Introducing Genetic Algorithm as an Intelligent Optimization Technique

Article · June 2014

DOI: 10.4028/www.scientific.net/AMM.568-570.793

2 authors:

Ali Ashkzari
Vrije Universiteit Amsterdam
3 PUBLICATIONS **4** CITATIONS
SEE PROFILE

Aydin Azizi
German University of Technology in Oman
39 PUBLICATIONS **59** CITATIONS
SEE PROFILE

Applied Mechanics and Materials *Online: 2014-06-10*
ISSN: 1662-7482, Vols. 568-570, pp 793-797
doi:10.4028/www.scientific.net/AMM.568-570.793
© 2014 Trans Tech Publications, Switzerland

Introducing Genetic Algorithm as an intelligent optimization technique

Ali Ashkzari[1,a], Aydin Azizi[1,b]

[1]Department of Computer Engineering, Eastern Mediterranean University, North Cyprus.

[2]Department of Mechanical Engineering, Eastern Mediterranean University, North Cyprus.

[a] Ali.Ashkzar@gmail.com, [b] azizi_aydin@yahoo.com

Keywords: Genetic algorithm; biological evaluation; optimization problem; adaptive problem solving; optimization technique.

Abstract. The Genetic Algorithm (GA) is a stochastic global search method that mimics the metaphor of natural biological evolution. GA operates on a population of potential solutions applying the principle of survival of the fittest to produce (hopefully) better and better approximations to a solution. Genetic algorithms are particularly suitable for solving complex optimization problems and for applications that require adaptive problem solving strategies. Here, in this paper genetic algorithm is introduced as an optimization technique.

Introduction

In nature, individuals best suited to competition for scanty resources survive. Evolving to keep adapted to a changing environment is essential for the members of any species. Although evolution manifests itself as changes in the species' features, it is in the species' genetical material that those changes are controlled and stored. Specifically evolution's driving force is the combination of natural selection and the change and recombination of genetic material that occurs during reproduction [1]. Evolution is an astonishing problem solving machine. It took a soup of primordial organic molecules, and produced from it a complex interrelating web of live beings with an enormous diversity of genetic information. Enough information to specify every characteristic of every species that now inhabits the planet. The force working for evolution is an algorithm, a set of instructions that is repeated to solve a problem. The algorithm behind evolution solves the problem of producing species able to thrive in a particular environment [2]. Genetic algorithms, first proposed by Holland in 1975 [3], are a class of computational models that mimic natural evolution to solve problems in a wide variety of domains [4]. Genetic algorithms are particularly suitable for solving complex optimization problems and for applications that require adaptive problem solving strategies.

Optimization techniques.

Placement and routing are two search intensive tasks. Even though *agent objects* use knowledge to reduce search time, a great deal of searching is still necessary. A good proportion of this search time will be spent optimizing the components' placement in the layout. In searching for optimum solutions, optimization techniques are used and can be divided into three broad classes as follow [4].

Numerical techniques. Numerical techniques are divided into direct and indirect methods. Indirect methods search for local extremes by solving the usually non-linear set of equations resulting from setting the gradient of the objective function to zero. The search for possible solutions (function peaks) starts by restricting itself to points with zero slope in all directions. Direct methods, such as those of Newton or Fibonacci, seek extremes by "hopping" around the search space and assessing the gradient of the new point, which guides the search. This is simply the notion of "hill climbing", which finds the best local point by climbing the steepest permissible gradient. These techniques can be used only on a restricted set of "well behaved" functions.

Enumerative techniques. Enumerative techniques are based on search every point related to the function's domain space, one point at a time. They are very simple to implement but usually require significant computation. These techniques are not suitable for applications with large domain spaces. Dynamic programming is a good example of this technique.

Guided random search techniques. Guided random search techniques based on enumerative techniques but use additional information to guide the search. Two major subclasses are simulated annealing and evolutionary algorithms. Both can be seen as evolutionary processes, but simulated annealing uses a thermodynamic evolution process to search minimum energy states. Evolutionary algorithms use natural selection principles. This form of search evolves throughout generations, improving the features of potential solutions by means of biological inspired operations. Genetic Algorithms (GAs) are a good example of this technique.

Agents could use many techniques for placement optimization. Currently it uses the Eval Agent class to implement a genetic algorithm. However, other classes could be created to implement other methods, such as Min-Cut, Force Directed or simulated annealing. They could be used in place of the Eval Agent class, without any other modification to other parts of the program. A future implementation using simulated annealing is very probable, but the genetic algorithm was chosen, as the first implementation, because of its novelty and because it has shown better results than simulated annealing [6].

The algorithm

A genetic algorithm emulates biological evolution to solve optimization problems. It is formed by a set of individual elements (the population) and a set of biological inspired operators that can change these individuals. According to evolutionary theory only the individuals that are the more suited in the population are likely to survive and to generate off-springs, thus transmitting their biological heredity to new generations. In computing terms, genetic algorithms map strings of numbers to each potential solution. Each solution becomes an individual in the population, and each string becomes a representation of an individual. There should be a way to derive each individual from its string representation. The genetic algorithm then manipulates the most promising strings in its search for an improved solution. The algorithm operates through a simple cycle:

a) Creation of a population of strings.

b) Evaluation of each string.

c) Selection of the best strings.

d) Genetic manipulation to create a new population of strings**.**

Each cycle produces a new generation of possible solutions (individuals) for a given problem. At the first stage, a population of possible solutions is created as a start point. Each individual in this population is encoded into a string (the chromosome) to be manipulated by the genetic operators. In the next stage, the individuals are evaluated, first the individual is created from its string description (its chromosome) and its performance in relation to the target response is evaluated. This determines how fit this individual is in relation to the others in the population. Based on each individual's fitness, a selection mechanism chooses the best pairs for the genetic manipulation process. The selection policy is responsible to assure the survival of the fittest individuals. The manipulation process applies the genetic operators to produce a new population of individuals, the offspring, by manipulating the genetic information possessed by the pairs chosen to reproduce. This information is stored in the strings (chromosomes) that describe the individuals. Two operators are used: Crossover and mutation. The offspring generated by this process take the place of the older population and the cycle is repeated until a desired level of fitness in attained or a determined number of cycles are reached.

The Objective and Fitness Functions. The objective function is used to provide a measure of how individuals have performed in the problem domain. In the case of a minimization problem, the fit individuals will have the lowest numerical value of the associated objective function. This raw measure of fitness is usually only used as an intermediate stage in determining the relative performance of individuals in a GA. Another function, the *fitness function*, is normally used to transform the objective function value into a measure of relative fitness, thus:

$$F(x) = g(f(x)) \qquad (1)$$

Where *f* is the objective function, *g* transforms the value of the objective function to a non-negative number and *F* is the resulting relative fitness. This mapping is always necessary when the objective function is to be minimized as the lower objective function values correspond to fitter individuals. In many cases, the fitness function value corresponds to the number of offspring that an individual can expect to produce in the next generation. A commonly used transformation is that of proportional fitness assignment. The individual fitness, $F(x_i)$, of each individual is computed as the individual's raw performance, $f(x_i)$, relative to the whole population, i.e.,

$$F(x_i) = \frac{f(x_i)}{\sum_{i=1}^{N_{ind}} f(x_i)} \qquad (2)$$

Where N_{ind} is the population size and *xi* is the phenotypic value of individual *i*. whilst this fitness assignment ensures that each individual has a probability of reproducing according to its relative fitness, it fails to account for negative objective function values. A linear transformation which offsets the objective function is often used prior to fitness assignment, such that,

$$F(x) = af(x) + b \qquad (3)$$

Where a is a positive scaling factor if the optimization is maximizing and negative if we are minimizing. The offset b is used to ensure that the resulting fitness values are non-negative.

Roulette Wheel Selection Methods. Many selection techniques employ a "roulette wheel" mechanism to probabilistically select individuals based on some measure of their performance. A real-valued interval, Sum, is determined as either the sum of the individuals' expected selection probabilities or the sum of the raw fitness values over all the individuals in the current population. Individuals are then mapped one-to-one into contiguous intervals in the range [0, Sum]. The size of each individual interval corresponds to the fitness value of the associated individual. To select an individual, a random number is generated in the interval [0, *Sum*] and the individual whose segment spans the random number is selected. This process is repeated until the desired numbers of individuals have been selected. The basic roulette wheel selection method is stochastic sampling with replacement (SSR). Here, the segment size and selection probability remain the same throughout the selection phase and individuals are selected according to the procedure outlined above. SSR gives zero bias but a potentially unlimited spread. Any individual with a segment size > 0 could entirely fill the next population

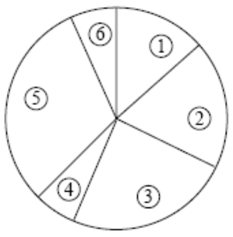

Figure 1. Roulette Wheel Selection

Crossover. Crossover is one of the genetic operators used to recombine the population genetic material. It takes two chromosomes and swaps part of their genetic information to produce new chromosomes. This operation is similar to sexual reproduction in nature. After the crossover point has been randomly chosen, portions of the parent's chromosome (strings) Parent 1 and Parent 2 are combined to produce the new offspring Son.

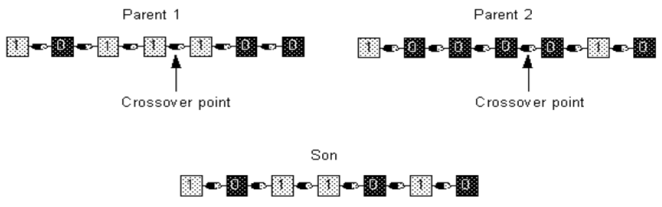

Figure 2. Crossover

The selection process associated with the recombination made by crossover assures that special genetic structures, called building blocks, are retained for future generations. These building blocks represent the fit genetic structures in the population.

Mutation. The recombination process alone cannot explore search space sections not represented in the population's genetic structures. This could make the search get stuck around local minima. Here mutation goes into action. The mutation operator introduces new genetic structures in the population by randomly changing some of its building blocks, helping the algorithm escape local minima traps. Since the modification is totally random and thus not related to any previous genetic structures present in the population, it creates different structures related to other sections of the search space. Mutation is implemented by occasionally altering a random bit from a chromosome (string), the Fig. shows the operator being applied to the fifth element of the chromosome.

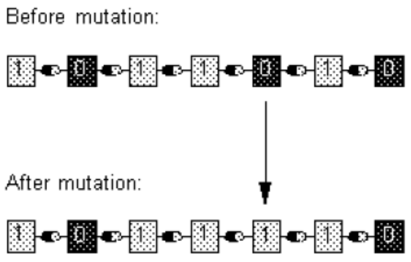

Figure 3. Mutation

roblem dependent parameters

This description of the genetic algorithms' computational model reviews the steps needed to create the algorithm. However, a real implementation takes account of a number of problem-dependent parameters. For instance, the offspring produced by the genetic manipulation (the next population to be evaluated) can either replace the whole population (generational approach) or just its less fitted members (steady-state approach). Problem constraints will dictate the best option. Other parameters to be adjusted are the population size, crossover and mutation rates, evaluation method, and convergence criteria.

Encoding

Critical to the algorithm performance is the choice of underlying encoding for the solution of the optimization problem (the individuals on the population). Traditionally, binary encodings have being used because they are easy to implement. The crossover and mutation operators described earlier are specific to binary encodings. When symbols other than 1 or 0 are used, the crossover and mutation operators must be tailored accordingly.

A large number of optimization problems have continuous variables. A common technique for encoding them in the binary form uses a fixed-point integer encoding, each variable being coded using a fixed number of bits. The binary code of all the variables can then be concatenated in the strings of the population. A drawback of encoding variables as binary strings is the presence of Hamming cliffs: large Hamming distances between the codes of adjacent integers. For instance, 01111 and 10000 are integer representations of 15 and 16, respectively, and have a Hamming distance of 5. For the genetic algorithm to change the representation from 15 to 16, it must alter all bits simultaneously. Such Hamming cliffs present a problem for the algorithm, as both mutation and crossover cannot overcome them easily.

It is desirable that the encoding makes the representation as robust as possible. This means that even if a piece of the representation is randomly changed, it will still represent a viable individual. For instance, suppose that a particular encoding scheme describes a circuit by the position of each of its components and a pointer to their individual descriptions. If this pointer is the description's memory address, it is very unlikely that, after a random change in its value, the pointer will still point to a valid description. But, if the pointer is a binary string of 4 bits pointing into an array of 16 positions holding the descriptions, regardless of the changes in the 4 bit string, the pointer will always point to a valid description. This makes the arrangement more tolerant to changes, more robust.

The evaluation step

The evaluation step in the cycle is the one more closely related to the actual system the algorithm is trying to optimize. It takes the strings representing the individuals of the population and, from them, creates the actual individuals to be tested. The way the individuals are coded in the strings will depend on what parameters one is trying to optimize and the actual structure of possible solutions (individuals). However, the resulting strings should not be too big or the process will get very slow, but should be of the right size to represent well the characteristics to be optimized. After the actual individuals have been created they have to be tested and scored. These two tasks again are much related to the actual system being optimized. The testing depends on what characteristics should be optimized and the scoring, the production of a single value representing the fitness of an individual, depends on the relative importance of each different characteristic value obtained during testing.

Conclusion

From the above discussion, it can be seen that the GA differs substantially from more traditional search and optimization methods. The four most significant differences are:

• GAs searches a population of points in parallel, not a single point.

• GAs do not require derivative information or other auxiliary knowledge; only the objective function and corresponding fitness levels influence the directions of search.

• GAs use probabilistic transition rules, not deterministic ones.

• GAs work on an encoding of the parameter set rather than the parameter set itself (except in where real-valued individuals are used).

It is important to note that the GA provides a number of potential solutions to a given problem and the choice of final solution is left to the user. In cases where a particular problem does not have one individual solution, for example a family of Pareto-optimal solutions, as is the case in multi objective optimization and scheduling problems, then the GA is potentially useful for identifying these alternative solutions simultaneously. This paper has given a brief introduction to the use of genetic algorithms in optimizing systems. In the limited space it is not possible to discuss all possible ways in which genetic algorithms have been applied to optimization problems.

References

[1] Clark, R.A.F. 1989 Wound repair. Curr. Opin. Cell Biol. 1, 1000–1008. (doi:10.1016/0955-0674(89)90072-0)
[2] Jennings, R. W. & Hunt, T. K. 1992 Overview of postnatal wound healing. In Fetal wound healing (ed. N. S. Adzick & M. T. Longaker), pp. 25–52. New York: Elsevier.
[3] Clark, R. A. F. 1996 Wound repair overview and general considerations. In The molecular and cellular biology of wound repair (ed. R. A. F. Clark), pp. 3–50, 2nd edn. New York: Plenum Press.
[4] Stadelmann WK, Digenis AG, Tobin GR. Physiology and healing dynamics of chronic cutaneous wounds. Am J Surg 1998;176(suppl. 2A):26S-38S.
[5] Diegelmann RF, Evans MC. Wound healing: an overview of acute, fibrotic and delayed healing. Front Biosci 2004; 9: 283-9.
[6] Martin P. Wound healing-aiming for perfect skin regeneration. Science 1997;276:7581.

Measurement Technology and its Application III
10.4028/www.scientific.net/AMM.568-570

Introducing Genetic Algorithm as an Intelligent Optimization Technique
10.4028/www.scientific.net/AMM.568-570.793

DOI References
[5] Diegelmann RF, Evans MC. Wound healing: an overview of acute, fibrotic and delayed healing. Front Biosci 2004; 9: 283-9.
10.2741/1184
[6] Martin P. Wound healing-aiming for perfect skin regeneration. Science 1997; 276: 7581.
10.1126/science.276.5309.75