Introduction to Genetic Algorithms

1.1 Introduction

Evolutionary computing was introduced in the 1960s by I. Rechenberg, then was developed by other researches. The theory of evolution was introduced by Charles Darwin (1859) to explain his observations of plants and animals in the natural ecosystem. He observed that as every new generation was associated with some changes, the less-fit individuals tended to lose the battle for survival in the competition for food.

Genetic Algorithms (GAs) mimics the theory of evolution and natural selection. Genetic Algorithms was invented and developed by John Holland in 1975. Holland proposed GA as a heuristic method based on "Survival of the fittest". GA was discovered as a useful tool for search and optimization problems.

It is also important to mention in this introduction GA limits. Like most stochastic methods, *GAs are not guaranteed to find the global optimum solution to a problem*, they are satisfied with finding ***"acceptably good"*** solutions to the problem.

GAs are extensively used in many applications across a large and growing number of disciplines. Although a GA is able to find very good solutions for a variety of applications, the amount of time consumed for large computations and iterations is enormous. Hence, software implementation of GAs for increasingly complex applications can cause unacceptable delays. Also, GAs lend themselves easily to pipelining and parallelization. These factors make GAs good candidates for hardware implementation.

From the optimization point of view, the main advantage of evolutionary computation techniques is that they do not have much mathematical requirements about the optimization problems. All they need is an

evaluation of the objective function. As a result, they are applied to non-linear problems, defined on discrete, continuous or mixed search spaces.

1.2 Search Space

Most often one is looking for the best solution in a specific set of solutions. The space of all feasible solutions (the set of solutions among which the desired solution resides) is called search space (also state space). Each and every point in the search space represents one possible solution. Therefore each possible solution can be “marked” by its fitness value, depending on the problem definition. With Genetic Algorithm one looks for the best solution among a number of possible solutions represented by one point in the search space i.e.; GAs are used to search the search space for the best solution e.g., minimum. The difficulties in this ease are the local minima and the starting point of the search (see Fig. 1.1).

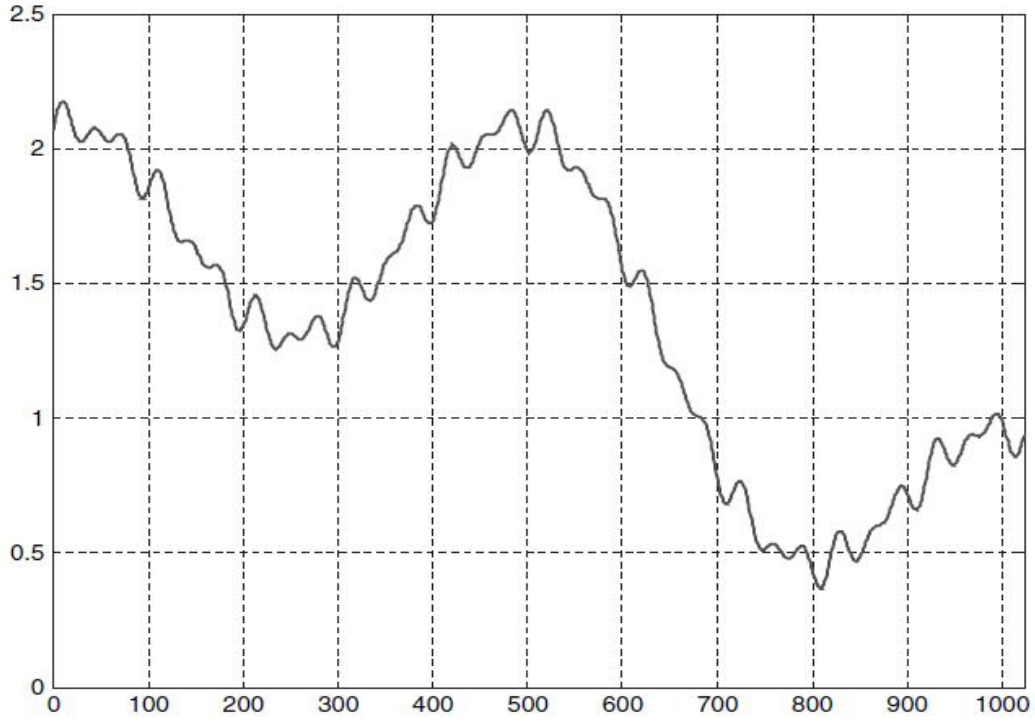

Fig. 1.1 An Example of search space

1.3 What is Genetic Algorithm

The science that deals with the mechanisms responsible for similarities and differences in a species is called Genetics. The word “genetics” is derived from the Greek word “genesis” meaning “to grow” or “to become”.

Genetic Algorithms are a class of evolutionary algorithms that use biologically-derived techniques such as ***inheritance, natural selection, crossover (or recombination) and mutation***. They work on the principle of **“survival of the fittest”,** where the less fit members of a particular generation are replaced by new members formed by combining parts of highly fit members. Traditionally, solutions to GAs are represented in binary as strings of 0s and 1s, but different encoding schemes are also possible.

GAs differ from other conventional optimization processes in:

1. GAs work with a coding of the parameter set, not the parameters themselves.
2. GAs search from a population of points, not a single point.
3. GAs use objective function information, not derivatives or other auxiliary knowledge.
4. GAs use probabilistic transition rules, not deterministic rules.

1.4 A Simple Genetic Algorithm

A genetic algorithm is a problem solving method that uses genetics as its model of problem solving. It’s a search technique to find approximate solutions to optimization and search problems.

GA handles ***a population*** of possible solutions. Each solution is represented through ***a chromosome***, which is just an abstract representation.

- ***Coding all the possible solutions into a chromosome is the first part in Genetic Algorithm.***

- A set of *reproduction operators* has to be determined, too. ***Reproduction operators are applied directly on the chromosomes,*** and are used to perform ***mutations*** and ***recombinations*** over solutions of the problem.
- Selection is supposed to be able to compare each individual in the population. Selection is done by using a fitness function. Each chromosome has an associated value corresponding to the fitness of the solution it represents. The fitness should correspond to an evaluation of how good the candidate solution is.
- ***The optimal solution is the one, which maximizes the fitness function.*** Genetic Algorithms deal with the problems that maximize the fitness function. But, if the problem consists in minimizing a cost function, the adaptation is quite easy. Either the cost function can be transformed into a fitness function, for example by inverting it; or the selection can be adapted in such way that they consider individuals with low evaluation functions as better.
- Once the ***reproduction*** and the ***fitness*** function have been properly defined, a Genetic Algorithm starts by ***generating an initial population of chromosomes***. This first population must offer a wide diversity of genetic materials. ***The gene pool should be as large as possible*** so that any solution of the search space can be engendered.

Generally, ***the initial population is generated randomly***. Then, the genetic algorithm loops over an iteration process to make the population evolve. Each iteration consists of the following steps:

• SELECTION: The first step consists in selecting individuals for reproduction. This selection is done randomly with a probability depending on the relative fitness of the individuals so that best ones are often chosen for reproduction than poor ones.

- REPRODUCTION: In the second step, offspring are bred by the selected individuals. For generating new chromosomes, the algorithm can use both recombination and mutation.
- EVALUATION: Then the fitness of the new chromosomes is evaluated.
- REPLACEMENT: During the last step, individuals from the old population are killed and replaced by the new ones.

The algorithmis stopped when the population converges toward the optimal solution.

The basic genetic algorithm is as follows:

- [start] Genetic random population of n chromosomes (suitable solutions for the problem)
- [Fitness] Evaluate the fitness f(x) of each chromosome x in the population
- [New population] Create a new population by repeating following steps until the New population is complete

- [selection] select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to get selected).
- [crossover] With a crossover probability, cross over the parents to form new offspring (children). If no crossover was performed, offspring is the exact copy of parents.
- [Mutation] With a mutation probability, mutate new offspring at each locus (position in chromosome)
- [Accepting] Place new offspring in the new population.
- [Replace] Use new generated population for a further sum of the algorithm.

- [Test] If the end condition is satisfied, stop, and return the best solution in current population.

• [Loop] Go to step2 for fitness evaluation.

The flowchart showing the process of GA is as shown in Fig. 1.2, while Fig. 1.3 shows the various processes of a GA system.

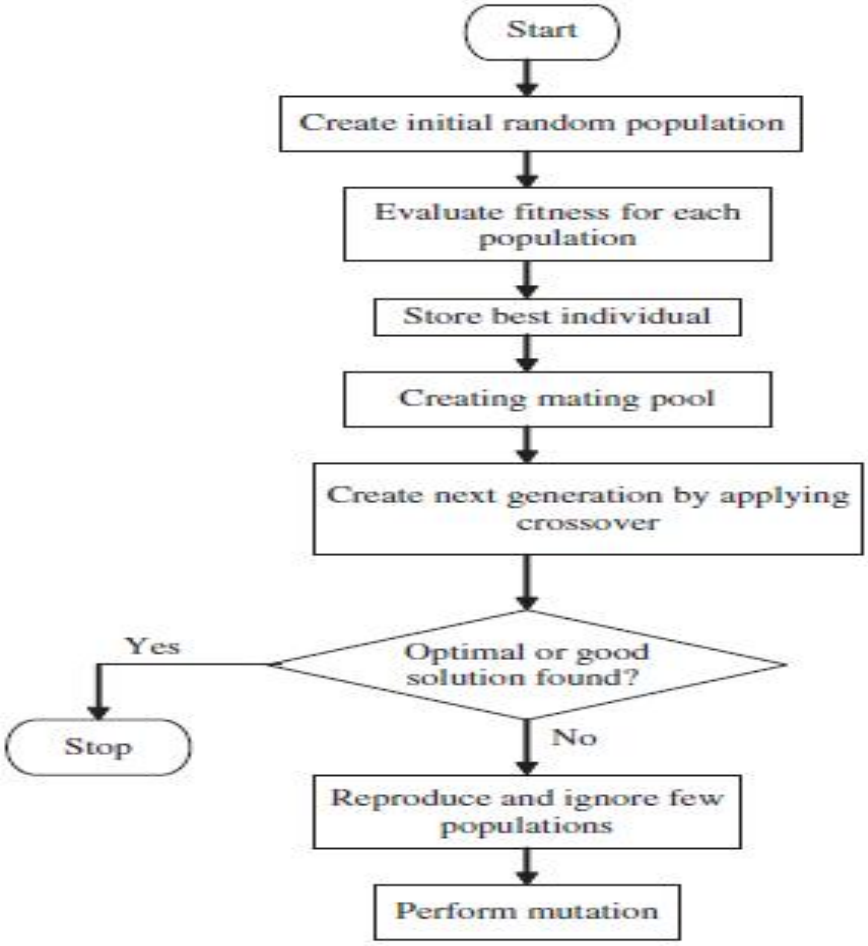

Fig. 1.2 Genetic Algorithm Flow Chart

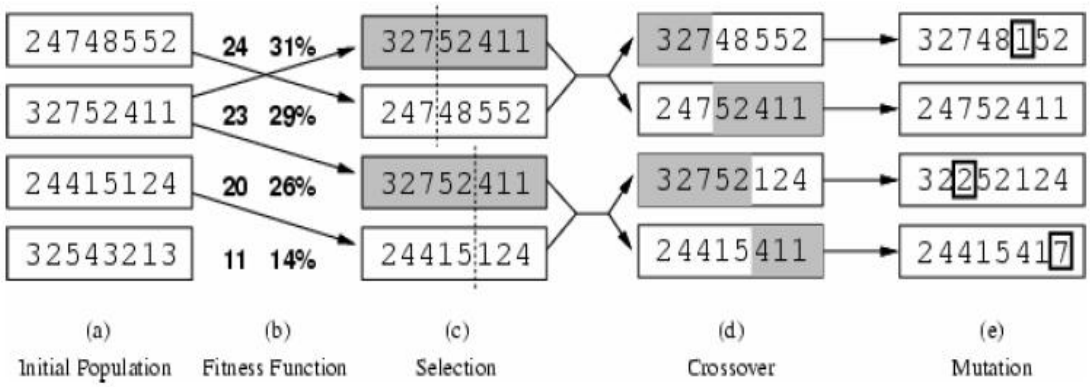

Fig. 1.3 The various processes of a GA system

In short, the basic four steps used in simple Genetic Algorithm to solve a problem are,

1. The representation of the problem
2. The fitness calculation
3. Various variables and parameters involved in controlling the algorithm
4. The representation of result and the way of terminating the algorithm

WHY GA WORK?

1. In each generation we check several solutions at once. Thus GA
2. is a kind of a parallel search.
3. Fitness and selection filter out bad solutions from good ones.
4. Offspring inherit properties of mostly good solutions.

1.5 Advantages and Limitations of Genetic Algorithm

The advantages of genetic algorithm includes,

1. Parallelism
2. Solution space is wider
3. Easy to discover global optimum
4. The problem has multi objective function
5. Easily modified for different problems.
6. Handles noisy functions well.
7. Handles large, poorly understood search spaces easily
8. Good for multi-modal problems Returns a suite of solutions.
9. Very robust to difficulties in the evaluation of the objective function.
10. They are resistant to becoming trapped in local optima
11. They perform very well for large-scale optimization problems
12. Can be employed for a wide variety of optimization problems

The limitation of genetic algorithm includes,

1. The problem of identifying fitness function
2. Definition of representation for the problem
3. Premature convergence occurs

4. The problem of choosing the various parameters like the size of the population, mutation rate, cross over rate, the selection method and its strength.
5. Cannot easily incorporate problem specific information
6. Not good at identifying local optima
7. No effective terminator.
8. Have trouble finding the exact global optimum
9. Require large number of response (fitness) function evaluations

1.6 Applications of Genetic Algorithm

Genetic algorithms have been used in many applications such as:

1. Nonlinear dynamical systems–predicting, data analysis
2. Robot trajectory planning
3. Strategy planning
4. Finding shape of protein molecules
5. Functions for creating images
6. Control–gas pipeline, pole balancing, missile evasion, pursuit
7. Design–semiconductor layout, aircraft design, keyboard configuration, communication networks
8. Scheduling–manufacturing, facility scheduling, resource allocation
9. Machine Learning–Designing neural networks, both architecture and weights, improving classification algorithms, classifier systems
10. Signal Processing–filter design
11. Combinatorial Optimization–set covering, traveling salesman (TSP), Sequence scheduling, routing, bin packing, graph coloring and partitioning