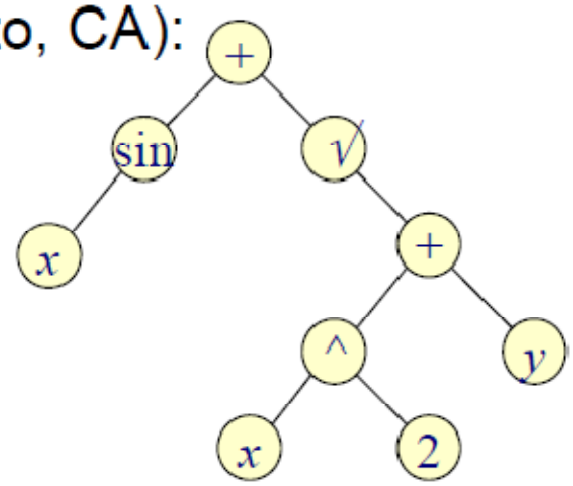
z_j represents an offset

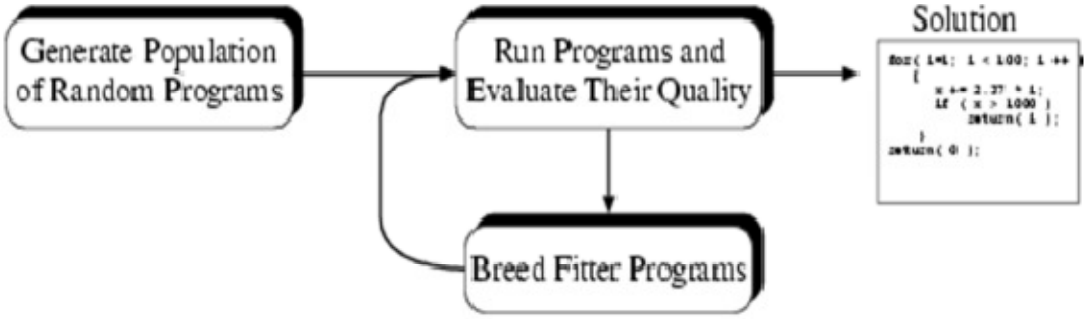
For the function used in the example, it has been shown that the optimum rate of convergence is represented by $\sigma = \frac{1.224\sqrt{f(x)}}{n}$, where n is the number of dimensions (Bäck and Schwefel 1993).

Genetic Programming

- Evolves a population of computer programs represented by trees
- J. Koza 1989 (Palo Alto, CA):

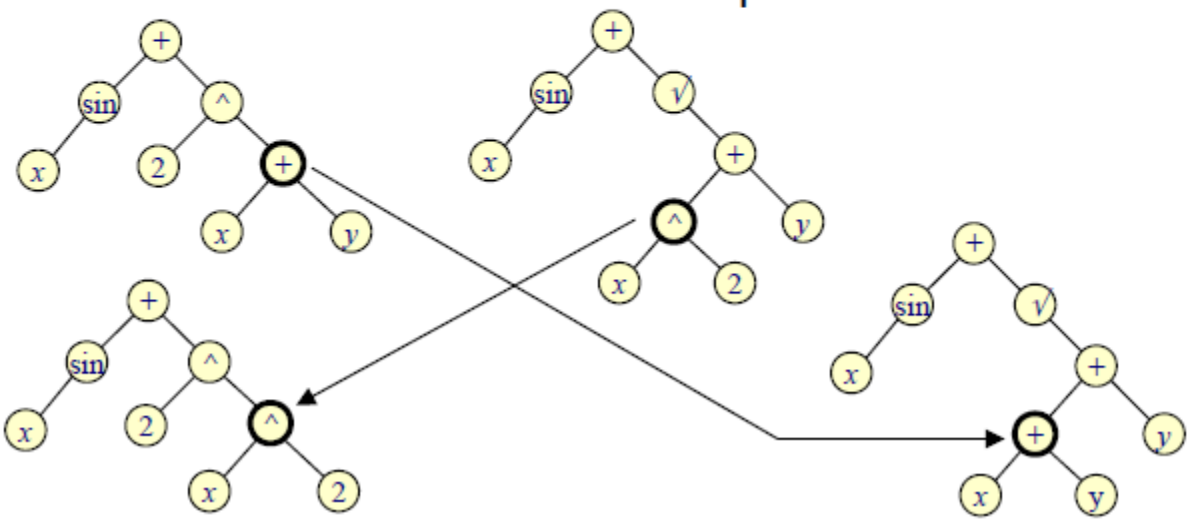
$$\sin(x) + \sqrt{x^2 + y}$$





GP: Crossover

- Given two parents crossover randomly selects a crossover point in each parent tree and swaps the sub-trees rooted at the crossover points.



GP: Mutation

- *Mutation* randomly selects a mutation point in a tree and substitutes the sub-tree rooted there with a randomly generated sub-tree
- Mutation is sometimes implemented as crossover between a program and a newly generated random program (“*headless chicken” crossover*).