Clustering Based Hierarchical Genetic Algorithm for Complex Fitness Landscapes

Rahul Kala
MTech
Department of Information Technology,
Indian Institute of Information Technology and Management Gwalior, Gwalior, MP, INDIA
rahulkalaiiitm@yahoo.co.in

Anupam Shukla
Associate Professor
Department of Information Technology,
Indian Institute of Information Technology and Management Gwalior, Gwalior, MP, INDIA
dranupamshukla@gmail.com

Ritu Tiwari
Assistant Professor
Department of Information Technology,
Indian Institute of Information Technology and Management Gwalior, Gwalior, MP, INDIA
rt_twr@gmail.com

*Corresponding Author
Room No 101, BH-1, ABV-IIITM Gwalior, Gwalior, MP, INDIA
Ph.: +91-9993746487

Citation: R. Kala, A. Shukla, R. Tiwari (2010) Clustering Based Hierarchical Genetic Algorithm for Complex Fitness Landscapes, *International Journal of Intelligent Systems Technologies and Applications* 9(2), 185-205.

Final Version Available At:
http://www.inderscience.com/info/inarticle.php?artid=34320

Abstract

We propose the use of a hierarchical Genetic Algorithm (GA) for optimization in complex landscapes. While the slave GA tries to find local optima in the restricted fitness landscape of low complexity, the master GA tries to identify interesting regions in the entire landscape. The slave GA is a conventional GA with high convergence. The master GA is more exploratory in nature. This GA clusters the fitness landscape with each cluster in control of a slave GA. The number of clusters decreases with time to get global characteristics. The novelty of the suggested approach lies in the tradeoff between the search for global optima and convergence to local optima that can be controlled between the two GAs. We tested the

algorithm and observed that the approach exceeds conventional GA as well as Particle Swarm Optimization in complex landscapes.

Keywords: Hierarchal Genetic Algorithms; Evolutionary Algorithms; Fitness Landscape; Dimensionality; Optimization; Genetic Algorithms; Machine Learning; Soft Computing; Complexity; Random Algorithms; Swarm Intelligence.

1. Introduction

Genetic Algorithms are valuable tools for the purpose of optimization (Mitchell 1998). The optimization powers of these algorithms have found a variety of use in many spheres and disciples. The continuous research in each of these fields has resulted in rising complexities that the GA is expected to solve. The conventional GA with simple fitness landscape is easily able to search the global optima, escaping all local optima (Manderick 1993). But this does not certainly hold well when the fitness landscape is too complex in nature with many hills and valleys over multiple dimensions. As the fitness function grows in complexity, the GA finds very difficult to locate the global optima (Yao 1993; Kala et al 2009a, 2009b). As a result the entire algorithm may many times behave random in nature.

The search for global optima in a complex landscape environment is a difficult problem. Since the problem is of very high dimensionality, one may be able to explore only a reasonably small part of the entire fitness landscape. The task is to explore the best areas that may aid in finding the global optima. The time of exploration is always small as per the requirements of the fitness landscape. Hence in a limited span of time we have to give the best possible output to the problem being optimized.

GAs use many operators called as Genetic Operators (Deb 1999, Deb and Agrawal 1999) to generate individuals from a lower generation to a higher generation. These operators may be controlled with the help of some system parameters. Fixing the correct values to these parameters is of a lot of importance for the optimal working of any GA. A major limitation of the GA is that these parameters are human controlled. As a result the users have to try again and again to find the optimal value of the parameters looking at the results and other indicators. These parameters are always prone to be sub-optimal that do not result in the best solutions. A lot of efforts exist to make the GA parameter less to avoid the problem (Lobo, Lima and Michalewics 2007). However we would always have to set some parameters of the system as per the No Free Lunch (NFL) (Wolpert and Mcready 1997) theorems.

The crossover and mutation are two commonly used parameters for control of the GA as per the fitness landscape and scenario of the GA (Syswerda 1989; Jong and Spears 1991, 1992; Eberhart and Shi 2006; Shukla, Kala and Tiwari 2010). The crossover encourages the individuals of the population to converge around the best points as they search for the optima. On the other hand mutation tries to encourage the exploratory nature of the individuals by making them explore at places around the present coverage. In this manner crossover contracts the search space while mutation expands the same. These parameters are set by the user looking at the convergence rate of the GA. A premature convergence may mean a decrease in crossover and increase in mutation rate and vice versa. In this manner the GA is able to search for optima in the fitness landscape.

The inability of a single GA to solve the problem of optimization for complex landscapes results in a hierarchical application of GA. In this paper we propose a two level hierarchy of the GAs or the master and slave GA. The slave GA is the low level GA

that works over a limited size of the fitness landscape and tries to find optima there. There are many slave GAs that simultaneously search for the optima in their respective domains. The master GA does the task of coordination, control and parameter setting of the slave GAs. Clustering is used as a mechanism of division of the fitness landscape. There is a lot of information change between the two levels of GA for devising a proper search strategy.

This paper is organized as follows. In section 2 we present the related works. Section 3 gives the general algorithmic framework. This includes the slave and master GA along with the passage of information between the GAs. Section 4 would present the various parameters of the proposed GA and the manner in which they need to be set. In section 5 we discuss some of the simulation results. We give the conclusions in section 6.

2. Related Work

Various attempts have been made previously for optimization in complex landscapes for complex problems. The Hierarchical Genetic Algorithm (HGA) proposed by Jong *et al* (2004) is a novel approach. Here the authors used the notion of modularity and hierarchy and developed an algorithm for optimization. In this approach an initial set of modules is first evolved. These are diverse set of modules that give a high performance and may be used into the solution being evolved. The genetic individual of hierarchy consists of a set of modules. Mutation and crossover were proposed to act upon this structure. Crossover carries out the task of exchange of modules among the individuals. The mutation does the task of alteration of the modules in order to add new characteristics onto it. An application of a hierarchical individual representation and adapted crossover and mutation operators may be seen in the recent works of Kumar *et al* (2008). Here the authors applied the Hierarchical Genetic Algorithm for the problem of multilevel allocation redundancy which is a well studied problem. Wang *et al* (2002) used another hierarchical representation for the problem of robotic path planning using HGA. Here the problem was to find out the optimal robotic path which was modeled by the authors as an optimization problem and solved by using HGA.

One of the highly complex problems is the problem of evolution of the neural network. Here we are supposed to evolve the architecture as well as the weights of the neural network by evolutionary algorithms. The search space is naturally very complex with too many dimensions. Yen and Lu (2000) used hierarchical representation for evolving a neural network. In their approach a three level hierarchy was used. The first level consisted of the information about the neural layers. The second level consisted of information about the neurons. In these bits marked the activation or deactivation of a particular neuron. The third level contained the parameters like weights and biases. These were coded into the individual as real values. The authors used one-point crossover and Gaussian mutation for all the levels. The final evaluation of the neural network was done by its performance over the training data with a performance penalty for larger networks.

The Island Model Parallel Genetic Algorithm (IMGA) is another novel concept proposed by Gordon et al (1992). Here the entire population is divided into sub-populations. Each sub-population has its own evolution procedure and operates in isolation to each other. This enables the generation of good individuals which possess the general characteristics of that sub-population. This further helps a lot in diversity preservation. The migration of individual in-between sub-populations is carried out using migration strategies. In migration individuals are transferred from one subpopulation to the other subpopulations. This brings in new characteristics to the subpopulation, which were developed and optimized by the other subpopulations. These are further developed

and enable convergence towards the global minima. Hence the entire algorithm operates in the cycles of isolation and migration. Using a similar structure Antonio (2006) suggested a HGA with age based structure. Here he also used controlled mutation in order to better the local search of the individual. The different hierarchies proposed by Antonio further make it possible for the use of different crossover operators running on different crossover techniques. A comparison of different crossover operators related to Elitist hybrid crossover with genetic improvement, Elitist parameterized uniform crossover and Age parameterized uniform crossover is provided in (António 2008). Sefrioui and Jacques (2000) also used different models at various hierarchies to carry out optimization.

The Hierarchical Fair Competition based Genetic Algorithms (HFCGA) are a class of Genetic Algorithms that solve the problem of premature convergence in HGA (Hu *et al* 2002, 2005). In this mechanism of evolution, the entire population pool is organized into hierarchies in terms of the fitness values. Every hierarchy has an admission threshold below which it does not entertain individuals and an expert threshold. An individual possessing a fitness value above this threshold is exported to another hierarchy. Every hierarchy has its own evolution that enables its optimal evolution. The various parameters are different for the different level of hierarchies. Oh *et al* (2009) presented an application of such algorithm in the problem of design of fuzzy cascade controllers where this algorithm was used for optimization of the fuzzy system.

Garai and Chaudhuri (2007) proposed a similar algorithm called the Distributed Hierarchical Genetic Algorithm (DHGA). Here they divided the entire sub space into smaller sub spaces. This algorithm operated in stages. At every stage there was a redefinition of the search space that took place. As per the performance the mutation operator was also adapted with every stage. The distribution was affected by migration that followed a coarser to finer rule where the resolution of the search space was increased as the algorithm executed. Lim *et al* (2007) further proposed the entire work of HGA onto a Grid Computing framework. The algorithm was called as the Grid Computing Hierarchical Parallel Genetic Algorithm (GE-HPGA).

A similar concept of hierarchies is implemented in the domain of Evolutionary Strategies (ES). Rudolph (1997) used a nested evolutionary strategy where the step size of the lower ES was modified after a few iterations. The entire approach consisted of an ES that carried out the optimizations as per the parameters that it possessed. The performance of the ES depends upon the choice of step size used. Further the step size needs to be adaptive and change as per the current scenario. This change was carried out by the use of another heuristic strategy that set the step sizes after few iterations of the optimization of the ES. Lohmann (1992) used nested ES for discrete and continuous variables. Here the inner generations were for the optimization of continuous variables. After a few iterations, the discrete variables were changed. This model specially highlighted the manner to work with optimization problems involving both discrete as well as the continuous variables. Apart from step size, the other major parameter that affects the hierarchical implementation of the ES is the isolation period. This is the total time which the ES is given in isolation without any change of parameters. Arnold and Castellarin (2009) used hierarchical ES to adapt the isolation period. Here the isolation period was increased or decreased based on the success of the earlier runs.

3. Algorithm

The entire algorithm operates in a hierarchical master-slave mode. The master does the task of coordination and parameter setting of the slave. The slave does the task of finding

the optima in the region allotted. In this manner the algorithm proceeds and explores the complex fitness space. The general master-slave framework of the algorithm is given in figure 1.

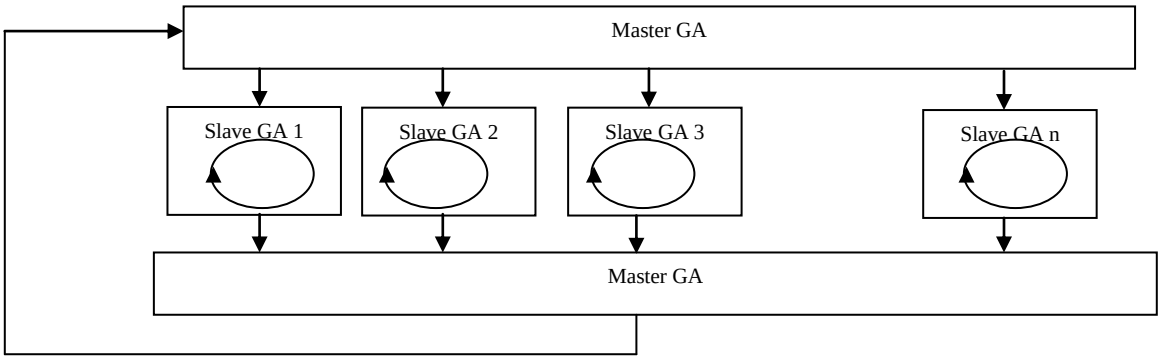

Figure 1: Hierarchical nature of the algorithm in terms of master and slave Genetic Algorithms

Each of the slaves represents a distinct segment of the fitness space. This segmentation of the fitness function is done by the application of clustering. Each slave represents a cluster. The number of slaves is the total number of clusters applied by the master GA.

The number of clusters or slaves is kept large at the first little iterations. As the algorithm proceeds with generations, we keep reducing the number of clusters. This may be interpreted as we start with an intention to explore and find out the local optima at specific parts of the fitness space. Later based on its findings, we try to search for the global optima. At the last few generations, the number of clusters is unity. Now the local GA searches for the entire search space and hence tries to find the global optima. This is the local to global search strategy of the algorithm.

3.1 Slave Genetic Algorithm

The slave GA is a conventional GA. The purpose of the slave GA is to search for the optima in the restricted search space. In other words the slave GA searches for the local optima. The basic purpose of the slave GA is to fully explore in-depth the given fitness landscape. This gives a clear idea of the nature and fitness of that region of the space with the investment of computation. Another important fact is that for many real-life situations the algorithm may need a good result early. In such scenarios it is better to go deep and converge into some local optima, rather than keep searching for global optima. The results and other findings of this GA are of a high importance to the master GA that tries to control the slave GA in a manner that the best solution is found within the time restrictions. The algorithm for the slave GA is shown in figure 2.

We know that the input given to the slave GA is almost always a simple fitness landscape with limited search space. Hence we adapt the various genetic operators and parameters to match this need of the slave GA. In the next sub-sections, we discuss the importance of the various parameters and operators. This forms a basis of the entire working of the algorithm.

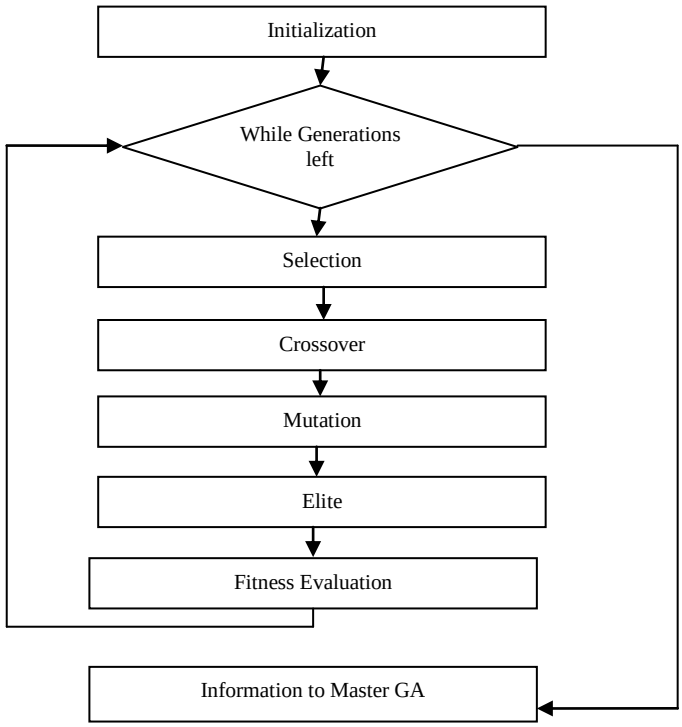

Figure 2: Slave Genetic Algorithm

3.1.1 Number of Individuals

The number of individuals in any GA denotes its exploratory nature. It also denotes the total computational power. A larger number of individuals denote a more exploration of the search space but with the computational cost. More number of individuals ultimately limits the generations and makes the algorithm more random. The more complex fitness landscapes usually employ more number of individuals. As the search space increases we need more and more number of individuals. Hence in this approach, we start with a highly limited number of individuals. As the generations proceed, the number of individuals has a general increases.

3.1.2 Number of Generations or Isolation Period

The number of generations in a local GA may also be called as the isolation period. This is the time when all local GAs operate in isolated of each other. The number of generations in GA again depends upon the complexity of the fitness landscape. A simple landscape might require a few generations as compared to a complex one. The larger the number of generations, the more is the exploration performed by the GA which means a larger computational cost as well. Accordingly, the number of generations is kept low at start and they have a general increase as we proceed with the algorithm.

3.1.3 Mutation Rate

The mutation rate in a GA decides its exploratory nature. It doesn't let the GA converge at some point, rather keeps the GA busy exploring newer regions in the fitness space. The larger the value of mutation, the more wide the GA tries to explore in search of the global

optima. At the initial iterations, the mutation rate is kept low as we are primarily interested in searching for local optima. This also restricts the GA from entering fitness space that is in possession with some other local GA. As the algorithm reaches higher generations, the mutation rate is increased. This is again because of the fact that much of the good parts of the fitness space are already explored at various lower generation runs. As a result we need a high mutation rate at the end to explore new areas.

3.1.4 Crossover Rate

The crossover rate decides the convergence of the GA. It tries to move all the individuals towards the best areas and ultimately converge at some point denoting optima. It behaves opposite to mutation. Here we keep the crossover high at start. As we keep progressing with the generations, the crossover is reduced in general.

3.1.5 Elite Count

The elite are the individuals passed directly from one generation to the other. They enable preservation of the best individuals from being eliminated or deformed during the genetic process. The elite count is usually kept constant to a low value. In this algorithm as well we keep the elite count constant to a small value that does not change with generations.

3.2 Master Genetic Algorithm

While the slave GA works in a restricted fitness space, the task of the global GA is to define and distribute this fitness space among the slave GAs. The other task of the master GA is the parameter control and the coordination of the slave GAs. Every slave GA is a cluster. We treat this cluster or slave GA analogous to the genetic individual while working with the master GA. Here each cluster itself has a set of individuals controlled by the slave GA. We discuss the various aspects of the master GA and the various genetic operators of this GA one by one. The algorithm for the master GA is shown in figure 3.

3.2.1 Individual Representation

Every slave GA or cluster contains a lot of information that makes up the individual of the master GA. The individual or cluster z represents the following information.

- Position in the fitness space or the fitness landscape. This is taken as the position of the best individual searched by the local GA (I_z^{rel})
- Relative Number of iterations ($Iter_z^{rel}$)
- Relative Mutation Rate (Mut_z^{rel})
- Relative Crossover Rate ($Cros_z^{rel}$)
- Relative Number of Individuals (Ind_z^{rel})
- Relative Convergence (C_z^{rel})
- Relative Diversity (D_z^{rel})

All relative parameters may be positive, negative or zero denoting the deviation of these values from the average number values as per the current generation. The other parameters denote their normal numeric values. All these are discussed in detail along with role in section 3.3. Here convergence and diversity are informational parameters that denote the status of the individual. They are not used while application of genetic operators or for evaluation of the fitness functions.

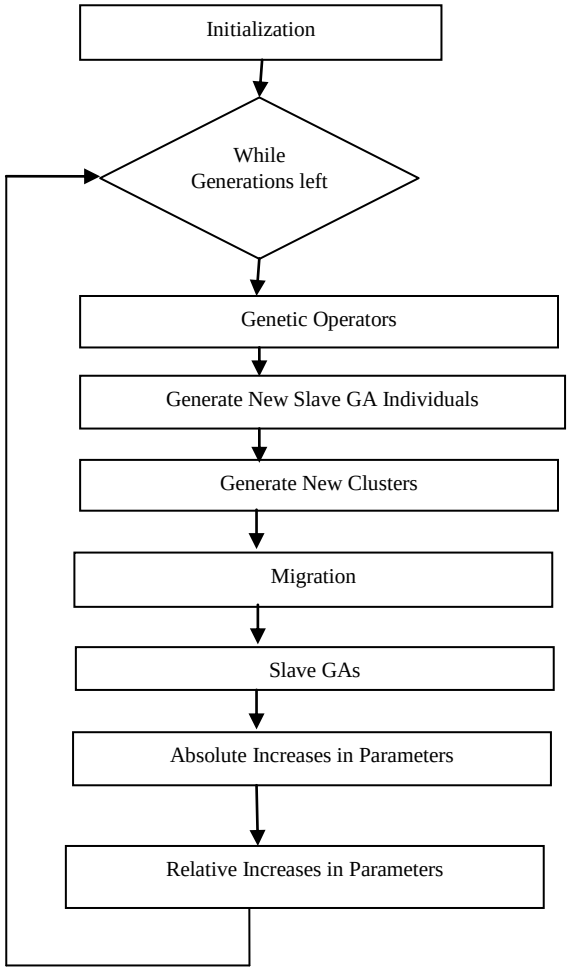

Figure 3: Master Genetic Algorithm

3.2.2 Number of individuals

The number of individuals or cluster in the master GA is variable. We start with a very high number of clusters as we need some initial idea of the landscape. As the generations increase, we keep decreasing the number of individuals or clusters. This induces more and more of global traits in the final solution. Another way to interpret this is that we mainly use the results of the smaller generation to make up the larger generation. The search starts with more exploration at different areas. Once the areas are identified, we try to look forward for possibility of global optima out of the generated results of lower generations.

3.2.3 Selection

The selection needs to ensure that two very distinct individuals do not get selected. In other words we need to ensure that two clusters do not undergo crossover that lie far apart in the fitness space. This is to preserve diversity at the master level. Killing of diversity as master GA would badly adverse the algorithm as its results may be magnified at the slave

GA operation. We hence use fitness scaling with stochastic selection to select the first parent for crossover. The second parent is selected randomly from the nearest *k* individuals.

3.2.4 Crossover

The crossover is applied between two individuals or clusters to generate new individuals or clusters that belong to the higher generation. We simply take the arithmetic mean of the cluster center positions along with the other parameters represented by the individual. In this manner the crossover is performed.

3.2.5 Mutation

In this operator we modify only the position of some cluster and make it move by some amount in the fitness space. The amount of movement depends upon the mutation rate used in the master GA.

3.2.6 Elite

The elite pass the best fitness cluster from one generation to the other. Its value is kept as unity which is constant for the entire algorithmic run. Here fitness is defined as the fitness of individual that resides at the cluster center or the fitness of the best individual of the corresponding local GA.

3.2.7 New Individuals and Clusters

This genetic operator adds new clusters and individuals into the population pool. The relative parameters are all initialized to zero. Since the fitness landscape is complex, it is always important to search for completely new areas in the fitness landscape in search of global optima. It might not be wise to restrict only to the earlier explored areas and to try to locate the optima based on the same areas. The addition of new individuals hopes the exploration of good fitness areas.

3.2.8 Migration

As per the notations of the standard hierarchical genetic algorithm, migration refers to the interchange of population among the local GAs. Migration in the proposed algorithm takes place by the operation of re-clustering. The previously studied genetic operators have already computed the locations of the new cluster centers. We further have a population of individuals ready obtained from the best few individuals of all the local GAs population. We add some more random individuals in this pool of individuals. Then re-clustering of the available individuals takes place. Every cluster is given individuals that lie closest to it. In this manner the execution of the local GA starts. It is not ascertained that every cluster will get individuals as denoted by their relative number of individuals. The relative number of individuals only denotes the number of individuals it would contribute to the total population pool before re-clustering starts.

3.3 Information Passage between GAs

The proper working of the algorithm requires a proper coordination being maintained between the various local GAs. This is done by the master GA. In order the master GA to perform the task, it is important to have passage of information from the master GA to the slave GA and vice versa. The master GA initializes and executes the slave GA. As a result it is able to pass information to the slave GA. This information includes the initial population along with the genetic parameters represented by the cluster.

The slave GA in turn passes the information of its best individuals as per its relative fitness. It also passes the information of its total diversity and convergence.

Diversity (D_z) of the z^{th} cluster is defined as the average distance between any two individuals. It may be given by the equation (1).

$$D_z = \frac{\sum_{i=1}^{N} \sum_{j=1, j \neq i}^{N} |I_i - I_j|}{N*(N-1)} \quad (1)$$

Here I_i denotes the i^{th} individual in the cluster, N is the total number of individuals in the cluster

Convergence (C_z) of the z^{th} cluster is the total change in fitness witnessed by the GA with respect to its best individual in the cluster. This is given by equation (2).

$$C_z = \text{fit}(\text{best}(P_{ini})) - \text{fit}(\text{best}(P_{final})) \quad (2)$$

The relative diversity (D_z^{rel}) and relative convergence (C_z^{rel}) for any cluster z are measured as their relative measures to all clusters in the master population. They are given by equation (3) and (4).

$$D_z^{rel} = \frac{D_z - \bar{D}}{\max(D_z) - \min(D_z)} \quad (3)$$

$$C_z^{rel} = \frac{C_z - \bar{C}}{\max(C_z) - \min(C_z)} \quad (4)$$

To calculate the other parameters i.e. Relative Number of iterations ($Iter_z^{rel}$), Relative Mutation Rate (Mut_z^{rel}), Relative Crossover Rate ($Cros_z^{rel}$) and Relative Number of Individuals (Ind_z^{rel}) we use the concept of diversity and convergence. Every cluster z has a diversity and convergence that affect its relative parameters in a multiplicative manner. An increase in diversity means an increase in the relative number of iterations, decrease in relative mutation rate, increase in relative crossover rate and decrease in relative number of individuals. Similarly an increase in convergence means an increase in the relative number of iterations, decrease in relative mutation rate, increase in relative crossover rate and decrease in relative number of individuals in the next generation.

The equation for Relative Crossover Rate ($Cros_z^{rel}$) is given by equation (5). Similarly the other relative parameters may be computed.

$$Cros_z^{rel}(g) = (D_z^{rel} + C_z^{rel}) * \frac{Cros^{abs}(1) * maxcros}{G} \quad (5)$$

Here $Cros^{abs}(1)$ is the initial crossover rate. The *maxcros* is the maximum percentage increase in the crossover rate with generations. In case any parameter (e.g crossover rate itself) decreases with generation, *Cros* is measured from the final generation *G* rather than initial generation *1*. The various parameters increase or decrease along with the number of generations (*g*) of the master GA. The number of individuals of master GA (*nclus*) or the number of clusters decreases uniformly with time along with the rise in generations. This is given by equation (6)

$$nclus(g+1) = nclus(g) - \frac{nclus(1)}{G} \quad (6)$$

Here *nclus(1)* is the maximum number of clusters that can appear in the master GA. These are the number of clusters at the first generations. G is the maximum number of generations in the GA.

The other parameters i.e Number of iterations ($Iter_z$), Mutation Rate (Mut_z), Crossover Rate ($Cros_z$) and Number of Individuals (Ind_z) change due to two factors. These are absolute and relative. The absolute is a general increase or decrease in the parameter value along with the master GA. The relative is the fine tuning done by the relative measures of these parameters to optimize the overall performance of the algorithm. For the number of iterations, this may be given by equation (7)

$$Iters_z(g) = Iters^{abs}(g) + Iters_z^{rel}(g) \quad (7)$$

Here $Iters^{abs}(g)$ is the absolute or a general increase in the total number of iterations along with time. This is given by equation (7).

$$Iters^{abs}(g+1) = Iters^{abs}(g) + \frac{Iters^{abs}(1) * maxinc}{G} \quad (8)$$

Here *maxinc* is the maximum percentage increase in the number of iterations as discussed earlier.

$Itersz^{rel}(g)$ is given by equation (8).

4 Various Genetic Parameters

The parameters that we have added and would like to study are the initial number of clusters, initial number of individuals, total generations, initial mutation, initial crossover, initial number of iterations, and percentage increase in mutation, crossover and iterations.

The initial number of clusters denotes the localized nature of the algorithm. The more the number of clusters, the more is the approach of the algorithm to return local optima as the final answer assuming a limited constant computation. It closely resembles the number of individuals in the conventional GA where more individuals add randomness. For most real life applications, which require a very fast results we may keep the number of clusters high. For applications that are not much time restrictive, the clusters may be kept low for more time to be spent on the global exploratory nature of the algorithm. Similar trend may be observed in total number of individuals used by local GA. The more the number of individuals, the more is the randomness or convergence to local optima which may be desirable in real life scenarios.

The initial mutation and initial crossover are kept sufficiently low and high respectively for easy search for the local optima by the local GA. These generalize the notion of crossover and mutation over the entire GA. A high initial mutation results in a lot of exploration outside the allotted search space in the initial iterations as well. This may be an attempt not to converge to local optima, but to search the global optima. As a result we take longer to generate good solutions, but the chances of being near the global optima are high. In contrast a high initial crossover rate means a desire to converge straight into the local optima which might be the case when good solutions are needed early in the algorithm run. The initial number of iterations further follows the tends of

initial number of individuals where more iterations mean a better exploration in a local GA and the solutions returned being more locally searched in nature. This is unlike when the search space in search was large.

The percent increases of the mutation, crossover and iterations denote the dynamic changes in the local GA or the local search algorithm. It denotes the speed with which the algorithm intends to work over the global picture. This further adapts the local GAs as per their local landscapes. Every GA is hence enabled to perform better in the scenario it is presently in when compared to the entire algorithm. It may be recalled that since the landscape is complex, we increase the randomness in the larger generations and the percentage increases denotes the rate of increase of randomness. In this algorithm trying to find a global solution is quite analogous to making the algorithm more random.

Very high percent increases would make the algorithm very random at a very early stage. This would not be useful if a good result needs to be generated at early stages. Similarly small percentage increases may not be able to find global optima. They are likely to be struck at some local optima.

5 Results

The algorithm was implemented and tested using JAVA as a platform. For the purpose of testing we studied the various functions used in literature. We finally selected a set of 11 test functions. Some of them were taken from the work of Garai and Chaudhuri (2007) and the others were taken from the work of Shi *et al* (2005). These functions along with the ranges and optima points are given in table 1. Each of these was executed and analyzed separately. The value of the parameters was fixed same for all these functions. All simulations were made on a 4 GB RAM, 3.0 3.0 GHz Core 2 Duo processor. The parameters used for the simulations were 10 as the initial number of clusters, I (problem specific) individuals, G generations (problem specific) of the master GA. The initial mutation rate was fixed as 0.03 and crossover rate as 0.8. The initial number of iterations of the slave GA was 100. The percentage increases were 1 for mutation and generations and 0.5 for crossover. All simulations took 2 to 4 seconds running time as per the set parameters except the last function which took approximately 5, 7 and 15 seconds for the three runs with different dimensions.

We used Particle Swarm Optimization (PSO) and standard Genetic Algorithm (GA) to compare the proposed algorithms. All simulations were carried out 20 times. The parameters of these algorithms were kept following the best practices at the same time keeping the runtime of all the algorithms similar. The mutation rate was fixed to 0.03 and crossover was fixed to 0.7. Individuals and generation could vary from 600 to 800 and 1500 to 7500 respectively for various problems. This meant a very large number of populations as generations due to low overheads of these algorithms. The means and standard deviation of the results obtained from the simulation results along with number of individual and generations for proposed algorithm is given in Table 2. The table also lists the points of optima in the fitness space as recorded with the best run.

Table 2 showcases the performances of the proposed algorithms compared to standard GA and PSO. The 11 functions given to the algorithm to optimize represent a range of simple to complex functions.

Table 1: The benchmark objective functions used for the testing of the algorithm

S. No.	Function	Dimension	Range	Minima
F1	$F = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)][30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	2	$-2 \leq x_1, x_2 \leq 2$	3
F2	$F = (x_1 - x_2)^2 + \left(\frac{x_1 + x_2 - 10}{3}\right)^2$	2	$-10 \leq x_1, x_2 \leq 10$	0
F3	$F = 100(x_1^2 - x_2)^2 + (1 - x_1)^2$	2	$-2 \leq x_1, x_2 \leq 2$	0
F4	$F = \sum_{i=}^{3} 100(x_i^2 - x_{i+1})^2 + (1 - x_i)^2$	4	$-2 \leq x_i \leq 2$	0
F5	$F = \sum_{i=}^{6} x_i^2$	6	$-1 \leq x_i \leq 1$	0
F6	$F = \frac{-1}{0.1 + (\sum_{i=1}^{10} \frac{x_i^2}{4000} - \prod_{i=1}^{10} (\cos\left(\frac{x_i}{\sqrt{i}}\right) - 1))}$	10	$-1 \leq x_i \leq 1$	-10
F7	$F = -\sum_{i=}^{3} x_i^2$	3	$-10 \leq x_i \leq 10$	-300
F8	$F = -\sum_{i=1}^{5} x_i e^{1-x_i} + (1 - x_i)e^{x_i}$	5	$-1 \leq x_i \leq 1$	-8.30
F9	$F = -\sum_{j=1}^{4} \frac{1}{1 + \sum_{i=1}^{4} (x_i - 1)^6}$	4	$-10 \leq x_i \leq 10$	-4.
F10	$F = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	$-5 \leq x_i \leq 5$	-1.031628
F11	$F = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	5, 8, 12	$-10 \leq x_i \leq 10$	0.0

In order to facilitate a judicious comparison between the algorithms, all the algorithms were executed for the same time durations. The time was 2 to 4 seconds for all functions except the last one for which the time was 5, 7 and 15 seconds for the three runs with different dimensions. The parameters of all the algorithms were fixed such that the optimization takes place to the largest possible extent in the allocated time duration. This is especially important to decide the convergence of the algorithm, as the various individuals need to converge after the desired time. Too early or late convergence would lead to sub-optimal results. The proposed algorithm, because of complex hierarchical operators would execute lesser number of fitness evaluations as compared to the other algorithms within the same time. However, as discussed, it is capable of giving better results in complex fitness landscape. Results show a better performance of the proposed

algorithm for the same execution time which advocates the overall improvement by this algorithm in optimization.

Table 2: Comparative analysis of working of the proposed algorithm with GA and PSO on the objective functions

S. No.	Optimal Value	Proposed Algorithm			GA		PSO	
		I, G	Mean	St Deviation	Mean	St Deviation	Mean	St Deviation
F1	3	100, 500	3.000038	0.000089	142.0500*	301.4589	3.000000	3.79e-8
F2	0	100, 500	0.000000	4.571e-9	0.000000	1.63e-23	0.000000	4.63e-9
F3	0	100, 500	0.000000	9.99e-5	0.000000	2.24e-20	0.000000	7.35e-9
F4	0	200, 1000	0.047769	0.04429	0.000000	1.40e-8	0.000158	0.000108
F5	0	100, 500	0.007559	0.014469	0.000000	1.83e-10	0.000000	3.15e-7
F6	-10	100, 500	-9.997000	0.002323	-10.00000	8.991e-9	-10.0000	2.03e-7
F7	-300	100, 500	-298.7401	0.671932	-269.462	20.45611	-299.917	0.03725
F8	-8.30	100, 500	-8.243602	6.7094e-6	-8.24361	3.70e-10	-8.24360	1.029e-6
F9	-4	100, 500	-3.997382	0.008396	-4.00000	3.15e-5	-4.00000	5.12e-9
F10	-1.031628	100, 500	-1.031628	2.5538e-4	-0.11236#	1.02411	-1.03163	7.81e-9
F11(i) (ii) (iii)	0 0 0	200, 1000 350, 3000 400, 4000	0.028758 0.025957 0.157634	0.048754 0.033986 0.369671	-0.95720$ 6.50802 31.3468	2.94720 4.50280 24.6291	0.01465 0.072406 1.051142	0.00300 0.01316 1.49311

* At a few runs the algorithm was trapped in the minima corresponding to value 839 and 30

At a few runs the algorithm was trapped in the minima corresponding to value -0.21546 and 2.10425

$ At a few runs the algorithm was trapped in the minima corresponding to value 9.41

We separately discuss the results in heads of simple and complex functions. The functions in table 1 numbered F1 to F9 are relatively simple functions with limited dimensionalities or simpler fitness landscape. These functions were given to the various algorithms to have a clearer comparison on the general functions between the three algorithms. Looking at the results we can clearly see that all the 3 algorithms were easily able to optimize these functions. The optimal values were significant in regard to the execution time which was intentionally kept small to judge the real-time performance in simpler functions. The convergence or optimality of the GA and PSO seem to be much higher for these functions. GA and PSO lead to more optimal values as compared to the proposed algorithms. Also the low values of standard deviation illustrate the stableness of these algorithms as compared to the proposed algorithm. But still the performance of the proposed algorithm may be stated significant. Recall that our aim was not the perfect

global minima, but a good performance in complex landscapes. In other words, we did not intend to converge deep inside the global minima, but rather appreciated the exploratory nature as well which acts as an overhead to keep the final value sub-optimal. Function F1 somehow did not show a good performance in the use of GA where the GA sometimes got trapped in local minima corresponding to 83 and 839 at some of the runs. In most cases however, the GA could converge to global minima in F1.

The scenario keeps changing as we start increasing the complexity. This is when the proposed algorithm starts depicting useful characteristics as while the other algorithms start facing problems. This was when we gave the inputs F10 and F11 with the dimensionalities of 5, 8 and 12. The proposed algorithm was able to generate optimal solutions in all the scenarios. The optimality may be stated satisfactory looking at the complexity of the landscape. However, the GA completely failed to find optimal solutions in these inputs. For F10 the GA could do reasonably well when it got optimal values at most of the runs, with non-optimal solutions at few runs. Similar was the case with the run of F11 with a dimensionality of 5. The optimal value was fetched most of the times with a few non-optimal solutions. However, the optimality was very poor for the other runs of F11. The PSO behaved decent with F10 and gave good solution at par with the proposed algorithm. However the scenario with F11 was different. The optimality was fine for the first run with dimensionality of 5, but kept deteriorating. PSO lost to the proposed algorithm in higher dimensions of 5 and 12. In dimensionality of 12, it mostly failed to give the correct solutions.

Based on the simulation with F11, it may be extrapolated that all three algorithms would pose problems of high complexity as we keep increasing the dimensionality. However, GA and PSO in a dimensionality of 12 lag behind the proposed algorithm. Continuing the simulation at higher dimensions would make the difference even larger. Many of the real life applications like evolution of ANN (Yao 1993) is a highly dimensional problem that takes a lot of time. At such high complexities, it may easily be generalized that the proposed algorithm would serve better.

In order to visually analyze the complexity of the function we draw the function F11 in 2-dimensions given in figure 4. It may again be seen that this extended over multiple dimensions pose a very complex problem because of which many algorithms fail.

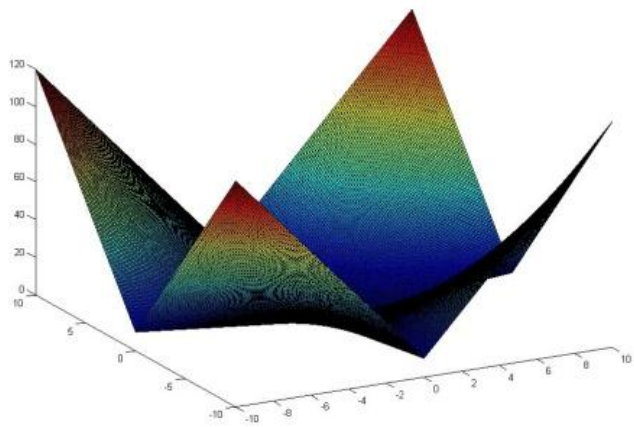

Figure 4: Fitness landscape (in 2 dimensions) of F11 objective function

The next thing of interest to study is the convergence. In order to see the algorithmic behavior, we see the performance of the algorithm with regard to the best individual. Since a large portion of the population is always randomly generated, we disregard the performance of average and worst individual which may carry a lot of random characteristics that is not of our interest. The performance may be studied separately for the master and the slave GA. Figure 5(a) shows the graph between the best individual in population v/s generations for the master GA when simulated for F11 with dimensionality of 12. It may be seen that the graph has a discrete behavior. Every plot of the master GA corresponds to a number of cycles of the slave GA that optimize the objective function. Hence the master GA is discrete in little nature. The slave GA on the contrary behaves differently. This is a typical GA with high crossover rate and low mutation rate to allow fast convergence. The graph between the fitness and generations for the first run of the first slave GA is given in figure 5(b).

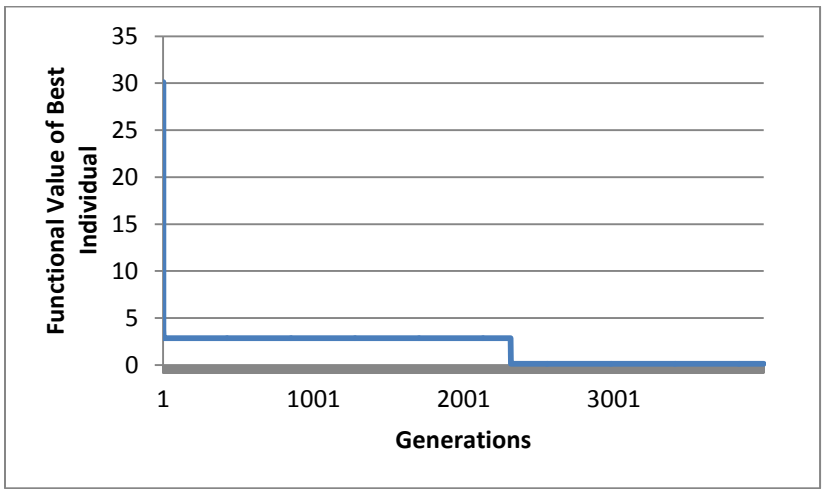

Figure 5(a): Convergences in master GA

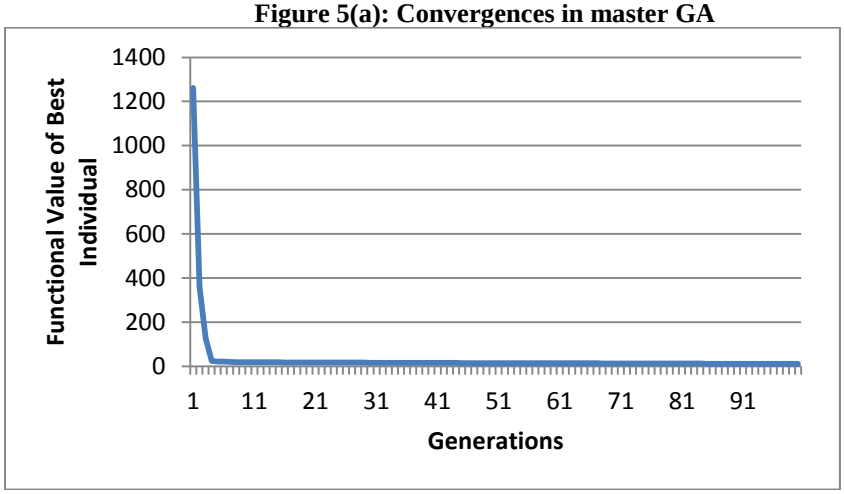

Figure 5(b): Convergences in slave GA

6 Conclusions

In this paper we proposed a hierarchical implementation of the GA. The entire GA was implemented in two hierarchies of master GA and slave GA. The role of the slave GA was simply to converge to some local minima. This was done to give optimal results in low times. The master GA was supposed to carry forward the task of coordination and parameter setting of the local GA.

The testing of the algorithm was done using the benchmark functions available in literature (Garai and Chaudhuri 2007; Shi et al 2005). 11 such functions were identified. These functions ranged in their complexity and dimensionality. We tested the performance of the algorithm in comparison to the conventional GA and PSO. The results revealed that the conventional GA and PSO were able to give a good performance in most of the simple objective functions. They penetrated deep into the optimization of the fitness value and returned a solution that was highly close to the global optima. The proposed algorithm also gave a decent performance but could not penetrate that deep in search of the global minima. The scenario started changing when we used functions of high complexity. Results denote that as the complexity increased, the GA and PSO started facing problems. Since most of the real life problems are much complex to the tested problem, it may be inferred that the proposed algorithm wit better scalability factors would perform much better in real-time situations or elsewhere.

The proposed algorithm gave a decent performance to complex functions. The real scalability test of the proposed algorithm lies in its use in most real life complex applications. These applications present a fitness landscape that is much more complex than the objective functions used. The work of testing and comparing the proposed algorithm in these domains may be done in future. The algorithm further makes an attempt to use the global GA as a means of parameter setting of the local GA by using the metrics of local GA as guides and relating and absolute changes for general improvement. This is a much complex relationship between parameters that require a much formal modeling and study. The improvement in this segment may have a deep impact on the algorithmic performance. Another important aspect of the algorithm is its tradeoff between the local and global characteristics. While we present many parameters to adapt the algorithm to any of these characteristics, a formal study in various specific scenarios and runtimes may be conducted in future.

References

[1] António, C. A. Conceição (2006), 'A hierarchical genetic algorithm with age structure for multimodal optimal design of hybrid composites', *Structural and Multidisciplinary Optimization*, Vol 31, pp 280-294

[2] António, C. A. Conceição (2006), 'A study on synergy of multiple crossover operators in a hierarchical genetic algorithm applied to structural optimisation', *Structural and Multidisciplinary Optimization*, Vol 38, pp 117-135

[3] Arnold, Dirk V, and Castellarin, Anthony, S (2009), 'A Novel Approach to Adaptive Isolation in Evolution Strategies', In *Proceedings of Genetic and Evolutionary Computation Conference*, GECCO-2009, New York, pp. 491-498

[4] Deb, Kalyanmoy (1999), 'An Introduction to Genetic Algorithms', In *Sadhana*, Vol 24, No 4, pp 205–230.

[5] Deb, Kalyanmoy, and Agrawal, Samir (1999), 'Understanding interactions among genetic algorithm parameters', In *Foundations of Genetic Algorithms 5*, Morgan Kaufmann Publications, pp 265-286

[6] Eberhart, Russell C., and Shi, Yuhui (2006), 'Comparison between genetic algorithms and particle swarm optimization', In *Evolutionary Programming VII*, Vol 1447, pp 611-616

[7] Fogarty, Terence C. (1989) , 'Varying the Probability of Mutation in the Genetic Algorithm', In Proceedings of the 3rd International Conference on Genetic Algorithms, Morgan Kaufmann Publishers , pp 104 - 109

[8] Garai, Gautam, and Chaudhury, BB (2007), 'A distributed hierarchical genetic algorithm for efficient optimization and pattern matching', *Pattern Recognition*, Vol 40, pp 212 – 228

[9] Gordon, VS, Whitley, D, and Böhn A (1992), 'Dataflow parallelism in genetic algorithms', In: MännerR,ManderickB(eds) *Parallel problem solving from nature 2*. Elsevier, Amsterdam, pp 533–542

[10] Hu, J., Goodman, E., Seo, K., and Pei, M. (2002), 'Adaptive Hierarchical Fair Competition (AHFC) Model for parallel evolutionary algorithms' In *Proceedings of Genetic and Evolutionary Computation Conference*, GECCO-2002, New York, pp. 772–779

[11] Hu, J., Goodman, E., Seo, K., Fan, Z., and Rosenberg, R. (2005), 'The Hierarchical Fair Competition (HFC) framework for continuing evolutionary algorithms', *Evolutionary Computation*, Vol. 13, no. 2, The MIT Press, pp. 241–277.

[12] Jong, Edwin D De, Thierens, Dirk, and Watson, Richard A (2004), 'Hierarchial Genetic Algorithms', In *Springer Lecture Notes in Computer Science*, Vol 3242, pp 232-241

[13] Jong, Kenneth A. De , and Spears, William M. (1991), 'An analysis of the interacting roles of population size and crossover in genetic algorithms', In *Springer Lecture Notes in Computer Science*, Vol 496, pp 38-47

[14] Jong, Kenneth A. De , and Spears, William M. (1992), 'A formal analysis of the role of multi-point crossover in genetic algorithms', *Annals of Mathematics and Artificial Intelligence*, Vol 5, No 1, pp 1-26

[15] Kala, Rahul, Shukla, Anupam, and Tiwari, Ritu (2009), Fusion of Evolutionary Algorithms and Multi-Neuron Heuristic Search for Robotic Path Planning, Proceedings of the *IEEE 2009 World Congress on Nature & Biologically Inspired Computing (NABIC '09)*, Coimbatote, India

[16] Kala, Rahul, Shukla, Anupam, Tiwari, Ritu, Roongta, Sourabh, and Janghel, RR (2009), Mobile Robot Navigation Control in Moving Obstacle Environment using Genetic Algorithm, Artificial Neural Networks and A* Algorithm, Proceedings of the *IEEE World Congress on Computer Science and Information Engineering*, Los Angeles/Anaheim, USA, pp 705-713,

[17] Kumar, Ranjan, Izui, Izui, Kazuhiro, Masataka, Yoshimura, and Nishiwaki, Shinji (2008), 'Multilevel Dependency Allocation Optimization Using Hierarchical Genetic Algorithms', *IEEE Transactions on Reliability*, Vol 57, No 4

[18] Lim, Dudy, Ong, Yew-Soon, Jin, Yaochu, Sendhoff, Bernhard, and Lee, Bu-Sung (2007), 'Efficient Hierarchical Parallel Genetic Algorithms Using Grid Computing', *Future Generation Computer Systems*, Vol 23, No 4, pp 658-670

[19] Lobo, Fernando C, Lima, and Claudio F, and Michalewics, Zbigniew (Eds.)(2007), *Parameter Setting in Evolutionary Algorithms*, Springer

[20] Lohmann, R (1992), 'Structure evolution and incomplete induction Parallel Problem Solving from Nature', Proceedings of the *2nd International Conference on Parallel Problem Solving from Nature*, Brussels, Amsterdam, Elsevier, pp 175–85

[21] Manderick, B, Weger, M de, Spiessens, P (1991), 'The genetic algorithm and the structure of the fitness landscape', In *Proceedings of the Fourth International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers
[22] Mitchell, Melanie (1998), *An Introduction to Genetic Algorithms*, MIT Press
[23] Muhlenbein, Heinz (1992), 'How genetic algorithms really work I. Mutation and Hill Climbing', Parallel Problem Solving from Nature,
[24] Oh, Sung-Kwun, Jung, Seung-Hyun, and Pedrycz, Witold (2009), 'Design of optimized fuzzy cascade controllers by means of Hierarchical Fair Competition-based Genetic Algorithms', *Expert Systems with Applications*, Vol 36, pp 11641-11651
[25] Rechenberg, I. (1973), *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipiender biologischen Evolution*, Frommann-Holzboog, Stuttgart.
[26] Rudolph, Gunter (1997), Evolutionary Strategis, In Evolutionary Algorithms and Their Standard Instances, *Handbook on Evolutionary Computation*, IOP Publishing Ltd and Oxford University Press
[27] Schwefel, H.P. (1995), *Evolution and Optimum Seeking*, John Wiley.
[28] Sefrioui, Mourad, and Periaux, Jacques (2000), 'A Hierarchical Genetic Algorithm Using Multiple Models for Optimization', In *Springer Lecture Notes in Computer Science*, Vol 1917, pp 879-888
[29] Shi, XH, Liang, YC, Lee, HP, Lu, C, Wang, LM (2005), 'An improved GA and a novel PSO-GA-based hybrid algorithm', *Information Processing Letters*, Vol 93, pp 255–261
[30] Shukla, Anupam, Tiwari, Ritu, and Kala, Rahul (2010), *Real Life Applications of Soft Computing*, CRC Press
[31] Srinivas, M. Patnaik, L.M. (1994) , 'Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms', IEEE Transaction on Systems, Man and Cybernetics, Vol 24, Issue 4, pp 656-667
[32] Syswerda, Gilbert (1989), 'Uniform Crossover in Genetic Algorithms', In *Proceedings of the 3rd International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, pp 2 - 9
[33] Wang, Chunmiao, Soh, Y C, Wang, Han, and Wang, Hui (2002), 'A Hierarchical Genetic Algorithm for Path Planning in a Static Environment with Obstacles', In *Proceedings of the 2002 Congress on Evolutionary Computation CEC'02*, Vol 1, pp 500-505
[34] Wolpert, David H, and Macready, William G (1997), 'No Free Lunch Theorems for Optimization', *IEEE Transaction on Evolutionary Computation*, Vol 1, No 1, pp 67-82
[35]Yao, Xin (1993), 'Evolutionary Artificial Neural Networks', In *International Journal of Neural Systems*, Vol 4, No 3, pp 203-222
[36] Yen, Gary C, and Lu Haiming (2000), 'Hierarchal Genetic Algorithm Based Neural Network Design', In *2000 IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks*, pp 168-175

Author Biographies

Mr. Rahul Kala

Mr. Rahul Kala is a student of Integrated Post Graduate Course (BTech + MTech in IT) in Indian Institute of Information Technology and Management Gwalior. His fields of research are hybrid system design, robotic planning, design of algorithms, artificial intelligence

and soft computing. He has published 10 papers in various national and international journals/conferences including Journal of Mobile Robotics and Intelligent Systems (JAMRIS), Springer LNCS, International Journal of Computer Science and Network Security (IJCSNS), IEEExplore conferences, ANNIE, etc.He secured All India 8th position in Graduates Aptitude Test in Engineeging-2008 with a percentile of 99.84. He is the winner of Lord of the Code Scholarship Contest organized by KReSIT, IIT Bombay and Red Hat. He secured 7th position in ACM-International Collegiate Programming Contest Kanpur Regional.

Dr. Anupam Shukla

Dr. Anupam Shukla is an Associate Professor in the IT Department of the Indian Institute of Information Technology and Management Gwalior. He has 20 years of teaching experience. His research interest includes Speech processing, Artificial Intelligence, Soft Computing, Biometrics and Bioinformatics. He has published around 80 papers in various national and international journals/conferences. He is Editor and reviewer in various journals. He received Young Scientist Award from Madhya Pradesh Government and Gold Medal from Jadavpur University.

Dr. Ritu Tiwari

Dr. Ritu Tiwari is an Assistant Professor in the IT Department of Indian Institute of Information Technology and Management Gwalior. Her field of research includes Biometrics, Artificial Neural Networks, Speech Signal Processing, Robotics and Soft Computing. She has published around 30 papers in various national and international journals/conferences. She has received Young Scientist Award from Chhattisgarh Council of Science & Technology and also received Gold Medal in her post graduation.