Volume 5, Issue 4, 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Evolutionary Computation an Optimization Technique

Indresh Kumar Gupta*
ABV-Indian Institute of Information Technology and Management,
Gwalior, Madhya Pradesh, India

Abstract— Evolutionary Computation concerns with computational system, which use ideas and get inspiration from the natural evolution. The main principle which has taken from the natural evolution is survival of fittest, i.e. Fittest solution will survive in each generation. Evolutionary Computation (EC) techniques can be used in optimization, designing and other computational problems. In this paper we present the generic evolutionary algorithm, and also summarized how evolutionary algorithms differ from classical optimization algorithms.

Keywords— Evolutionary Computation (EC), Evolutionary Algorithms (EAs), Optimization, Fitness Function.

I. INTRODUCTION

Evolutionary Computation (EC) refers to computer-based problem solving systems that use computational models of evolutionary processes, such as natural selection, survival of the fittest and reproduction, as the fundamental components of such computational systems. It was Charles Darwin's theory of *natural selection* that became the foundation of biological evolution (Alfred Wallace developed a similar theory at the same time, but independently of Darwin). The Darwinian theory of evolution [1] can be summarized as: In a world with limited resources and stable populations, each individual competes with others for survival. Those individuals with the "best" characteristics (traits) are more likely to survive and to reproduce, and those characteristics will be passed on to their offspring. These desirable characteristics are inherited by the following generations, and (over time) become dominant among the population [2][3].

A second part of Darwin's theory states that, during production of a child organism random events cause random changes to the child organism's characteristics. If these new characteristics are a benefit to the organism, then the chances of survival for that organism are increased. . Jean-Baptiste Lamarck's theory of evolution was that of *heredity*, i.e. the inheritance of acquired traits. The main idea is that individuals adapt during their lifetimes, and transmit their traits to their offspring. The offspring then continue to adapt. According to Lamarckism, the method of adaptation rests on the concept of use and disuse: over time, individuals lose characteristics they do not require, and develop those which are useful by "exercising" them [4].

II. GENERIC EVOLUTIONARY ALGORITHM

Evolution via natural selection of a randomly chosen population of individuals can be thought of as a search through the space of possible chromosome values. In that sense, an evolutionary algorithm (EA) is a stochastic search for an optimal solution to a given problem [5][6][7] . The evolutionary search process is influenced by the following main Components of an EA:

- A representation of solutions to the problem as a chromosome;
- A function to evaluate the fitness, or survival strength of individuals;
- Initialization of the populations;
- Selection operators ; and
- Reproduction operators

Generic Evolutionary Algorithm
Initialize the generation counter gen=0;
Create randomly n-dimensional population, T(0) which consist the μ individuals;
While stopping condition(s) , is not meet ;
Evaluate the fitness, $f(X_i(gen)), of\ each\ indiviual, X_i(gen)$;
Perform reproduction to create offspring;
Select the new population , T(gen+1);
Increment the generation counter, gen =gen+1:
End

III. REPRESENTATION – THE CHROMOSOME

In nature, organisms have certain characteristics that influence their ability to survive and to reproduce. These characteristics are represented by long strings of information contained in the chromosomes of the organism. Chromosomes are structures of compact intertwined molecules of DNA, found in the nucleus of organic cells .each chromosome contain a large number of genes, where a gene is the unit of heredity. Genes determine many aspects of anatomy and physiology through control of protein production. Each individual has a unique sequence of genes. An alternative form of a gene is referred to as an *allele*. In the context of EC, each individual represents a candidate solution to an optimization problem. The characteristics of an individual are represented by a chromosome, also referred to as a genome. These characteristics refer to the variables of the optimization problem, for which an optimal assignment is sought. Each variable that needs to be optimized is referred to as a *gene*, the smallest unit of information. Characteristics of an individual can be divided into two classes of evolutionary information: genotypes and phenotypes. A genotype describes the genetic composition of an individual, as inherited from its parents; it represents which allele the individual possesses. A phenotype is the expressed behavioural traits of an individual in a specific environment; it defines what an individual looks like. An important step in the design of an EA is to find an appropriate representation of candidate solutions (i.e. chromosomes). The efficiency and complexity of the search algorithm greatly depends on the representation scheme [8][9][10][11]. The classical representation scheme for GAs is binary vectors of fixed length. In the case of an n-dimensional search space, each individual consists of *n*- variables with each variable encoded as a bit string. If variables have binary values, the length of each chromosome is *n*- bits. However , GA are not restricted to use only integer values; in fact, it can be assigned to use any other integer or non-integer values just by changing the string length and lower and upper bounds :

$$x_j = x_j^{min} + \frac{x_j^{max} - x_j^{min}}{2^{l_j} - 1} DV(s_j)$$

Where l_j is the string length used to code the j^{th} varible $DV(s_j)$ is the decoded value of string s_j, x_j^{max} and x_j^{min} corresponds to maximum and minimum value of variable x_j.

IV. INITIAL POPULATION

The first step in applying an Evolutionary Algorithm to solve an optimization problem is to generate an initial population. The standard way of generating an initial population is to assign a random value from the allowed domain to each of the genes of each chromosome. The goal of random selection is to ensure that the initial population is a uniform representation of the entire search space. If regions of the search space are not covered by the initial population, chances are that those parts will be neglected by the search process [7][8][12][13] .

The size of the initial population has consequences in terms of computational complexity and exploration abilities. Large numbers of individuals increase diversity, thereby improving the exploration abilities of the population. However , the more the individuals, the higher the computational complexity per generation. While the execution time per generation increases, it may be the case that fewer generations are needed to locate an acceptable solution. A small population, on the other hand will represent a small part of the search space. While the time complexity per generation is low, the EA may need more generations to converge than for a large population. In the case of a small population, the EA can be forced to explore more of the search space by increasing the rate of mutation[14][15].

V. FITNESS FUNCTION

In the Darwinian model of evolution, individuals with the best characteristics have the best chance to survive and to reproduce. In order to determine the ability of an individual of an EA to survive, a mathematical function is used to quantify how good the solution represented by a chromosome is [16][17]. Fitness function maps the representation of chromosomes to scalar values. Fitness function Usually provides an absolute measures of fitness that is, the solution represented by a chromosome is directly evaluated using objective function , for example if objective function to be maximize then fitness function may be same as the objective function. Similarly if objective function to be minimize then fitness function may be the 1/objective function .

VI. SELECTION

Selection is one of the main operators in EAs, and relates directly to the Darwinian concept of survival of the fittest. The main objective of selection operators is to emphasize better solutions. This is achieved in two of the main steps of an EA:

Selection of New Population: A new population of candidate solutions is selected at the end of each generation to serve as the population of the next generation. The new population can be selected from only the offspring, or from both the parents and the offspring. The selection operator should ensure that good individuals do survive to next generations.

Selection of Parents that will take part Recombination Process: Offspring are created through the application of crossover and/or mutation operators. In terms of crossover, "superior" individuals should have more opportunities to reproduce to ensure that offspring contain genetic material of the best individuals. In the case of mutation, selection mechanisms should focus on "weak" individuals. The hope is that mutation of weak individuals will result in introducing better traits to weak individuals, thereby increasing their chances of survival.

- Random Selection: It is simplest selection approach, in which selection probability of each individual is $1/\mu$ (μ is the population size). No fitness information is used, which implies that best and worst individuals have exactly the same chance of surviving to the next generation.
- Proportionate Selection: If all population member fitness is F_{avg} , then i^{th} individual with fitness F_i gets an expected F_i/F_{avg} number of copies [18].
- Roulette Wheel Selection: In this approach , a wheel is divided into μ parts , and each individual in the population gets a parts of roulette wheel whose portion is proportional to the fitness value that means larger the fitness value is , larger the portion is . Then wheel is spun μ times and each time when wheel is stop individual corresponding to that portion is selected [19]. In proportional roulette wheel selection, selection probability of individual is directly proportional to fitness value. If the fitness value for i^{th} individual is F_i then its selection probability is $P_i = F_i/\sum_{j=1}^{\mu} F_j$.

  Rank Based Selection: When fitness value of an individual is very high then roulette wheel suffers. If an individual is 75% fittest, then it's got 75% portion of the roulette wheel and selected most of the times. To overcome this rank based selection is performed [20] in which individual is sorted based on their fitness value. The best fitness, worst fitness have value μ and 1 respectively. Based on ranking, selection probability of i^{th} individual is $P_i = \text{rank}(i)/\mu(\mu - 1)$.
- Tournament Selection: In this approach a tournament is performed among $t < \mu$ (μ is the population size) individual , the individual who has the highest fitness win the tournament and inserted into the matting pool. If we want size of mating pool be s, then s time tournament is performed. Tournament winner creates the mating pool [21].
- Selection of New Population: In each generation from μ parents λ offspring are produced using crossover and mutation, now individual to be selected for next generation. Two fundamental approach are $(\mu + \lambda)$ and (μ, λ) .In $(\mu + \lambda)$ algorithm produced λ offspring from μ parents, with $1 \leq \mu \leq \lambda < \infty$. From μ parents and λ offspring μ individual are selected to survive in next generation. This strategy implement the elitism to guarantee that the fittest individual survive in next generation. In (μ, λ) strategy μ best individual are selected from λ offspring with $1 \leq \mu < \lambda < \infty$. As elitism is not used, thus this strategy shows a lower selective pressure than for the plus strategies. Elitism refers to the process of ensuring that the best individuals of the current population survive to the next generation [22].

VII. REPRODUCTION OPERATORS

Reproduction is the process of producing offspring from selected parents by applying crossover and/or mutation operators. Crossover is the process of creating one or more new individuals through the combination of genetic material randomly selected from two or more parents. If selection focuses on the most-fit individuals, the selection pressure may cause premature convergence due to reduced diversity of the new population [23][24][25]. Mutation is the process of randomly changing the values of genes in a chromosome. The main objective of mutation is to introduce new genetic material into the population, thereby increasing genetic diversity. Mutation should be applied with care not to distort the good genetic material in highly fit individuals. For this reason, mutation is usually applied at a low probability. To promote exploration in the first generations, the mutation probability can be initialized to a large value, which is then reduced over time to allow for exploitation during the final generations.

VIII. STOPPING CONDITIONS

The evolutionary operators are iteratively applied in an EA until a stopping condition is satisfied. The simplest stopping condition is to limit the number of generations that the EA is allowed to execute, or alternatively, a limit is placed on the number of fitness function evaluations. This limit should not be too small, otherwise the EA will not have sufficient time to explore the search space [1][2][22].

In addition to a limit on execution time, a convergence criterion is usually used to detect if the population has converged. Convergence is loosely defined as the event when the population becomes stagnant. In other words, when there is no genotypic or phenotypic change in the population. The following convergence criteria can be used:

- Terminate when no improvement is observed over a number of consecutive generations. This can be detected by monitoring the fitness of the best individual. If there is no significant improvement over a given time window, the EA can be stopped. Alternatively, if the solution is not satisfactory, mechanisms can be applied to increase diversity in order to force further exploration. For example, the mutation probability and mutational step sizes can be increased.
- Terminate when there is no change in the population. If, over a number of consecutive generations, the average change in genotypic information is too small, the EA can be stopped.
- Terminate when an acceptable solution has been found. If X^* represents the optimum of the objective function, then if the best individual, X, is such that $f(X) \leq |f(X^* - \epsilon)|$, an acceptable solution is found; ϵ is the error threshold. If ϵ is too large, solutions may be bad. Too small values of ϵ may cause the EA never to terminate if a time limit is not imposed.
- Terminate when the objective function slope is approximately zero.

IX. EVOLUTIONARY COMPUTATION VERSUS CLASSICAL OPTIMIZATION

For liner, quadratic, strongly convex, unimodal and other specialized optimization problems the classical optimization algorithms have found more successful and efficient than evolutionary algorithm. While evolutionary algorithms have been shown to be more efficient for discontinuous, non-differentiable, multimodal and noisy problem. Evolutionary computation (EC) and classical optimization (CO) differ mainly in the search process and information about the search space used to guide the search process [26][27]:

Search Process: CO uses deterministic rules to move from one point in the search space to the next point. EC, on the other hand, uses probabilistic transition rules. Also, EC applies a parallel search of the search space, while CO uses a sequential search. An EA search starts from a diverse set of initial points, which allows parallel search of a large area of the search space. CO starts from one point, successively adjusting this point to move toward the optimum.

Search Surface information: CO uses derivative information, usually first order or second-order, of the search space to guide the path to the optimum. EC, on the other hand, uses no derivative information. The fitness values of individuals are used to guide the search.

X. CONCLUSION

Evolutionary computation uses the idea from the biological evolution to solve the computational problems. Algorithm of evolutionary computing follows the "survival of fittest in natural evolution "means at each iteration the fittest solution will survive. Any evolutionary algorithm have five components : Representation , fitness evaluation , population initialization , selection , reproduction . In this paper we have presented these components in details and working of generic evolutionary algorithm. Finally we discuss how evolutionary algorithms differs from the classical optimization algorithms.

REFERENCES

[1] Darwin Charles, R. (1859). On the origin of species by means of natural selection, or the preservation of favoured races in the struggle for life.

[2] Fogel, D. B. (2006). Evolutionary computation: toward a new philosophy of machine intelligence (Vol. 1). John Wiley & Sons.

[3] Fogel, D. B. (2000). What is evolutionary computation?. *Spectrum, IEEE*, *37*(2), 26-28.

[4] Lamarck, Jean Baptiste Pierre Antoine de. "Philosophie zoologique." (1968).

[5] Back, T., Fogel, D. B., & Michalewicz, Z. (1997). *Handbook of evolutionary computation*. IOP Publishing Ltd.

[6] Larrañaga, P., & Lozano, J. A. (Eds.). (2002). *Estimation of distribution algorithms: A new tool for evolutionary computation* (Vol. 2). Springer Science & Business Media.

[7] Bäck, T., & Schwefel, H. P. (1993). An overview of evolutionary algorithms for parameter optimization. *Evolutionary computation*, *1*(1), 1-23.

[8] Gen, M., & Cheng, R. (2000). *Genetic algorithms and engineering optimization* (Vol. 7). John Wiley & Sons.

[9] Bean, J. C. (1994). Genetic algorithms and random keys for sequencing and optimization. *ORSA journal on computing*, *6*(2), 154-160.

[10] Han, K. H., & Kim, J. H. (2000). Genetic quantum algorithm and its application to combinatorial optimization problem. In *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on* (Vol. 2, pp. 1354-1360). IEEE.

[11] Janikow, C. Z., & Michalewicz, Z. (1991, July). An experimental comparison of binary and floating point representations in genetic algorithms. In *ICGA* (pp. 31-36).

[12] Rahnamayan, S., Tizhoosh, H. R., & Salama, M. M. (2007). A novel population initialization method for accelerating evolutionary algorithms. *Computers & Mathematics with Applications*, *53*(10), 1605-1614.

[13] Deb, K., Anand, A., & Joshi, D. (2002). A computationally efficient evolutionary algorithm for real-parameter optimization. *Evolutionary computation*, *10*(4), 371-395.

[14] Back, T. (1996). Evolutionary algorithms in theory and practice. Oxford Univ. Press.

[15] Bäck, T., & Schwefel, H. P. (1993). An overview of evolutionary algorithms for parameter optimization. *Evolutionary computation*, *1*(1), 1-23.

[16] Jin, Y. (2005). A comprehensive survey of fitness approximation in evolutionary computation. *Soft computing*, *9*(1), 3-12.

[17] Jin, Y., Olhofer, M., & Sendhoff, B. (2002). A framework for evolutionary optimization with approximate fitness functions. *Evolutionary Computation, IEEE Transactions on*, *6*(5), 481-494.

[18] Holland, J. H. (1975). Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. U Michigan Press.

[19] Goldberg, D. E., & Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms. *Foundations of genetic algorithms*, *1*, 69-93.

[20] Spears, W. M., De Jong, K. A., Bäck, T., Fogel, D. B., & De Garis, H. (1993, January). An overview of evolutionary computation. In *Machine Learning: ECML-93* (pp. 442-459). Springer Berlin Heidelberg.

[21] Barkow, J. H., Cosmides, L. E., & Tooby, J. E. (1992). *The adapted mind: Evolutionary psychology and the generation of culture*. Oxford University Press.

[22] Bäck, T., Fogel, D. B., & Michalewicz, Z. (Eds.). (2000). *Evolutionary computation 1: Basic algorithms and operators* (Vol. 1). CRC Press.

[23] Eiben, A. E., & Smith, J. E. (2003). *Introduction to evolutionary computing*. Springer Science & Business Media.

[24] Whitley, D. (2001). An overview of evolutionary algorithms: practical issues and common pitfalls. *Information and software technology*, *43*(14), 817-831.

[25] Alba, E., & Tomassini, M. (2002). Parallelism and evolutionary algorithms. *Evolutionary Computation, IEEE Transactions on*, *6*(5), 443-462.

[26] Schwefel, H. P. P. (1993). *Evolution and optimum seeking: the sixth generation*. John Wiley & Sons, Inc..

[27] Price, K., Storn, R. M., & Lampinen, J. A. (2006). *Differential evolution: a practical approach to global optimization*. Springer Science & Business Media.

