ABSTRACT

Genetic algorithms have been applied with positive results in many areas including scheduling problems, neural net training, face recognition and other NP-complete problems. Much research trying to improve the performance of genetic algorithms and solve its inherited problems of premature convergence and lack of local search have been conducted. Many variations of genetic algorithms tries to cope with these problems by changing the individual operations that are used in genetic algorithms to improve solutions. Another popular way to improve genetic algorithms is to run them in parallel, some parallel genetic algorithms have performed very well compared to the standard non-parallel genetic algorithm. Parallel genetic algorithms focus their efforts at simulating multiple species and include not only the standard operations for crossover and mutation but also operations for migration between different populations.

The problems of premature convergence and the lack of local search inherited in genetic algorithms are discussed in this paper, as well as some different techniques for solving these problems.

The goal of this paper is to provide and oversight of the standard genetic algorithm and its genetic operations in general and describe parallel genetic algorithms in particular. An example of a specific parallel genetic algorithm called Dual Species Genetic Algorithm is provided. The research investigated in this paper suggests that there are no genetic algorithm that fits all problems but instead different algorithms must be used for different problems.

General Terms

Genetic algorithms

Keywords

Genetic Algorithm, Parallel Genetic Algorithm, Dual Species Genetic Algorithm, Search Algorithm, Path finding, GA, PGA, DSGA

1. INTRODUCTION

Genetic Algorithms(GAs) are a branch of the larger field Evolutionary Computation. GAs have been an interesting field for computer scientists ever since they were first introduced by Holland in 1975[8]. They are appealing because of their robustness and ability to optimize complex solutions containing large number of interacting parts. GAs incorporate methods inspired by the natural evolution to evolve and adapt species to their environment. Roughly speaking, a GA adapts a group of solutions to the problem at hand, in the same way nature adapts species. A GA evolves a population of randomly generated individuals (each representing a solution to the problem) through iterations by applying genetic operations such as crossover, mutation and possibly other methods.

Even though the Standard Genetic Algorithm(SGA) is a powerful optimization tool it does have some problems. The basic problems of the SGA is its lack of local search and its risk of premature convergence[12]. Furthermore SGA require long execution-times when applied to big and complex problems[2].There are many variations of GAs, most considering variations of the standard genetic operations to try to accommodate for the problems inherited in the standard GA. A relatively new branch of GAs called Parallel Genetic Algorithms(PGA) has been developed because of the ease at which GAs can be computed in parallel. In PGAs a number of subpopulations are evolved alongside each other providing the possibility of distributing the execution to multiple processors and increase performance.

This paper provides an overview of the SGA and an explanation of the subfield of PGAs with a specific example called Dual Species Genetic Algorithm(DSGA). The different variations of the standard genetic operations that exists to deal with the problems inherited in genetic algorithms are discussed. Section 2 describes the background of GAs and important subjects related to GAs. Section 3 describes the workings of the SGA, its problems and the genetic operations and their variations. Section 4 discusses PGAs in general and describes the DSGA in particular. Section 5 presents the results of a test that compares a SGA to DSGA. Section 6 concludes the paper with a discussion of the hardships of using GAs and suggests two subjects for future work.

2. RELATED WORK

In the last three decades GAs have been used for optimization and problem solving in many different applications, therefore a lot of research has been conducted and many variations of GA techniques have been developed. Applications can vary between less complex systems for optimization such as simple classification problems to very complex problems such as path optimization[13], gesture recognition[10], stock market predictions[7, 19] and many other subjects. This section describes the research that has been conducted in the field of GAs.
2.1 Genetic Algorithm Operations
Most of the different GA techniques focus on some variation in the standard genetic operations already present to improve the shortcomings of the standard GA. Modification of the selection operation is a popular research subject trying to increase the GAs exploration of the solution search space and to prevent premature convergence[11, 16, 9]. There are also many different techniques for creating offspring by crossover and mutating selected individuals.

When implementing a GA one has to choose the most suitable genetic techniques for the problem at hand from among all the different techniques available. The problem of choosing the crossover and mutation operations are made even more complicated from the fact that there seems to be no best performing operation that fits all problems. Also it has been suggested that different combinations of crossover and mutation techniques perform better than others[18]. Research has been conducted on the subject of choosing the most suitable genetic operators and methods for automatic selection of crossover[21] and mutation[20] has been proposed to simplify the selection.

Attempts to use additional operators "besides the standard genetic operators" to adapt a GA to a specific field has been conducted with positive results. In the GABIL system described by DeJong et al. [3] a GA is used to learn boolean concepts represented by a disjunctive set of rules from propositional logic. An extension to the GABIL system is proposed where two new operators are introduced both motivated by generalization operators used in many symbolic learning methods. The results from this revised system were ambiguous (the extension provided improved performance on some problems and a decreased performance on others). This implies that no single additional operation is appropriate for all kinds of problems.

2.2 Individual Learning
One aspect of evolution not involved in GAs is individual learning, the experience gained during an individuals lifetime is ignored. Evolution that incorporates individual learning is called Lamarckian Evolution(LE), from the late nineteenth century scientist Lamarck. Although LE is not supported by current scientific evidence (although the ideas of LE quite resembles those of epigenetics) studies have shown that LE can sometimes improve the effectiveness of GAs[5, 1]. Baldwin proposes a explanation of LE through an indirect effect on individual learning called the Baldwin Effect. Baldwin suggests that individuals capable of learning to a greater extent than others will adapt faster to the environment and therefore support a stronger population and a more effective evolution.[15] Several studies trying to model the Baldwin Effect has been made. Hinton and Nowlan[6] proposed a model for evolving a population of simple neural networks. Results from their studies indicates that the ability for an individual to learn is a key component in evolution.

3. THE STANDARD GENETIC ALGORITHM
Genetic algorithms try to maximize the fitness of individuals by employing operations inspired by the theory of evolution. Evolution is incorporated into standard GAs by providing interpretations for mutation, offspring production and selection. The SGA performs a randomized beam search by firstly creating a population of individuals and then improving these over iterations by applying genetic operations. In this section a pseudo code representation of a SGA and a discussion of the termination criteria as well as the problems of GAs will be provided. This section also provides explanations of the genetic operations and the standard variations of them that have been developed to improve the performance of GAs.

3.1 The Workings of SGA
The idea of the SGA is to let the best performing or the most "fittest" individuals be selected for breeding, and in this way encouraging good performing individuals. The individuals selected for breeding are then crossed over to create offspring according to a specified crossover rate. Some of the individuals in the offspring are mutated as to bring new material to build solutions from. The resulting offspring populate the next generation of the population, usually by replacing the least fittest individuals. Through iterations of applying these operations, the population evolves towards containing more and more fit individuals. Usually a chromosome representation of the individuals is provided so as to make the crossover operations simple, typical representations are arrays of bits, arrays of other standard types or more complex structures. A index in a array representation is called a gene, respectively, the value of index i in an array representation is called the value of gene . The fitness of a individual is a measure of how good the individual performs when applied to the problem at hand and is calculated by the fitness function. The fitness is used by the selection method to select individuals for crossover and mutation. The representation mapping and fitness function is always problem dependent.

Execution of the SGA:
Population ← initiatePopulation()
repeat
calculateFitness(Population)
Offspring ← crossover(Population)
Mutated ← mutate(Population)
Population.replaceOld(Offspring, Mutated)
until hasConverged(Population) || solutionFound(Population)

In many cases it can be hard to find a good termination criteria for when to stop evolving the population, as many other things, this is dependent on the problem you are trying to solve. Common termination criteria include:

- No progress is seen during the last few iterations
- An individual with good enough fitness is found
- A fixed number of generations have been generated

A nice quality of GAs are that a pretty good estimate can be found in just a few generations, and letting the population evolve further will probably generate better and better solutions, unless the population prematurely converges to a local maximum.
3.2 Problems of the SGA

In his book [15] Mitchell discusses the problem of premature convergence and local search. Premature convergence happens when a group of individuals gains a much higher fitness than the other individuals in the population. This group of individuals will dominate the population and will probably be the only ones selected for breeding. Consequently, when a group dominating it means that a individual with lower fitness will have to mutate in a very precise way as to gain a higher fitness so that it may be able to compete with the dominating group and be selected for breeding. Premature convergence in most cases leads to the discovery of a local maximum because it inhibits diversity in the population and prevents exploration of the whole search space.

The lack of local search is a problem in the SGA because it inhibits the exploration of variations in a specific individuals characteristics. Because of the way a GA produces new solutions, through crossover and mutation “both randomizing methods”, the outcome solution of these methods may perform very bad or very well compared to the initial solution and might have little in common with it, hence the lack of local search. A combination of GAs and hill climbing search has been proposed as to circumvent this problem.

3.3 Genetic Operations

Many variations of GAs and its genetic operations have been developed to try to cope with the problems of premature convergence and the lack of local search mentioned earlier. Following is an explanation of the genetic operations and the most common variations that exist to adapt to certain situations and to cope with the problems of the SGA.

3.3.1 Selection

The selection method plays a great role in population evolution for it is the responsibility of the selection method to select individuals to be crossed over or mutated depending on their fitness. The selection should be made so that the characteristics of individuals with high fitness are explored further while still searching the whole solution search space. Following is a explanation of the most popular selection techniques.

Roulette Wheel Selection. Roulette Wheel Selection(RWS) also known as “Fitness Proportionate Selection” as described by Mitchell[15], assigns a probability of selection to each individual. The probability of individual $i$ being selected is:

$$P_i = \frac{f_i}{\sum_{j=1}^{n} f_j}$$  \hspace{1cm} (1)

where $n$ is the number of individuals in the population and $f_i$ is the fitness of individual $i$. Individuals are then selected stochastically based on their probability of selection, thus, a individual with high fitness will have higher chance of being selected while there is still a chance for the less fit individual to be selected. In this way diversity is maintained within the population.

Rank Selection. Rank Selection acts in a similar way as RWS with one difference, the probability of selection is based on the rank of the individual instead of its fitness. The individual with lowest fitness will have rank 1, the second lowest will have rank 2 etc. the individual with highest fitness will have rank $n$, $n$ being the number of individuals in the population. The probability of selection of individual $i$ ($I_i$) is calculated by:

$$P_i = \frac{r(I_i)}{\sum_{j=1}^{n} j}$$  \hspace{1cm} (2)

where $r(I_i)$ is the rank of individual $i$. Rank Selection gives a higher probability of selection to individuals with low fitness, this provides a slower convergence than RWS but maintains diversity at a higher degree. Versions of Rank Selection where the selection pressure can be adjusted have been developed. One such version of Rank Selection have been developed by Whitley and is discussed in his paper about rank selection and selection pressure[22]. The importance of selection pressure will be discussed in the following subsection.

Tournament Selection. Tournament selection(TS) is easy to code and works well in both parallel and non-parallel GAs[14], furthermore in contrast to RWS, TS provides a easy way of adjusting the selection pressure and can therefore easily be adapted to a specific problem. Adjusting the selection pressure has a direct effect on population convergence, high selection pressure results in higher convergence rates and vice versa. Selecting an appropriate selection pressure has great impact on the performance of the GA, if the selection pressure is too high, only individuals with very high fitness will be allowed to breed, this will result in a fast convergence but will probably result in a convergence to a local maximum, if the selection pressure is too low then the GA will take unnecessary long time to converge.

TS selects individuals by randomly choosing a group of $s$ individuals, then selecting the individual with the highest fitness in this group. Adjusting $s$ here directly adjusts the selection pressure, the bigger the group, the smaller chance of being selected.

3.3.2 Crossover

The crossover operation is responsible for producing two offspring individuals from two parent individuals and is analogous to biological reproduction. Figure 1 explains the most common crossover variations as described in [15]. The correct crossover operation to use depends on the problem. For instance, in a problem where the different parts of the solution depends on each other to produce a individual with high fitness such as in the Traveling Salesman Problem a crossover method which does not upset this dependence is preferable.

3.3.3 Mutation

The mutation operator is responsible for introducing new characteristics into already existing solutions, this is usually done by randomly choosing one value of the solution
and changing this to a randomly chosen value. In a bit array representation of a solution this would involve randomly choosing one index and changing the value at this index. If the value would be a 1 then the mutation operation would change this to a 0. Mutation has great importance in a GA for it is the only operation that introduces new material to build solutions of. Without mutation a population would converge within just a few generations, probably ending up at a local maximum.

4. PARALLEL GENETIC ALGORITHMS

As we have previously mentioned, in Section 1 & 2, there are several improvements of GA. The most popular variations are different kinds of Parallel Genetic Algorithms (PGA)[2]. The different PGAs are optimized for different uses and implementations. Parallelization for the sake of being able to run the algorithm faster with the help of multiple processors. Keeping relatively isolated subpopulations is a parallelization method for acquiring greater diversity. PGAs can be divided into three general classes, Coarse-grained, Fine Grained, and Master-Slave.

4.1 Master Slave Model
Master-slave Model (MSM) is mainly a variation to increase speed, scale and calculation power for GA. Distribution of crossover and mutation operations, and in some cases fitness calculation, can be done to different processors. This allows utilization of the computing power of several processors or distributed computer systems to solve the problem [4]. With the use of the additional resources from distributed computing, we are not as dependent on the development of hardware for single systems, to be able to able to compute more complex problems. On single systems such problems might not be possible to solve within a reasonable time.

MSM assigns a fraction of the population to the available processors for evolutionary operation. Then it can work in two ways. The first one is synchronous, which only has the benefit of faster computing. Once the population is assigned to the processors, synchronous MSM waits for all the processors to complete their operations and return the result before evaluating the new population. The second one is asynchronous, which does not wait for slow processors to return their result. These work a bit different than standard GA, but we will not cover that in this paper.

4.2 Fine-grained Parallel Genetic Algorithm
Fine-grained Parallel Genetic Algorithm (FGA)[2] has only a single population. It has a spacial structure, which restricts the individuals in the population to only interact directly with it’s neighbors. It has a problem that the performance of the algorithm degrades as the size of the population increases.

4.3 Coarse Grained Parallel Genetic Algorithm
Coarse-grained Parallel Genetic Algorithm (CGA) is also known as multiple-deme PGA or “island” PGA[2]. CGA divides the population into subpopulations which can be computed on separate processors. The subpopulation then have an amount of migration between them. Benefits of having smaller populations is that favorable traits spread faster within the population. The drawback is that the rapid rise of fitness stops at a lower fitness value than with a single large population. At first sight this seems as a very simple method to use. The problems arise when implementations of it are analyzed more closely. When setting a low migration rate between the populations, we get the previously mentioned problem with a low quality of the overall solution. But if we should increase the migration level too much, the behavior changes back to that of a single large population.

4.4 Dual Species Genetic Algorithm
Dual Species Genetic Algorithm (DSGA)[12] is a variation of PGA that falls under the class CGA. DSGA solves GAs problem of premature conversion. It is also likely that is gives a result with higher quality than standard CGAs, but we will not discuss this further in this paper. As illustrated in Figure 2, DSGA generates a population, divided into two subpopulations. Operations on the subpopulation 1(p1) and subpopulation 2(p2) are made with different parameters for the crossover and mutation. In p1, an individual is crossed more frequently with similar individuals, than with individuals with fewer similarities. p1 mutate with the general mutation operator. In p2, individuals with fewer similarities are crossed at a higher rate and mutate with a greater mutation operator, than in p1.
similarity of the individuals is measured with equation $S(3)$.

The Crossover operations take individuals as parameters to produce two new individuals. The probability for Crossover 1 and Crossover 2 to be used on a pair of individuals is dependent on the similarity of the two individuals. The probability for Crossover 1 and Crossover 2 to be used on a pair of individuals is to produce two new individuals. The probability for Crossover 1 to be used is given by

$$p_1 = \begin{cases} 1, & S(X, Y) \geq 0.8 \\ 0.2, & S(X, Y) < 0.2 \\ S(X, Y), & \text{else} \end{cases}$$

In contrast with $p_1$, $p_2$ has a negative correlation with its crossover operation. The less the two individuals have in common, the higher is the probability that they will be crossed.

$$p_2 = \begin{cases} 1, & S(X, Y) < 0.2 \\ 0.2, & S(X, Y) > 0.8 \\ 1 - S(X, Y), & \text{else} \end{cases}$$

Using these two crossover methods gives a wider spread search, since in Crossover1 we refines the chromosome which are similar, while Crossover2 widens the search spread by giving the least similar ones a higher chance to evolve.

4.4.2 Mutation
After the crossover, there is a chance that the individuals are mutated. The mutation operation takes one individual as input and delivers one new individual. For $p_1$ the mutation probability is $p_{1_m}$. Since the point with $p_2$ is to be able to avoid, or at least break free, from local peaks, $p_{2_m}$ is made self adaptive. This is done by making it considerably larger than $p_{1_m}$ and dependent of the diversity of the two subpopulations. The definition of $p_{2_m}$ is:

$$p_{2_m} = \begin{cases} 2p_{1_m}, & \text{if } fit_{\text{max}} - fit_{\text{ave}} > fit_{\text{max}} - fit_{\text{ave}} \\ 4p_{1_m}, & \text{else} \end{cases}$$

Where $fit_{\text{max}}$ and $fit_{\text{max}}$ are the maximum fitness of $p_1$ and $p_2$. In turn $fit_{\text{ave}}$ and $fit_{\text{ave}}$ are the average fitness of population $p_1$ and $p_2$.

5. TESTS AND RESULTS
M. Li and J Li have implemented SGA and DSGA on Shekel’s Foxholes equation (9) to test their functionalities[12]. The equation has many local optima, in this case 25 local optima, spread over a wide area, as can be seen in Figure 3. The spread of the many optima makes the equation fitting to test the spread of the results for the search algorithms. Figure 4 and Figure 5 are 2D representations of Equation (9), with the final results from the search algorithms (the blue crosses). $p_{1_m}$ has been set to 0.05.

$$f_1(x_1, x_2) = 500 - \frac{1}{0.002 + \sum_{j=1}^{25} \frac{1}{1 + \sum_{i=1}^{5}(x_i-a_{ij})^2}}$$

$$a_{1j} = 16 \times ((j - 1) \% 5 - 2)$$

$$a_{2j} = 16 \times ((j - 1) \% 5 - 2)$$

$$-60 \leq x_1, x_2 \leq 60$$
fit your problem can be a hard task in itself, even though there are methods to simplify the task, such as adapting crossover and mutation methods, you still have to adjust the rate at which crossover and mutation occurs to fit your problem. Further decisions include population size, genome representation and the number of generations to iterate upon the population. The conclusion from this is that tweaking a GA to a specific problem can be hard and will probably consist of a lot of trials and errors but when all parameters are set correctly you have a well functioning, well optimizing algorithm that will probably produce a optimized solution to your problem or at least a very good estimate.

The most important thing to keep in mind when using GAs is to try to avoid premature convergence. As we have discussed in this paper, using a rank based selection method will help in avoiding this problem. Rank selection will select a good portion of the best individuals but still allow some bad individuals to breed, maintaining a good diversity in the population. The problem of local search can be managed by adjusting the mutation rate. The mutation rate should be set so that individuals do not mutate too often or too strongly. A GA with a mutation rate that is too high will mutate the whole population and render the crossover operation worthless.

DSGA shows that GAs can be improved beyond the capabilities of the SGAs. The numerous variations of genetic operations that are available further suggests that varying combinations of operations will perform differently on different problems. As of today we have yet to discover a genetic algorithm that performs perfect on all problems, the variations in search space topology is one major reason for this. Before solving a problem we have no information about the search space and can only come up with guesses or estimations on how the best solution will look like.

Further research on individual learning but in the context of epigenetics needs to be conducted. Also further work on DSGA for running on distributed systems is needed.

7. REFERENCES


