More Efficient Genetic Algorithm For Solving Optimization Problems

S. Ghoshray, K. K. Yen
Department of Electrical and Computer Engineering
Florida International University
Miami, Florida 33199
E-mail: Sghosh01 @solix.fiu.edu

Abstract

Genetic algorithms (GA) are stochastic search techniques based on the mechanics of natural selection and natural genetics. By using genetic operators and cumulative information, genetic algorithms prune the search space and generate a set of plausible solutions. This paper describes an efficient genetic algorithm defined as Modified Genetic Algorithm (MGA). The proposed algorithm is developed by making a marriage between the Simple Genetic Algorithm (SGA) and the Simulated Annealing (SA). In this proposed algorithm, all the conventional genetic operators, such as, selection, reproduction, crossover, mutation, have been used, but they have been modified by a set of new functions such as, a selection function 1, a selection function 2, a mutation function, etc., which utilizes the concept of successive descent as seen in simulated annealing. In this way, MGA can be implemented to solve various optimization problems more accurately and quickly.

1. Introduction

John Holland laid the foundation of GAs with the goal to create computer algorithms by simulating the characteristics of a natural system [1]. It was intended that, if nature can produce from a random population, a population with individuals that are better fit to the environment, it is possible to develop an algorithm to solve complex problems by utilizing the concept from nature. In this respect, a wide but diverse range of applications [2, 3, 4, 5] bear testimony to the fact that GAs are very useful in solving complicated problems by mimicking some facets of natural evolution. Similarly, Simulated Annealing (SA) is a stochastic based computational technique that owes its origin to statistical mechanics. Kirkpatrick et al [8] were the first to propose and demonstrate the application of simulation techniques borrowed from statistical physics to the large optimization problems.

A combinatorial optimization problem is either a minimization problem or a maximization problem and consists of : (1) a set of instances, (2) a finite set of candidate solutions for each instance and (3) a function that assigns to each instance and each candidate solution a positive solution value. This combinatorial optimization problem is solved by finding an optimal solution for each instance of the optimization problem. In this research, we propose a modified genetic algorithm by combining the powers of both, a simple genetic algorithm and simulated annealing. The solution we try to obtain here is found near to the optimum, but with more accuracy and in a shorter run-time. We have organized the paper as follows: section II provides an overview of GAs, section III gives a probabilistic view of different genetic operators, section IV introduces the concept of SA by presenting some pseudo-codes. The next section brings us to the proposed Modified Genetic Algorithm (MGA) with different new definitions and algorithms. Finally, we conclude by summarizing and focusing on future work.

2. A Preview of Genetic Algorithms

Genetic algorithms represent a highly idealized model of a natural process and as such can be legitimately viewed as a very high level of abstraction. Genetic algorithms are loosely based upon the Darwinian principles of biological evolution. Biological strategies of behavior adaptation and synthesis are used to enhance the probability of survival and propagation during their evolution. Environmental pressures requiring these strategies have affected profound changes in biological organisms. These changes are manifested in structural and functional organization, and internal knowledge representations [6]. A GA randomly generates a set of possible solutions to the problem which is being investigated. This is termed the initial generation. Each successive incremental improvement in a solution structure becomes the basis for the next generation. The process continues until the desired number of generations has been completed.

In a natural genetic system, the chromosomes consist of genes. Each gene has a value and position. The combination of chromosomes form the total genetic prescription for the construction and operation of some organism [7]. While developing a GA, strings and characters are used in order to simulate chromosomes and genes corresponding to the natural genetic system. When using a genetic algorithm to solve a problem, the problem is represented by a string, and an evaluation function is defined. The evaluation function uses the value of the string as a parameter to evaluate the results of the problem. Similar to natural systems, in an artificial genetic system we have populations and generations. A set of strings is used to represent the populations. Each string is evaluated through the evaluation function and the new generation is formed by using the specific genetic operators.

A genetic algorithm is an interactive procedure that maintains a population of strings which constitute the set of candidate solutions to the specific problem. During each generation, the strings in the current population are rated for their effectiveness as solutions. On the basis of these evaluations, a new population of candidate solutions is formed using genetic operators such as reproduction, crossover, and mutation. There are four major preparatory steps required to use the conventional genetic algorithm to solve a problem [8], namely determining

(1) the representation scheme,

(2) the fitness measure,

(3) the parameters and variables for controlling the algorithm, and

(4) a way of designating the result and a criterion for terminating a run

3. Characteristics Of The Genetic Operators

In this section, we are going to briefly describe the genetic operations mentioned earlier. The genetic operation of *reproduction* is based on the Darwinian principle of reproduction and survival of the fittest. Probability plays a very important role in the overall implementation of genetic algorithm. In the reproduction operation, an individual is probabilistically selected from the population on the basis of its fitness and the selected individual is then copied into the next generation of the population without any change. In this selection process, the better an individual's fitness is the more likely it is to be selected.

The genetic operation of *crossover* allows the creation of new individuals which accounts for the generation of new points in the search space to be tested. Crossover starts with two parents independently selected probabilistically from the population on the basis of their fitness.

Individuals from the population can be selected and, in general, are selected more than once during a generation to participate in the operations of reproduction and crossover. Indeed, the differential rates of survival, reproduction, and participation in genetic operations by more fit individuals is an essential part of the genetic algorithm.

The operation of *mutation* begins with the probabilistic selection of an individual from the population on the basis of its fitness. A mutation point along the string is chosen at random, and the single character at that point is randomly changed. The altered individual is then copied into the next generation of the population. The usefulness of mutation comes in situations requiring the restoration of genetic diversity that may have been lost in a population because of premature convergence. Mutation is used very sparingly in most genetic algorithm work.

The Darwinian selection of individuals to participate in the operation of reproduction, crossover, and mutation on the basis of their fitness is an essential aspect of the genetic algorithm. When an individual is selected on the basis of its fitness to be copied (with or without mutation) into the next generation of the population, the effect is that the new generation contains the characteristics it embodies. These characteristics consists of certain values at certain positions of the character string and, more importantly, certain combinations of values situated at two or more positions of the string. When two individuals are selected on the basis of their fitness to be recombined, the new generation contains the characteristics of both of these parents. The probabilistic selection used in the genetic algorithm is an essential aspect of the algorithm. The genetic algorithm allocates every individual, however poor its fitness may be, some chance of being selected so that it can participate in the operations of reproduction, crossover, and mutation. In that respect, the genetic algorithm is not merely a greedy hill climbing algorithm, rather, the genetic algorithm contains the characteristics of simulated annealing [9, 10].

4. Simulated Annealing Algorithms

Simulated Annealing (SA) is a stochastic computational technique derived from statistical mechanics for finding near globally minimum-cost solutions to large optimization problems. Kirkpatrick et al [9]. First proposed and demonstrated the application of simulation techniques from statistical physics to problems of combinatorial optimization. Finding the global minimum value of an objective function with many degrees of freedom subject to conflicting constraints is an NP-complete problem. Therefore, the objective function will tend to have many local minima. A procedure for solving optimization problems of the above nature can be implemented by following the Successive Descent algorithm by kirkpatrick et al [8] which follows the evolution of a solid thermodynamic equilibrium with a decreasing succession of temperature values. The pseudo code for this Descent algorithm goes as follows:

```
begin Successive Descent
        Select an initial state i in the search space of E such that $i \in E$
        Repeat
            Generate a random state j in the search space of E such that j is a neighbor of i
            if Cj - Ci < 0
            then i: = j
        until Cj > Ci for all j in the neighborhood of i
end Successive Descent
```

In this algorithm, E is a finite set of possible configurations that constitute a finite search space. Ci is the cost function associated with configuration i. In SA, the algorithm attempts to accept a neighboring configuration that increases the cost function C. The probability of accepting such a neighborhood move is determined by Pi = exp(-s/T), where s = an increase in cost function = Cj - Ci, T = the control parameter in the optimization process analogous to temperature in SA. With this in mind, therefore, the minimization algorithm can be written in pseudo-code as follows:

```
begin minimization
    select an initial state $i \in E$
    select an initial control parameter T > 0
    select parameter change counter t = 0
    Repeat
        set repetition counter r = 0
        Repeat
            Generate state j : j is a neighbor of i, $i \in E$
            compute S = Cj - Ci
            If S < 0 then i : = j
                else if rand (0,1) < exp(-s/T)
                    then i: = j
            r : = r+1
        until r = R (t)
        t : = t +1
        T : = T (t)
    until stopping Rule: = True
end minimization
```

5. Proposed Genetic Algorithm

In this section, we are going to explain the workings of our proposed genetic algorithm based on the algorithms explained in the earlier sections. This proposed Modified Genetic Algorithm (MGA) is developed by unifying Genetic Algorithm (GA) and Simulated Annealing (SA). The following pseudo code presents an insight into proposed MGA:

```
begin   MGA
        Make initial population at random
        While Termination Criterion is True Do
```

```
            Begin
            Select parents from the population
            Make the selected parents to produce children
            Add the children to the population

            While $p_m < 0.005$ Do
                Begin
                Mutate
                End
            Reduce population
            End
            Output the result

    end     MGA
```

In the above algorithm, p_m is the probability of mutation. The success of the MGA depends on how effective we are in defining the genetic operators used in this modified algorithm. Let us now define the genetic operators as follows:

A. Size of Initial Population: i ; this represents the randomly chosen initial population which depends on the problem specification. That means, i depends on the search space E and, also $i \in E$

B. Set of Possible regular populations: S_i ; this represents all the possible populations with i elements.

C. Set of Extended Populations: S_{i+} ; this represents all the populations which is generated by adding the children into the original populations.

D. Selection Functions 1: f_{S1} ; this function randomly selects the initial populations from the search space E.

E. A Selection Function Z: $f_{S2}(p, x)$; it is a function of two variables. The variable p is the parameter that randomizes the selection function f_{SZ} and x is any member of S; $x \in S_i$ such that $f_{S2}(p,x)$ selects the set of parents that are chosen for matting at the onset of the algorithm.
$f_{S2}(p, x) = y \subseteq x , p \in P$

F. A Reproduction Function: f_{RP} (q, y) ; this function is randomized by q and it works over populations $y \subseteq S$; to produce offspring z.
$f_{RP}(q,y) = z, z \in S_i , q \in Q.$

G. A Mutation Function: f_M (r, z) ; this function is randomized by parameter r, $r \in R$. The goal here is not to be entrapped in the local minima and accept the neighborhood move by invoking the following function call:
f_M (r, z) = Z' , Z' $\in$ neighborhood of Z such that
C (z') > C (z) , that is, cost function assigned with
configuration Z' is greater than that assigned with Z.

H. A Reduction Function: $f_{Rd}(r, t_+) = t$; here the extended populations are reduced to regular populations of original size, signaling the end of the genetic programming loop.

I. An Evaluation Function: f_{eval} (x) ; this is a Boolean function that controls the termination of the algorithm.

Based on the above notations and definitions, the MGA is presented as follows:

```
    begin MGA (Modified Genetic Algorithm)

        Get any one initial population applying
        $f_{S1} = x \in S_i$

        While $f_{eval}(x)$ = True do

        Begin

            Get p, q, r, s
            $y = f_{SZ}(p, x)$
            $z = f_{RP}(v, y)$
            $z' = f_m(r, z)$
            $t = x \cup z'$
            $x = f_{Rd}(s, t)$

        end
        output the present population

    end MGA
```

6. Results And Discussions

In this section, we are going to have a discussion on the comparative performances of the proposed genetic algorithm, simple genetic algorithm and simulated annealing. The problem we will focus on is to maximize the function f (x_1 ,

$x_2) = 21.5 + x_1 \text{Sin}(4\pi x_1) + x_2 \text{Sin}(20\pi x_2)$ with the given constraints as follows : $-3.0 \leq x_1 \leq 12.1$, $4.1 \leq x_2 \leq 5.8$. Also, in this experiment, the assumptions taken are as detailed below.

Assumptions: (i) Population size = 20, (ii) Probability of crossover $p_c = 0.25$, (iii) Probability of mutation $p_m = 0.001$. We have calculated the value of total length of chromosome = 7 + 8 = 15. The chromosomes for the proposed genetic algorithm will look like as follows : V_1 = 100110100000000, V_2 = 110001001001101, V_3 = 00001000001100, V_4 = 100001100010110, etc. Some of the probabilities of selection p_i for chromosomes are calculated as $p_1 = 0.067099$, $p_2 = 0.019547$, ... $p_{20} = 0.35244$. The probability of selection is calculated as $p_i = \text{eval}(v_i) / F$ in which F is the total fitness of the population calculated as $F = \sum_{1}^{20} eval(V_i)$. Let us now define two characteristics that is used to compare the performances of different genetic algorithms. Accuracy (μ) is defined as the value of the best found chromosome in the proposed algorithm relative to the value of the global optimum found in the original GA. Therefore, $\mu = f_{best} / f_{global}$. Imprecision (λ) is defined as the standard deviation of the optimal vector found. This measure evaluates the closeness of individual components selected in the found optimal
chromosome. Therefore,

$$\lambda = \sqrt{\frac{\sum_{i=1}^{\#\,element} (\dot{index}_i)^2}{\#\,elements - 1}}$$

Our experiments have shown the values of accuracy μ for different algorithms are as follows: 95.71 for GA, 96.47 for SA and 98.45 for the proposed MGA. The values of imprecision λ for different algorithms are as follows: 2.243 for GA, 2.108 for SA and 2.012 for the proposed MGA.

7. Conclusion

This paper has established the drawbacks of the classical GA in solving optimization problem. We have also presented a refined genetic algorithm by efficiently combining simple genetic algorithms with Simulated Annealing. The empirical results show that Modified Genetic Algorithms provide efficient search heuristics for solving various optimization problems. The model proposed has a short run time compared to that of SGA and the results produced have been qualitatively correct. Also, the results show that MGA provides less imprecision. However, the main difficulty remaining is in the efficient modeling of selection function f_{s1} that randomly selects the initial population from the search space. We hope to address this issue in more detail in a future publication. Our forthcoming research would also focus on developing more efficient parallel selection algorithms by using fractal methods. The attached figure shows a flowchart to describe the workings of genetic algorithms.

References

1. Holland, J. H. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, 1975.

2. Pettey, C. B., Leuze, M. R., Grefenstette, J. J. *A Parallel Genetic Algorithm.* In Proceedings of the Second International Conference on Genetic Algorithms, 1987.

3. Grefenaatette, J. J. & Baker, J. E. *How Genetic Algorithms Works: A Critical Look at Implicit Parallelism.* In Proceedings of the Third International Conference on Genetic Algorithms, 1989.

4. Schaffer, J. D., Caruana, R. A., Eshelman, L. J. & Das, R. *A Study of Control Parameters Affecting Online Performance of Genetic Algorithm for Function Optimization.* In Proceedings of the Third International Conference on Genetic Algorithms, 1989.

5. Lucasins, C. B.,. & Kateman, G. *Application of Genetic Algorithms in Chemometric.* In Proceedings of the Third International Conference on Genetic Algorithms, 1989.

6. Austin, S. *An Introduction to Genetic Algorithms.* AI Expert, March 1990.

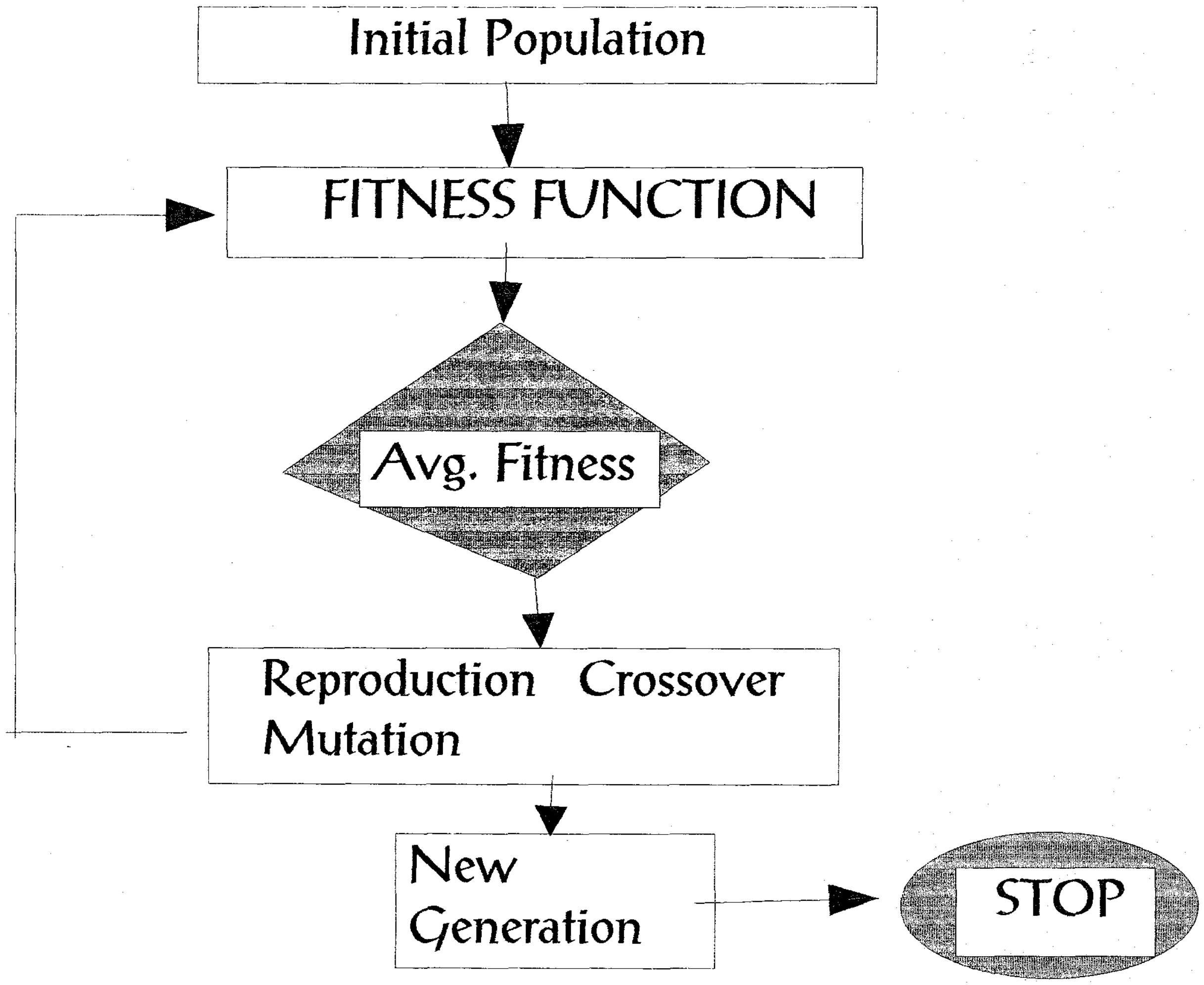
7. Goldberg, D. E. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, 1989.

8. Koza, J. R. *Genetic Programming II Automatic Discovery of Reusable Programs.* The MIT Press, 1994.

9. Kirkpatrick, S., Gelatt, C. D. & Vechi, M. P. *Optimization by Simulated Annealing.* Science 220, 1983. pgs. 671-680.

10. Aarts, E. & Korst, J. *Simulated Annealing and Boltzman Machines.* Wiley 1989.

Figure 1 Flowchart Showing The Workings Of Genetic Algorithm


Initial Population
FITNESS FUNCTION
Avg. Fitness
Reproduction Crossover Mutation
New Generation
STOP