Evolutionary Computation: A review on concepts and issues

Santosh Kumar Satpathy [1], Anirban Mitra[2] , R K Mohanty [3]
(Department of Computer Science and Engineering, Gandhi Engineering College, BPUT, INDIA)

ABSTRACT: *Scientists and Researchers in various field faces computational challenges to find possible solutions which are optimal in nature. A variety of search techniques have been developed for exploring such problem spaces and quite a few approaches are based on algorithms of principles of natural evolution. This paper introduces the fascinating world basic concepts and principles of evolutionary computation, followed by brief overview on four most popular methods, i.e. genetic algorithms, genetic programming, evolutionary strategies, and evolutionary programming. This work is a literature survey in nature, followed by brief discussion on critical issues of evolutionary computational techniques for finding the best possible solution.*

KEYWORDS : *problem spaces , evolutionary computation, genetic algorithms*

I. INTRODUCTION

The word Evolution, from nature's point of view, is a two-step process consisting of random variation and selection (Mayr, 1988, pp. 97-98). A population of individuals is exposed to an environment and responds with a collection of behaviors. Some of these behaviors are better in terms of meeting the demands of the environment as compared to rest. The process of Selection tends to eliminate those individuals that demonstrate inappropriate behaviors. The survivors reproduce, and the genetics underlying their behavioral traits are passed on to their offspring of next generation. In particular, the principle of the 'survival of the fittest' proposed by Charles Darwin (1859) has especially captured the popular imagination. But this replication happens with error and none of the individual genotypes can remain free of random mutations. The process of random genetic variation leads to novel behavioral characteristics, and the process of evolution continue. Over successive generations, increasingly appropriate behaviors accumulate, thus making he species more adaptive to nature. (Atmar, 1994). The manner in which functional adaptations are encoded in genetics is transparent to selection realized behaviors that results from the interaction of the genotype by competitive selection. Useful variations had higher chance of being preserved in the struggle for existence, leading to a process of continual improvement (Darwin, 1859, p. 130). According to Jacob (Jacob, 1977), evolution is entirely opportunistic, and can only work within the variation present in extant individuals. The process of evolution can be modelled with algorithmic approach. Natural evolution does not occur in discontinuous time intervals, but the use of a digital computer requires discrete events. Over successive iterations of variation and selection, the algorithm can drive a population toward particular optima on a response surface that represents the measurable worth of each possible individual that might reside in a population. The term was first proposed in 1991, but the field has a history of four decades. Many independent efforts to simulate evolution on a computer were attempted in the 1950s and 1960s. Three broadly similar avenues of investigation in simulated evolution have survived as main disciplines and are evolution strategies, evolutionary programming, and genetic algorithms. Measure of performance is computed through assessing the "fitness" of each trial solution, and a selection mechanism determines which solutions to retain as "parents" for the subsequent generation. The two papers reprinted during

1990s, Fogel (1994, 1995) and Back et al. (1997), provide surveys of evolutionary computation. Fogel (1994) offered an introduction to a special issue of the IEEE Transactions on Neural Networks devoted to evolutionary computation, while Back et al. (1997) offered the first paper of the IEEE Transactions on Evolutionary Computation. These two publications represent important milestones in the acceptance of evolutionary algorithms as practical tools for addressing Complex problems in engineering.

One of the important areas in present time for research in evolutionary computing is towards development of efficient search techniques based upon the principles of natural evolution. The theory of natural selection proposes that the plants and animals that exist today are the result of millions of years of adaptation to the demands of the environment. At any given time, a number of different organisms may co-exist and compete for the same resources in an ecosystem. The organisms that are most capable of acquiring resources and successfully procreating are the ones whose descendants will tend to be numerous in the future. Evolutionary computation techniques abstract these evolutionary principles into algorithms that may be used to search for optimal solutions to a problem. In a search algorithm, a number of possible solutions to a problem are available and the task is to find the best solution possible in a fixed amount of time. For a search space with only a small number of possible solutions, all the solutions can be examined in a reasonable amount of time and the optimal one found. This exhaustive search, however, quickly becomes impractical as the search space grows in size. Traditional search algorithms randomly sample (e.g., random walk) or heuristically sample (e.g., gradient descent) the search space one solution at a time in the hopes of finding the optimal solution. The key aspect distinguishing an evolutionary search algorithm from traditional algorithms is that it is population-based. Through the adaptation of successive generations of a large number of individuals, an evolutionary algorithm performs an efficient directed search. Evolutionary search is generally better than random search and is not susceptible to the hill-climbing behaviors of gradient based search .

search algorithm from traditional algorithms is that it is population-based. Through the adaptation of successive generations of a large number of individuals, an evolutionary algorithm performs an efficient directed search. Evolutionary search is generally better than random search and is not susceptible to the hill-climbing behaviors of gradient based search.

Algorithm

```
 t: = 0;
initialize P(t);
evaluate P(t);
while not terminate do
P′ (t): = select1(P(t));
P′ ′ (t) : = variation(P′ (t));
evaluate(P′ ′ (t));
P(t+1) : = select2(P′ ′ (t) ∪ Q);
 t:=t+1;
 od
```

II. BASIC CONCEPT OF EVOLUTIONARY COMPUTATION

In an evolutionary algorithm, a representation scheme is chosen by the researcher to define the set of solutions that form the search space for the algorithm. A number of individual solutions are created to form an initial population. The steps are then repeated iteratively until a solution has been found which satisfies a pre-defined termination criterion. Each individual is evaluated using a fitness function that is specific to the problem being solved. Based upon their fitness values, a number of individuals are chosen to be parents. New individuals, or offspring, are produced from those parents using reproduction operators. The fitness values of those offspring are determined. Lastly, the survivors are selected from the old population and the offspring to form the new population of the next generation. The mechanisms determining which and how many parents to select, how many offspring to create, and which individuals will survive into the next generation together represent a selection method.

Different selection methods have been proposed by various researchers in their literature having varied in complexity. Most selection methods ensure that the population of each generation is the same size. The four most common evolutionary algorithms, authors quotes in their work are genetic algorithms (Holland, 1975), genetic programming (Koza, 1992, 1994), evolutionary strategies (Rechenberg, 1973), and evolutionary programming (Fogel et al., 1966). The basic differences between each of the approaches involve the nature of the representation schemes, the reproduction operators, and the selection methods.

2.1 Genetic Algorithms

The most popular technique in evolutionary computation research has been the genetic algorithm. In the traditional genetic algorithm, the representation used is a fixed-length bit string. Each position in the string is assumed to represent a particular feature of an individual, and the value stored in that position represents how that feature is expressed in the solution. Usually, the string is “evaluated as a collection of structural features of a solution that have little or no interactions” (Angeline, 1996, p. 4), (Goldberg, 1989). Some other types of operators has also been developed, but are used less frequently (e.g., inversion, in which a subsequence in the bit string is reversed). A primary distinction that may be made between the various operators is whether or not they introduce any new information into the population. Crossover, for example, does not while mutation does. All operators are also constrained to manipulate the string in a manner consistent with the structural interpretation of genes.

2.2 Genetic Programming

In genetic programming, the representation used is a variable-sized tree of functions and values. Each leaf in the tree is a label from an available set of value labels. Each internal node in the tree is label from an available set of function labels. The entire tree corresponds to a single function that may be evaluated (Angeline, 1996). Typically, the tree is evaluated in a leftmost depth-first manner. A leaf is evaluated as the corresponding value. A function is evaluated using as arguments the result of the evaluation of its children.

Genetic algorithms and genetic programming are similar in most other respects, except that the reproduction operators are tailored to a tree representation. The most commonly used operator is subtree crossover, in which an entire subtree is swapped between two parents. In a standard genetic program, all values and functions are assumed to return the same type, although functions may vary in the number of arguments they take. This closure principle (Koza, 1994) allows any subtree to be considered structurally on par with any other subtree, and ensures that operators such as sub-tree crossover will always produce legal offspring.

2.3 Evolutionary Strategies

In evolutionary strategies, the representation used is a fixed-length real-valued vector. As with the bitstrings of genetic algorithms, each position in the vector corresponds to a feature of the individual. However, the features are considered to be behavioral rather than structural. (Angeline, 1996, p. 4). The main reproduction operator in evolutionary strategies is Gaussian mutation, in which a random value from a Gaussian distribution is added to each element of an individual's vector to create a new offspring. Another operator that is used is intermediate recombination, in which the vectors of two parents are averaged together, element by element, to form a new offspring. The effects of these operators reflect the behavioral as opposed to structural interpretation of the representation since knowledge of the values of vector elements is used to derive new vector elements (Spears et al., 1993).

2.4 Evolutionary Programming

The representations used in evolutionary programming are typically tailored to the problem domain (Spears et al., 1993). One representation commonly used is a fixed-length real-valued vector. The primary difference between evolutionary programming and the previous approaches is that no exchange of material between individuals in the population is made. Thus, only mutation operators are used. For real-valued vector representations, evolutionary programming is very similar to evolutionary strategies without recombination. A typical selection method is to select all the individuals in the population to be the N parents, to mutate each parent to form N offspring, and to probabilistically select, based upon fitness, N survivors from the total 2N individuals to form the next generation.

III. ON EVOLUTIONARY ALGORITHM

The common underlying idea behind the technique of evolutionary process is similar. For a given population of individuals, the environmental pressure causes natural selection (survival of the fittest) and hereby the fitness of the population grows (Rudolph, 1997). It is easy to call such a process as optimization. For a given objective function to be maximized one can randomly create a set of candidate solutions and use the objective function as an abstract fitness measure such that the higher, is the better. Based on this fitness, some of the better candidates are chosen to seed the next generation by applying recombination and/or mutation. Recombination is applied to two selected candidates (so) called parents, and results in one or two new candidates (called) children. Mutation is applied to one candidate and results in one new candidate. Further, applying recombination and mutation leads to a set of new candidates (the offspring). Based on their fitness these offspring compete with the old candidates for a place in the next generation. This process can be continued until a solution is found or a previously set time limit is reached. According to Darwin, the emergence of new species, adapted to their environment, is a consequence of the interaction between the survival of the fittest mechanism and undirected variations. It may be noted that a selection operators can be either deterministic, or stochastic. In the latter case fitter individuals have a higher chance to be selected than less fit ones, but typically even the weak individuals have a chance to become a parent or to survive.

A generalized evolutionary algorithm can be given as described as:

```
Step-1: Initialize population with random individuals (candidate solutions)
Step-2: Evaluate (compute fitness of) all individuals
WHILE not stop
DO Select genitors from parent population.
Step-3: Create offspring using variation operators on genitors
Step-4: Evaluate newborn offspring Replace some parents by some offspring. 'OD'
```

It may be noted that this scheme falls in the category of generate-and-test, also known as trial-and-error, algorithms. The fitness function represents a heuristic estimation of solution quality and the search process is driven by the variation operators (recombination and mutation creating new candidate solutions) and the selection operators. Evolutionary algorithms are distinguished within in the family of generate-and test methods by being population based, i.e., process a whole set of candidate solutions and by the use of recombination to mix information of two candidate solutions.

IV. CRITICAL AND CURRENT ISSUES IN EVOLUTIONARY COMPUTATION

One of the crucial issues during executing an evolutionary algorithm is to try to preserve the genetic diversity of the population as long as possible. Reverse to many other optimization methods, evolutionary algorithm uses a whole population of individuals, which perhaps one of the main cause of its efficiency (Beyer, 2000). But, if that population starts to concentrate in a very narrow region of the search space, all advantages of handling many different individuals perish and only the burden of computing their fitnesses remains. This phenomenon is called premature convergence. In general, there are two driving forces behind an evolutionary algorithm, selection and variation. The first one represents a push toward quality and is reducing the genetic diversity of the population. The second one, implemented by recombination and mutation operators, represents a push toward novelty and is increasing genetic diversity. For making a evolutionary algorithm to work properly, an appropriate balance between these two forces has to be maintained, which is presently very challenging and there is not much conceptual support for a practical design.

4.1 Current Issues

Researchers in field of evolutionary algorithm have begun to examine more complex representation schemes and to apply a variety of selection methods. Many genetic algorithm researchers are examining the use variable-length representations and analyzing how such representations grow in size over the course of evolution (Wu & Lindsay, 1996). Many genetic algorithms now use selection methods, such as elitist recombination, in which parents compete with their offspring for survival into the next generation (Schwefel, 1995), (Thierens, 1997). Some genetic programming researchers have begun to examine the effects of allowing multiple types of functions and values into the representation. The benefits of such strongly typed genetic programming are only beginning to be explored (Haynes et al., 1996).

V. CONCLUSION

Natural evolution can be considered as a powerful problem solver achieving more adaptable species in nature. Computer based evolutionary processes can also be used as efficient problem solvers for optimization, constraint handling, machine learning and modeling tasks. Many real world phenomena from the study of life, economy, and society can be investigated by simulations based on evolving systems. The concept of evolutionary algorithm is designed form an emerging field of applications of the Darwinian ideas. The authors expect that computer applications based on evolutionary principles will gain popularity in the coming years. This work focused on basic concepts and principles of evolutionary computation, followed by brief overview on four most popular methods, i.e. genetic algorithms, genetic programming, evolutionary strategies, and evolutionary programming. Being a literature survey in nature, this work discusses on concepts and critical issues of evolutionary computational techniques for finding the best possible solution.

REFERENCES

[1]. Angeline, P.J. (1996) "Genetic programming's continued evolution," Chapter 1 in K.E. Kinnear, Jr. and P.J. Angeline (Eds.), Advances in Genetic Programming 2. Cambridge, MA: MIT Press, pp. 1 -20.

[2]. C. Darwin, (1859) "The Origin of Species by Means of Natural Selection or the Preservation of Favoured Races in the Struggle for Life", Mentor Reprint, 1958, NY.

[3]. D. B. Fogel (1994) "An introduction to simulated evolutionary optimization", IEEE Trans- Neural Networks, Vol. 5:1, pp. 3-14.

[4]. D. B. Fogel (1995) "Evolutionary Computation: Toward a New Philosophy of Machine Intelligence". IEEE Press, Piscataway, NJ.

[5]. D. E. Goldberg (1989) "Genetic Algorithms in Search, Optimization and Machine Learning", Addison Wesley, Reading, MA.

[6]. Darwin, C. (1859) "On the Origin of Species by Means of Natural Selection", London: John Murray.

[7]. E. Mayr (1988) "Toward a New Philosophy of Biology: Observations of an. Evolutionist", Belknap, Harvard.

[8]. F. Jacob,(1977) "Evolution and tinkering," Science, Vol. 196, pp. 1161-1166.

[9]. Fogel, L.J., Owens, A.J. & Walsh, M.J. (1966) “Artificial Intelligence through Simulated Evolution”. New York: John Wiley.

[10]. G. Rudolph (1997) “Convergence Properties of Evolutionary Algorithms”. Verlag Dr. Kovacs, Hamburg, 1997.

[11]. H. G. Beyer (2000) “The Theory of Evolution Strategies”, Natural Computing Series, Springer, Berlin.

[12]. H. P. Schwefel (1995), “Evolution and Optimum Seeking”, Sixth-Generation Computer Technology Series. Wiley, New York.

[13]. Haynes, T.D., Schoenefeld, D.A. & Wainwright, R.L. (1996) “Type inheritance in strongly typed genetic programming,” Chapter 18 in K.E. Kinnear, Jr. and P.J. Angeline (Eds.), Advances in Genetic Programming 2. Cambridge, MA: MIT Press.pp. 359-376.

[14]. Holland, J.H. (1975) “Adaptation in Natural and Artificial Systems”. Ann Arbor, MI: University of Michigan Press.

[15]. Koza, J.R. (1992) “Genetic Programming: On the Programming of Computers by Means of Natural Selection”. Cambridge, MA: MIT Press.

[16]. Koza, J.R. (1994) “Genetic Programming II: Automatic Discovery of Reusable Programs”. Cambridge, MA: MIT Press.

[17]. Spears, W.M., DeJong, K.A., Back, T., Fogel, D.B., & deGaris, H. (1993) “An overview of evolutionary computation,” Proceedings of the 1993 European Conference on Machine Learning.

[18]. T. Back, U. Hammel, and H.-P. Schwefel (1997), “ Evolutionary computa tion: comments on the history and current state," IEEE Trans. Evolutionary Computation, Vol 1:1, pp-3-17.

[19]. Thierens, D. (1997) “Selection schemes, elitist recombination, and selection intensity,” Proceedings of the Seventh International Conference on Genetic Algorithms. San Francisco, CA: Morgan Kauffman, pp. 152-159.

[20]. W. Atmar (1994), “ Notes on the simulation of evolution", IEEE Trans. Neural Networks, Vol. 5:1, pp. 130-147, 1994

[21]. Wu, A. & Lindsay, R.K. (1996) “A survey of intron research in genetics,” In Voigt, H., Ebelin, W., Rechenberg, I., & Schwefel, H. (Eds.), Parallel Problem Solving from Nature - PPSN IV. Berlin: Springer-Verlag, p. 101-110.