CHAPTER 4

Evolutionary Computing

Syllabus Topic : Basic Evolutionary Processes

4.1 Basic Evolutionary Processes

- The basic Darwinian's evolutionary process constitutes following two things :
  1. One or more population of individual that compete for limited resource
  2. The notion of dynamically changing populations due to the birth and death of individuals.
  3. A concept of fitness which reflects the ability of an individual to cope with and survive and reproduce in its environment.
  4. A concept of variational inheritance : offspring closely resembles their parents, but are not identical.
- A simple evolutionary models focus on evolution over time of a single fixed – size population of individuals in a fixed environment with fixed mechanism for reproduction and inheritance.

Syllabus Topic : EV : A Simple Evolutionary System

4.2 EV : A Simple Evolutionary System

- Let us now understand how a simple evolutionary system can be simulated and observed its behavior over a time.
- Very first challenge here is, how to represent the individuals that make up an evolving population.
- One way to describe an individual as a fixed length vector of L features.
- These L features that are chosen should have potential to estimate an individual's fitness.

Scanned with CamScanner

For example, individual might be represented as L = 5 following features.

- <eye color, hair color, skin color, height, weight >
- The above vector specifies genetic makeup of an individual. i.e., its genotype specified as a chromosome with five genes (eye color, hair color, skin color, height, weight) whose values result in an individual with a particular trait of set.
- Alternatively, we could consider such vectors as description of the observable physical traits of individuals. i.e. their phenotype.
- In addition to specifying a geno/phenospace, we need to define the "laws of motion" for an evolutionary system.
- Consider the following pseudo code :

Simple EV

Generate an initial population of N individuals

Do forever

- Select an individual from a current population to be a parent.
- Use the selected parent to produce new offspring that is similar to but not the exact copy of the parent.
- Select an individual of the population to die.

End Do

- The above model is highly simplified model of the biological evolutionary systems.
- Many things are yet not considered. For example, we have not considered distinction between genotype and phenotype. We have ignored the distinction between male and female and have only a sexual reproduction. Asexual production is ignored.

More Detailed EV is as follows :

EV

- Randomly generate the initial population of N individuals (Using a uniform probability distribution over the entire geno/phenospace)
- Compute the fitness of each individual in an initial population Do Forever

Choose a parent as follows :

- Select a parent randomly using a uniform probability distribution over the current population.

Scanned with CamScanner

Use the selected parent to produce a single offspring by :

- Making an identical copy of the parent and then probabilistically mutating it to produce the offspring.

Compute the fitness of the offspring :

Select a member of the population to die by :

- Randomly selecting a candidate for deletion from the current population using a uniform probability distribution and keeping either the candidate or the offspring depending on which one has higher fitness.

End Do

- In the above algorithm, we compute the fitness of each individual.
- The fitness of the individual is computed using so called objective fitness.
- Also, offspring is produced by mutating a parent.
- We assume that each gene of an individual is equally likely to be muted. This means that on an average only one gene in every individual is muted.
- If there are L genes, each gene has an independent probability of 1/L of being selected to undergo a mutation.

Syllabus Topic : Evolutionary Systems as Problem Solvers

4.3 Evolutionary Systems as Problem Solvers

- We can modify the EV in many ways so that it would become more realistic model or natural evolutionary system.
- We see the evolution as a computational tool. So, what computation an EV is performing?
- Scientists design the systems with clear goals in mind, what functions to perform and what objectives to be met.
- Computer scientists design and implement algorithms for sorting, searching, optimizing.
- Although EV system seems to be simple, it has potential to solve problems that require searching complex search space, solve hard optimization problem and are capable of adapting to changing environment.

Consider following simple change to EV.

Scanned with CamScanner

EV

- Generate an initial population of N individuals
- Do until a stopping criterion is met:
- Select an individual from a current population to be a parent.
- Use the selected parent to produce new offspring that is similar to but not the exact copy of the parent.
- Select an individual of the population to die.
- End Do
- Return the individual with the highest global objective fitness.
- Note that the above algorithm adds a stopping criterion and returns an answer.

Syllabus Topic : Historical Perspective

4.4 A Historical Perspective

4.4.1 Early Algorithmic Views

- In 1930s, the Swell Wright proposed that an evolutionary system can be used for exploring a multi-peaked fitness space and that it has a capability of dynamically forming clusters around the peaks of high fitness.
- This perspective gave rise to the concept of an evolutionary system as an optimization process.
- Today evolutionary computation is dominantly used for solving optimization problems.
- An evolutionary system may be considered as a complex and adaptive system that changes its makeup and its responses over time as it interacts with a dynamically changing landscape.

4.4.2 The Catalytic 1960s

- The basic idea of viewing evolution as a computational process was conceived during 1960s.
- The ideas really took root and began to grow in the 1960s because of the increasing availability of inexpensive digital computers that could be used for simulation and modelling by the scientist.
- Several groups were convinced by the idea that even simple evolutionary models could expressed in computational form that could be used for solving complex computer based problems.

- At the technical university of Berlin, RechenBerg and schwefel began formulating ideas about how evolutionary process could be used to solve difficult real-valued parameter optimization problem.
- From these early ideas, the family of algorithms called "evolution strategies" emerged.
- In 1966, the use of evolutionary system viewed as a tool for achieving a goals of artificial Intelligence. The intelligent agents were represented as a finite state machines. And an evolutionary frame work called "Evolutionary programming" was developed.
- Holland A (at the university of Michgan), found that the evolutionary process could be considered as a key element in designing and implementing robust adaptive system that were capable of dealing with uncertain and changing environment.
- This idea led to an initial family of "reproductive plans" which later formed the basis for "Simple Genetic Algorithms".

4.4.3 The Explorative 1970s

- The analysis of EAs in the 1960failed left two issues unresolved.
  1) Characterizing the behavior of implementable systems
  2) Understanding better, how they might be used for solving problems.
- To implement a simple EA, many design decisions need to take such as population management issues, mechanism for selecting parent, producing offspring etc.
- As a result, much of the research in 1970s were concentrated in gaining additional insight into these issues.
- Most of these activities occur in the three main groups: Evolutionary Programming, Evolution strategies, and genetic algorithms.

Evolutionary Programming

- EP paradigm basically involves a fixed sized population of N parents, each of which produced a single offspring.
- Both the parents and children are combined into the population of size 2N to produce next generation offspring of size N. These 2N parents are rank ordered by their fitness and only N individuals are allowed to survive.
- The initial work had proposed both asexual reproduction and sexual reproduction.
- 'Asexual' reproduction involves single parent with a mutation operator.
- 'Sexual' reproduction involves two parents combining to form offspring via recombination operator such as crossover.

- Empirical studies focused on a variety of issues, like including appropriate initialization strategies, deciding the probability of mutation, an appropriate recombination operator etc.

Evolution Strategies (ES)

- The evolutionary strategies focus on real-valued function optimization. Hence individual in ES, were represented as vectors of real numbers.
- There were various variations of ES models.
- In $(1 + \lambda)$ - ES model 1 parent produced λ offspring. The fittest of the $1+\lambda$ individuals was selected to be the single parent for the next generation of offspring.
- This asexual reproduction took the form of mutating one or more parent's gene value (real number) via a normally distributed perturbation with zero mean and a standard deviation σ.
- Studies showed that performance of such a system were highly sensitive to the choice of mean and σ.
- This resulted in something called as **adaptive mutation operator**.
- The representation of individuals was extended from a vector of size N to vector of size 2N. Here, additional N genes represented the variance σi to be used to mutate a Parameter i using normally distributed perturbation $G(0,\sigma i)$.
- Now the question was which mutation operator to be used to mutate the σi.
- The rule (1 : 5 rule) state that if more than one of five mutations using $G(0,\sigma i)$ is successful then mutation is in danger of being too exploitative. That means the step size is too small and σi needs to be increased.
- If the rate of success falls below 20% then the mutation is too explorative and σi is decreased.
- With the adaptive mutation operator and appropriate choice of λ, promising results were reported.

Genetic Algorithms

- The early GA focus was on to developing more application - independent algorithm.
- The GA uses the string like representation for individuals along with the string oriented genetic operators for producing offspring.
- The simplest string representation is the binary string.
- The mutation was achieved by flipping the bit with a fixed probability.

Scanned with CamScanner

- 1 point crossover operator was used, in which a randomly selected initial sequence of gene from one parent is combined with the remaining sequence of the genes from a second parent to produce offspring with features of both parents.
- Most of the early studies involve generational GA. I generation GA a fixed sized population of N parent produce a new population of N offspring which unconditionally replaces the parent population regardless of the fitness.
- However, parents are stochastically selected for producing offspring in proportion to their fitness.
- The fitter parent contributes significantly more genetic material to the next generation
- GAs were studied from both analytic and empirical point of view. The fixed length string results in schema theorem by Holland, concerning the behavior of a simple GA.

The Unifying 1990s

- Till 1990s, remarkable research and development of EA, EP and GA was done independently without interaction among the various groups.
- In 1990s many EA conference provided platform for various scientist to present their particular viewpoints, challenging other approaches etc.

Syllabus Topic : Canonical Evolutionary Algorithms

4.5 Canonical Evolutionary Algorithms

- Evolutionary algorithms are a heuristic-based approach to solve problems that cannot be easily solved in polynomial time, such as classically NP-Hard problems, and anything else that would take far too long to exhaustively process.

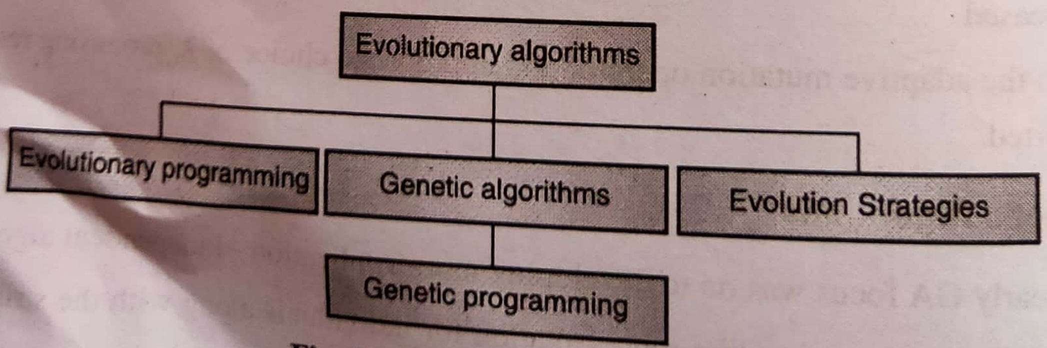


Fig. 4.5.1 : Classification of EA

Syllabus Topic : Evolutionary Programming

4.5.1 Evolutionary Programming

- EP was originated from the research of L.J. Fogel in 1962.
- EP is derived from the simulation of adaptive behavior in evolution.
- It differs substantially from GA in that evolutionary programming (EP) emphasizes the development of behavioral models and not genetic models. The main objective was to evolve intelligent behaviors.
- EP considers phenotypic evolution
- EP iteratively applies two evolutionary operators :
  1. Variation through application of mutation operators
  2. Selection
- In EP, **M** parents generates **M** offspring.
- The chromosomes are represented as finite state machines (FSMs).
- The objective is to optimize these FSMs in order to provide a meaningful representation of behaviour based on the interpolation of the symbol.
- Note that, EP does not use **crossover** operators. Rather, the main process here is the mutation operation.
- Here mutation is used to randomly create new offspring from the parent population. There exist five possible mutation operators :
  1. Modify an output symbol.
  2. Modify a state transition.
  3. Add a state.
  4. Delete a state.
  5. Change the initial state.
- The basic EP method involves 3 steps
  (1) Choose an initial POPULATION of trial solutions at random.
  (2) Each solution is replicated into a new POPULATION.
  Each of these OFFSPRING solutions are mutated according to a distribution of MUTATION types.
  (3) Each OFFSPRING solution is evaluated by computing it's FITNESS.

A stochastic tournament is held to determine N solutions to be retained for the POPULATION of solutions,

```
PSEUDO CODE (Algorithm EP is)

    // start with an initial time
    t := 0;

  // Initialize strategy parameters
InitistrategyS (t)

    // initialize a random population of individuals
    initpopulation P (t);

    // evaluate fitness of all initial individuals of population
    evaluate P (t);

    // test for termination criterion (time, fitness, etc.)
    while not done do

      // perturb the whole population stochastically
      P'(t) := mutate P (t);

      // evaluate it's new fitness
      evaluate P' (t);

      // stochastically select the survivors from actual fitness
      P(t+1) := survive P(t),P'(t);

      // increase the time counter
    t := t + 1;
    od
end EP.
```

The main components of an EP

- **Initialization**

  Initial population is generated by randomly selecting individual solutions

- **Evaluation**

  Fitness function measures the "behavioral error" of an individual with respect to the environment of that individual provides an absolute fitness measure of how well the problem is solved.

  Survival in EP is usually based on a relative fitness measure.

Scanned with CamScanner

- A score is computed to quantify how well an individual compares with a randomly selected group of competing individuals.
- Individuals that survive to the next generation are selected based on this relative fitness
- The search process in EP is therefore driven by a relative fitness measure, and not an absolute fitness measure.

Mutation

Mutation is the only means of variation in EP. (Crossover is not used at all)

Selection

- Main purpose to select new population.
- A competitive process (Usually tournament selection) where parents and offspring compete to survive.

Behaviors of individuals are influenced by strategy parameters.

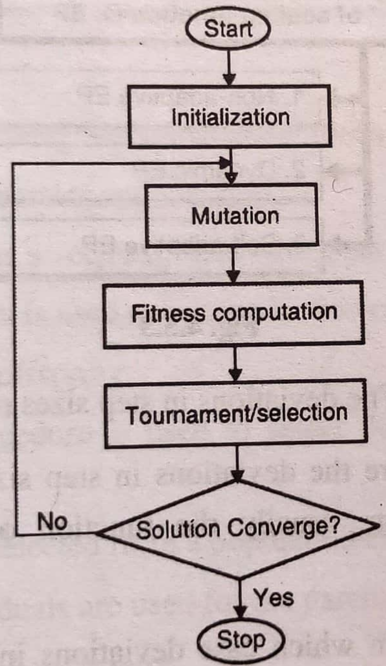


Fig. 4.5.2 : Flowchart : Evolutionary Programming

Mutation

Two important parameters, mutation operator and mutation step size need to be discussed.

Mutation operator

As discussed above, mutation is the only way of introducing variations in EP.

In general, the mutation is defined as ,

$$X'_{ij}(t) = X_{ij}(t) + \Delta X_{ij}(t)$$

$X'_i(t)$ is the offspring and $\Delta X_i(t)$ is the mutational step size.

Mutational step size

- Noise is sampled from some probability distribution.
- Deviation of noise is determined by a strategy parameter, σ_{ij}
- Generally, the step size is calculated as,

$$\Delta X_{ij}(t) = \text{Ø}(\sigma_{ij}(t)) \cdot \eta_{ij}(t)$$

Where, Ø is a scaling function that scales the contribution of noise.

Based on the characteristics of scaling function Ø , EP can have following three categories.

Categories of characteristics of scaling function Ø, EP

- 1. Non-adaptive EP
- 2. Dynamic EP
- 3. Self-adaptive EP

Fig. 4.5.3

1. **Non-adaptive EP :** The deviations in step sizes remain static.
2. **Dynamic EP :** Where the deviations in step sizes change overtime using some deterministic function, usually the function ϕ, a function of the fitness of individuals.
3. **Self-adaptive EP :** In which case deviations in step sizes change dynamically. The best values of σ_{ij} are learned in parallel with the decision variables, X_{ij}

Strategy Parameters

- Deviation σ_{ij} are called strategy parameters.
- Each individual has its own strategy parameter.

An individual is represented by a tuple.

$$X_i(t) = (X_i(t), \sigma_i(t))$$

Scanned with CamScanner

Selection operator

- The selection operator is used to create the new population.
- New population is selected from parents and their offspring.
- M represents the number of parents.
- Each individual in a parent set P(t) creates one child by mutation.
- So there will be exactly μ no. of children created. P(t) : μ parents, P'(t) : μ offspring
- Pair-wise competition (tournament) is held in round-robin format :

Each solution x from P(t) ∪ P′(t) is evaluated against q other randomly chosen solutions

- For each comparison, a "win" is assigned, if x is better than its opponent - The μ solutions with the greatest number of wins are retained to be parents of the next generation.

Syllabus Topic : Evolution Strategies

4.5.2 Evolution Strategies

- ES was developed by Rechenberg and schwefel in 1960s.
- It focuses on real valued parameter optimization.
- Individuals are represented as a vector of real-valued parameters.
- ***Gaussian mutation*** operators is used as a reproduction operator.
- M parent generates **k** >> M offspring.
- ***Deterministic*** selection procedure is used to select the possible parents for the next generation.
- Only best **λ** individuals are selected from a population P(s) that is of size **μ**.
- And only these best **λ** individuals are used for the parent population Q(s).
- This approach is called **G(μ, λ) strategy.**
- If the λ best individuals are treated as elite that enters the next generation then the strategy is called (λ + μ) strategy.

Canonical versions of the ES

There are two variations of ES :

(μ / ρ, λ) – ES or alternatively (μ, λ) – ES (Comma notation) ...(1)

And (μ/ρ + λ) – ES or alternatively (μ + λ) – ES (Plus notation) ...(2)

- Here μ is number of candidate solutions in the parent generation.

Scanned with CamScanner

- λ is the number of candidate solutions generated from the parent generation.
- $\rho \leq \mu$ is the mixing number i.e., the number of parents involved in the creation of an offspring.
- After creating λ offspring and calculating their fitness, the best μ of them are chosen deterministically, by either of the above mentioned selection method (1) or (2).
- In the first method $(\mu/\rho, \lambda)$, parents are deterministically selected only form offspring.(Comma notation).
- In the second method$(\mu/\rho + \lambda)$, the parents are deterministically selected from both the parents and offspring (plus notation).
- Both the (μ, λ) and the $(\mu + \lambda)$ selection schemes are strictly deterministic and are based on rank rather than an absolute fitness value
- Selection is based on the ranking of the individuals' fitness F(y) taking the μ best individuals. In general, an

  ES individual a : = (y, s, F(y))

  Comprises the *objective parameter vector* $y \in Y$ to be optimized
- A set of strategy parameters **s.**
- The individual's observed fitness F(y) being equivalent to the objective function f(y), i.e., $F(y) \equiv f(y)$ in the simplest case.
- The conceptual algorithm of the $(\mu/\rho, + \lambda)$ - ES is given below:

  1. Initialize parent population $P_\mu = \{a_1,\ldots,a_\mu\}$.
  2. Generate λ number of offspring a forming the offspring population $P_\lambda = \{a_1,\ldots, a_\lambda\}$ where each offspring a is generated by :
     (a) Select (randomly) ρ parents from P_μ
     (b) Recombine the ρ selected parents a to form a recombinant individual r.
     (c) Mutate the strategy parameter set s of the recombinant r.
     (d) Mutate the objective parameter set y of the recombinant r using the mutated strategy parameter.

- Select new parent population (using deterministic truncation selection) from either

  1. The offspring population P_λ or (comma notation)
  2. The offspring P_λ and parent P_μ population (plus notation)

  Goto 2. until *termination criterion* fulfilled.

Scanned with CamScanner

(1 + 1) – ES

- The simplest evolution strategy operates on a population of size two: the current point (parent) and the result of its mutation (one offspring).
- Only if the mutant's (offspring's) fitness is at least as good as the parent one, it becomes the parent of the next generation. Otherwise the mutant is disregarded. This is a (1 + 1) – ES.

(1 + λ) – ES

- Here the λ mutants can be generated from one parent and they all compete with the parent, c.
- Depending on the search space and objective function, the recombination and/or the mutation of the strategy parameters may or may not occur in specific algorithm.
- For example, a (μ + λ) – ES, does not use recombination. It draws its new μ parents for the next generation from both the old μ parents and the λ offspring (generated from these parents) by taking the best μ individuals.
- Evolution Strategies of type (μ + 1) are also referred to as steady state ESs, i.e., strategies without a generation gap: They produce only one offspring (λ = 1) per generation. After evaluating its fitness F(y), the worst individual is removed from the population.

Syllabus Topic : A Unified View of Simple EAs - A Common Framework, Population Size

4.6 A Unified View of Simple EAs - A Common Framework, Population Size

A simple EA consists of the following basic elements:

1. Parent population size m,
2. Survival population size n
3. Parent selection methods
4. Survival selection methods

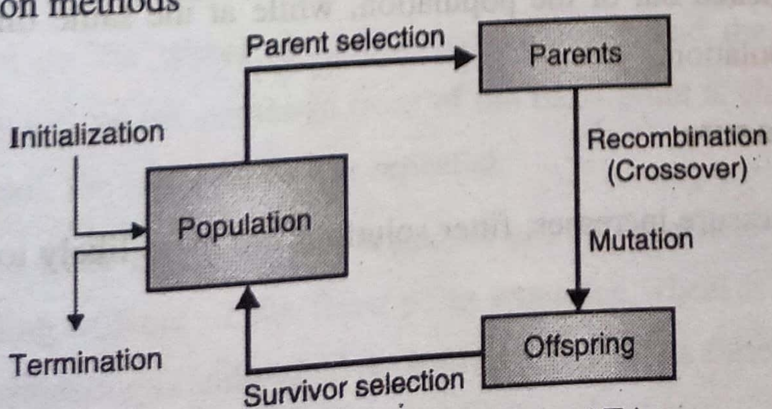


Fig 4.6.1 : A general schema of EA

1. Population Management

It involves managing parent population and children (offspring) population.

Two different population management models exist :

1.	Generational model
2.	Steady-state model

→ 1. Generational model

- Each individual survives for exactly one generation.
- The entire set of parents is replaced by the offspring.

→ 2. Steady-state model

- One offspring is generated per generation.
- One member of population replaced.

☞ Generation Gap

The proportion of the population replaced.

☞ Fitness based competition

Selection can occur in two places :

1. Parent selection (selects mating pairs)

Parent Selection is the process of selecting parents which mate and recombine to create off-springs for the next generation.

2. Survivor selection (replaces population)

The Survivor Selection Policy determines which individuals are to be kicked out and which are to be kept in the next generation. It is crucial as it should ensure that the fitter individuals are not kicked out of the population, while at the same time diversity should be maintained in the population.

☞ Selection pressure

As selection pressure increases, fitter solutions are more likely to survive, or be chosen as parents

Scanned with CamScanner

2. Parent Selection methods

Parent Selection methods

- a) Fitness Proportionate Selection
- b) Stochastic Universal Sampling (SUS)
- c) Tournament Selection
- d) Rank Selection
- e) Uniform Selection

Fig 4.6.2

➜ a) Fitness Proportionate Selection

- In this every individual can become a parent with a probability which is proportional to its fitness. Therefore, fitter individuals have a higher chance of mating and propagating their features to the next generation. Therefore, such a selection strategy applies a selection pressure to the more fit individuals in the population, evolving better individuals over time.
- Consider a circular wheel. The wheel is divided into **n pies**, where n is the number of individuals in the population. Each individual gets a portion of the circle which is proportional to its fitness value.
- Two implementations of fitness proportionate selection are possible :

☞ Roulette Wheel Selection

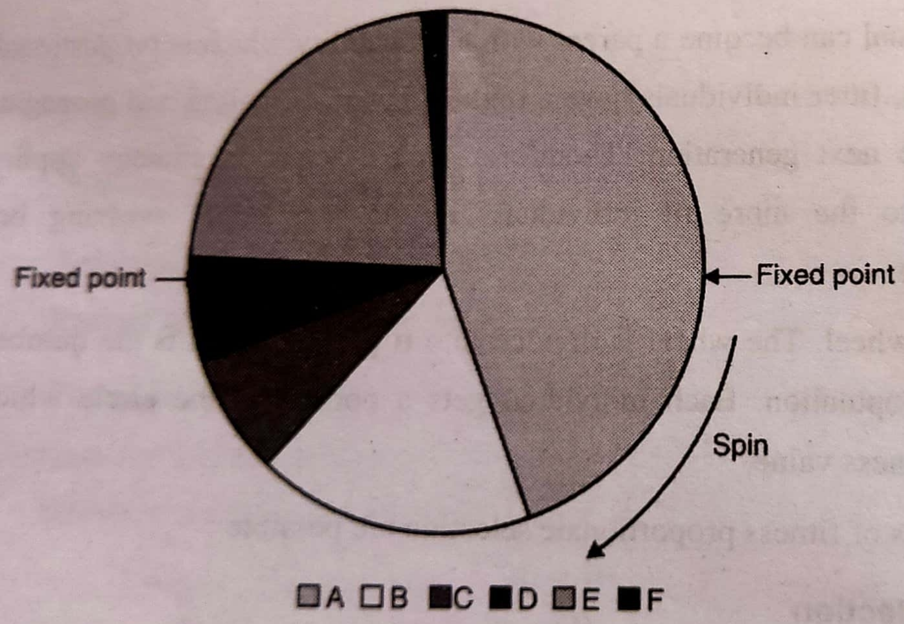
- In a roulette wheel selection, the circular wheel is divided as described before. A fixed point is chosen on the wheel circumference as shown and the wheel is rotated. The region of the wheel which comes in front of the fixed point is chosen as the parent. For the second parent, the same process is repeated.
- It is clear that a fitter individual has a greater pie on the wheel and therefore a greater chance of landing in front of the fixed point when the wheel is rotated. Therefore, the probability of choosing an individual depends directly on its fitness.

Scanned with CamScanner

- Implementation wise, we use the following steps :
  - Calculate S = the sum of a finesses.
  - Generate a random number between 0 and S.
  - Starting from the top of the population, keep adding the finesses to the partial sum P, till P < S.
  - The individual for which P exceeds S is the chosen individual.

b) Stochastic Universal Sampling (SUS)

- Stochastic Universal Sampling is quite similar to Roulette wheel selection, however instead of having just one fixed point, we have multiple fixed points as shown in the following image. Therefore, all the parents are chosen in just one spin of the wheel. Also, such a setup encourages the highly fit individuals to be chosen at least once.



Chromosome	Fitness value
A	8.2
B	3.2
C	1.4
D	1.2
E	4.2
F	0.3

Fig. 4.6.3

- It is to be noted that fitness proportionate selection methods don't work for cases where the fitness can take a negative value.

c) Tournament Selection

- In K-Way tournament selection, we select K individuals from the population at random and select the best out of these to become a parent. The same process is repeated for selecting the next parent. Tournament Selection is also extremely popular in literature as it can even work with negative fitness values.

Scanned with CamScanner

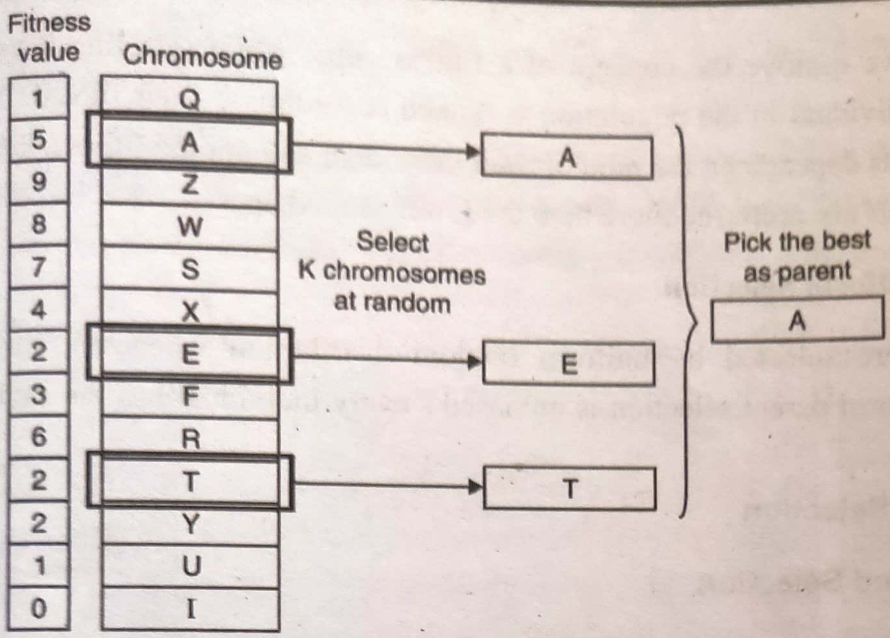
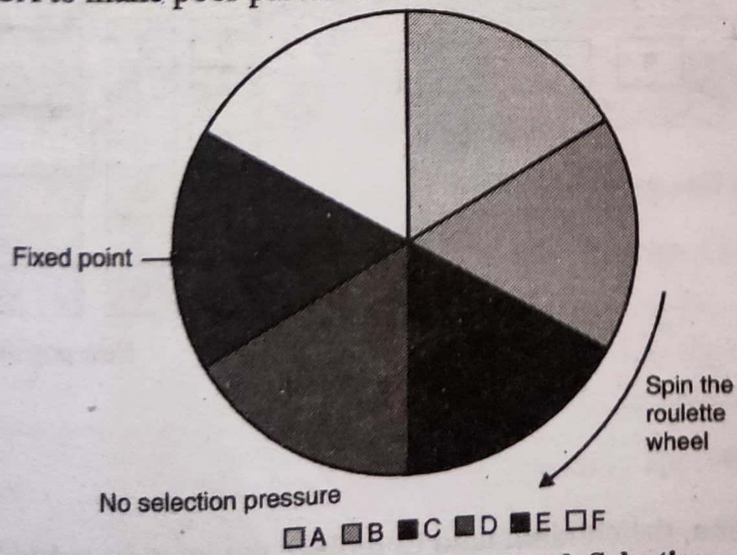


Fig. 4.6.4 : Example of tournament selection

d) Rank Selection

- Rank Selection also works with negative fitness values and is mostly used when the individuals in the population have very close fitness values (this happens usually at the end of the run).
- This leads to each individual having an almost equal share of the pie (like in case of fitness proportionate selection) as shown in the following image and hence each individual no matter how fit relative to each other has an approximately same probability of getting selected as a parent.
- This in turn leads to a loss in the selection pressure towards fitter individuals, making the GA to make poor parent selections in such situations.



Chromosome	Fitness value
A	8.1
B	8.0
C	8.05
D	7.95
E	8.02
F	7.99

Fig. 4.6.5 : Rank Selection

- In this, we remove the concept of a fitness value while selecting a parent. However, every individual in the population is ranked according to their fitness. The selection of the parents depends on the rank of each individual and not the fitness. The higher ranked individuals are preferred more than the lower ranked ones.

➜ **e) Uniform Selection**

Parents are selected by uniform random distribution whenever an operator needs one/some. Uniform parent selection is unbiased - every individual has the same probability to be selected.

3. Survival Selection

☞ **Age Based Selection**

- In Age-Based Selection, we don't have a notion of fitness. It is based on the premise that each individual is allowed in the population for a finite generation where it is allowed to reproduce, after that, it is kicked out of the population no matter how good its fitness is.
- For instance, in the following example, the age is the number of generations for which the individual has been in the population. The oldest members of the population i.e. P4 and P7 are kicked out of the population and the ages of the rest of the members are incremented by one.

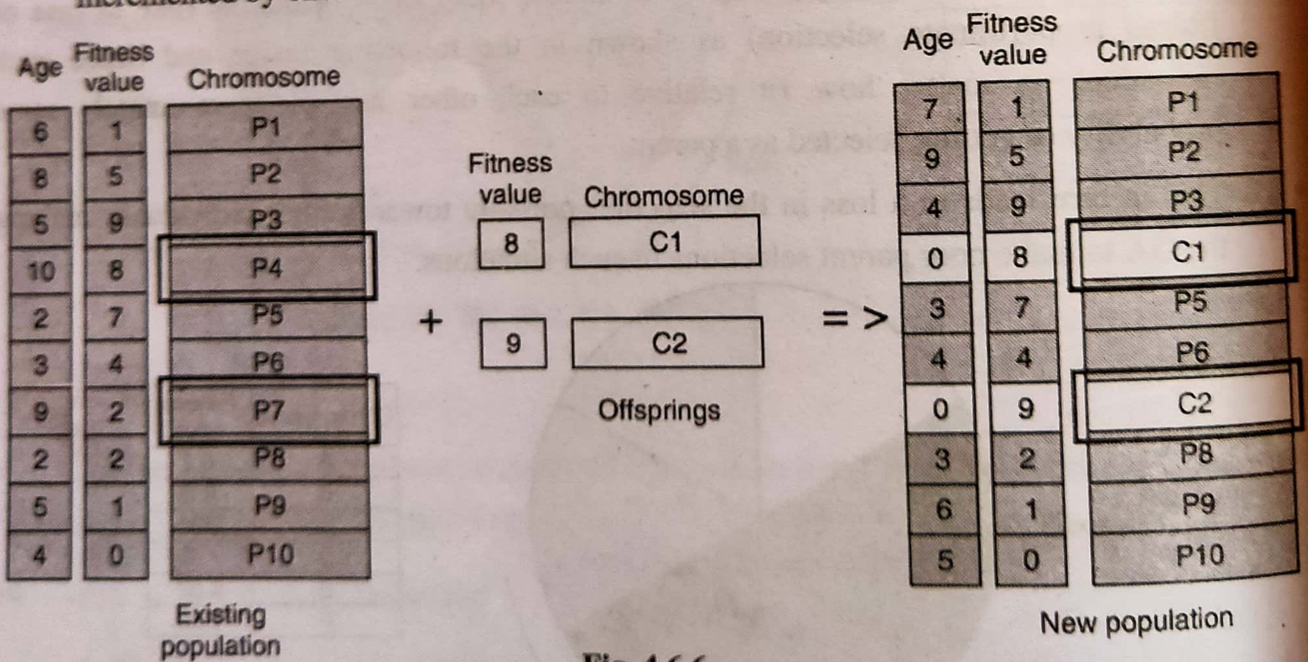


Fig. 4.6.6

☞ **Fitness Based Selection**

- In this fitness based selection, the children tend to replace the least fit individuals in the population. The selection of the least fit individuals may be done using a variation of any of the selection policies described before - tournament selection, fitness proportionate selection, etc.

- For example, in the following image, the children replace the least fit individuals P1 and P10 of the population. It is to be noted that since P1 and P9 have the same fitness value, the decision to remove which individual from the population is arbitrary.

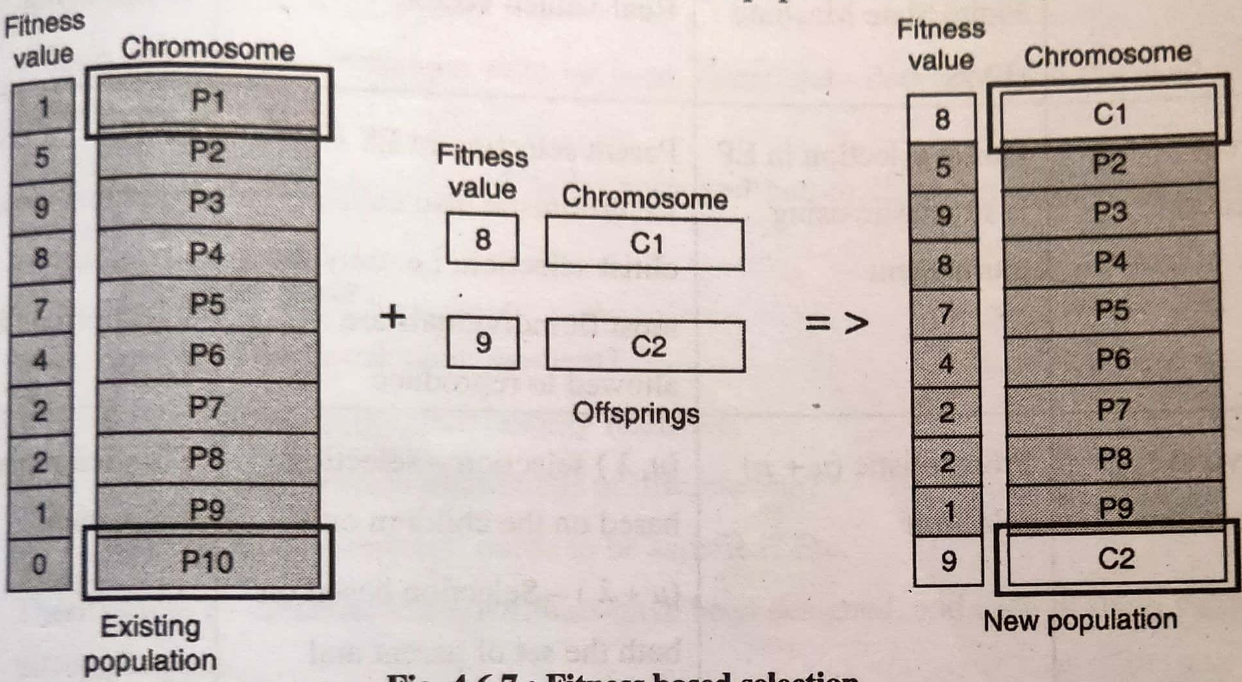


Fig. 4.6.7 : Fitness based selection

Elitism

In simple terms, it means the current fittest member of the population is always propagated to the next generation. Therefore, under no circumstance can the fittest member of the current population be replaced.

4.7 Difference between EP, ES and GA

Parameter	EP	ES	GA
Inventor	Fogel at California school in 1962.	Rechenberg and schwefel at German school in 1960s	Named by Holland at Michigan school in 1970s
Objective	To evolve intelligent behavioral model	Typically applied to numerical optimization.	To evolve genetic model

Parameter	EP	ES	GA
Representation	Individual solution is represented as a Finite State Machine (FMS).	The individuals are represented by Real-valued vector.	Individuals are represented as a binary string.
Parent selection	Parent selection in EP is stochastic using tournament	Parent selection in ES is Deterministic also called elitist selection. i.e. only the most fit individuals are allowed to reproduce.	Both deterministic and stochastic selection can be used.
Survival Selection	Probabilistic $(\mu + \mu)$ *selection*	(μ, λ) selection – selection based on the children only $(\mu + \lambda)$ – Selection based on both the set of parent and children	Children replace the parents
Mutation	Mutation in EP is Gaussian perturbation. Self-adaptation of mutation step size. The strategy parameters control the mutation of The object parameters.	Gaussian Mutation is used. Self-adaptation of mutation step size. The strategy parameters control the mutation of The object parameters.	GA mutation tend to be fixed sixe Mutation.
Recombination	EP does not use recombination to produce offspring. Rather it only uses mutation.	ES uses recombination such as cross over to produce offspring.	GA uses various recombination operators.

Scanned with CamScanner

Additional question

Q. 1 Is it advisable to apply genetic algorithm for all kinds of optimization problems? Justify.

Ans. :

- No, it is not advisable to apply genetic algorithm for all kinds of optimization problems.
- There is no single optimization algorithm that can solve all kinds of optimization problems.
- Which optimization technique is to be used, completely depends upon the kind of the problem being solved.
- There are several questions need to be answered before choosing a right optimization technique such as,
  - o Is problem linear?
  - o Is it possible to calculate gradient?
  - o Is your problem computationally intensive?
  - o Is objective function continuous or discrete one?
  - o Are there any constraint needs to be satisfied? Etc.
- There is lots of optimization algorithms have been designed, and each of them suited for different tasks.
- For example, the simplex algorithm and its variants work great for linear programs, problems that can be expressed in terms of linear combinations of a set of continuous decisions. This approach is good for small problems.
- Bigger linear programs often result from representational issues and a resulting explosion of decisions and constraints. Such problems can be still solved quite rapidly by simplex or interior point algorithms. Linear integer programming techniques like branch and bound or branch and cut often work better in these circumstances.
- Another option for problems with *discrete decisions that have nonlinear cost models* is dynamic programming, especially if there is a natural ordering to the decisions.
- Genetic algorithms are slow, so they're not good for problem classes like the ones we described above, where special purpose algorithms exist specifically to solve them.
- GAs are excellent for the problems that are hard to represent as a member of a known class of problems. For example, optimizing a statistic on the outputs of a simulation model over multiple random trials. There's no closed-form representation for this kind of problem.

Scanned with CamScanner

- GAs are most appropriate for **complex non-linear** models where finding a location of the global optimum is a difficult task.
- *Multi-objective optimization* is also a particular strength of genetic algorithms, because it takes advantage of the GA's population-based search.
- GAs, and especially MOEAs (Multi-objective evolutionary algorithms) are therefore great in an interactive decision-aiding context, where the user is free to tweak the problem and see what results emerge.
- The list of topics to which genetic algorithms have been applied is extensive.
- Circuit optimization : pick values of electronic components that meet timing constraints while minimizing leakage current.
- Production planning for manufacturing: choose a product mix that balances risk and profit while minimizing overproduction, using probability models for yield and demand over time.
- Monitoring system design: where to place sensors to minimize cost while maximizing coverage.

Review Questions

Q. 1 Explain how Evolutionary programming differs from Evolutionary strategies. *(Ans. : Refer Section 4.7)*

Q. 2 How Genetic algorithms differ from evolutionary programming and evolutionary strategies? *(Ans. : Refer Section 4.7)*

Q. 3 Differentiate between evolution programming and evolution strategies. Explain the main components of evolutionary programming. *(Ans. : Refer Sections 4.7 and 4.5.1)*

Q. 4 Explain different parent selection methods used in EV systems. *(Ans. : Refer Section 4.6)*

Q. 5 What are the different survivor selection methods? Explain each in details. *(Ans. : Refer Section 4.6)*

Q. 6 Comment on the parent selection in ES and EP. *(Ans. : Refer Section 4.5.1 and 4.5.2)*

❑❑❑

Chapter Ends...

Scanned with CamScanner