Hierarchical Evolutionary Strategy for Complex Fitness Landscapes

Rahul Kala*, Ritu Tiwari, Anupam Shukla,
Soft Computing and Expert System Laboratory,
Indian Institute of Information Technology and Management Gwalior

*Corresponding author
Rahul Kala
Soft Computing and Expert System Laboratory,
Indian Institute of Information Technology and Management Gwalior
Morena Link Road,
Gwalior, Madhya Pradesh-474010
India

Web: http://rkala.99k.org/
Email: rahulkalaiiitm@yahoo.co.in
Ph: +91-9993746487

Citation: R. Kala, A. Shukla, R. Tiwari (2010) Hierarchical Evolutionary Strategy for Complex Fitness Landscapes, *Journal of Information Science and Technology* 7(2), 36-57.

Final Version Available At: http://pascal.iseg.utl.pt/ojs/index.php/jist/article/view/110

ABSTRACT

Evolutionary Strategies (ES) are effective forms of Evolutionary Algorithms that enable solving optimization problems. Effective use of ES algorithm has been made in numerous fields. Major optimization problems of today possess a very complex fitness landscape with numerous modalities. The optimization in these complex landscapes is much more difficult as it is possible only to explore a relatively small section of the entire landscape. Also the fitness function behaves in a very sensitive manner with a large amount of change for small changes in the parameter values. We hence propose a hierarchical ES to optimally explore the fitness landscape and return the optima. The inner or the slave ES is controlled by a controlling algorithm or the master. The master has a number of slave ES, each trying to find a solution at some different part of the complex high dimensional fitness landscape. Each ES tries to find the optimal point in its local surroundings. Hence the variable step size is initially kept low. As the iterations of the master increase, we keep reducing the number of ESs and increase the step size to give it a global nature. This is the local to global nature search performed by the algorithm. Since the fitness landscape is complex, the master mutates the locations of the ESs and adds new ESs (deleting the non-optimal ones) as iterations or generations proceed. 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 hierarchies. Experimental analysis shows that the proposed algorithm gives a decent performance in simple optimization problems, but a better performance as we increase the complexity, when compared with the conventional Genetic Algorithm, Particle Swarm Optimization and conventional ES.

Keywords: Hierarchal Evolutionary Strategies; Evolutionary Strategies; Evolutionary Algorithms; Fitness Landscape; Dimensionality; Optimization

1. INTRODUCTION

Evolutionary Algorithms (EA) take an analogy from the natural evolutionary process for solving the optimization problems. Here one generation of individuals contributes for the making of the higher generation of individuals. The fitness improves as we iterate to the higher generations (Mitchell 1998). EAs mainly incorporate Genetic Algorithms, Genetic Programming and Evolutionary Strategies (ES). The ES is a novel method of problem solving with limited individuals (Rechenberg 1973; Schwefel, 1995). Here recombination takes place among the parents to generate the children. Mutation is applied to all the generated children. The best individuals (may or may not comprise of parents) go to the next generation. The ES may hence be abbreviated by ($\lambda/\rho,^{+} \mu$) which means that population consists of λ parents. μ children are made in the next generation out of these parents. Each child is generated by recombination of ρ number of parents. Then we take combined λ parents and μ children to generate new λ individuals of the next generation. If '+' is not used in the notation, then only the best of μ children make the generation. In this manner the algorithm keeps iterating along with generations and generates newer individuals.

ES have been applied to numerous problems that range from simple functional optimization to design optimization and robotic path planning. These algorithms give optimal results that usually converge well into the global minima (Manderick 1993). However this may not be necessarily true for highly complex landscapes. The complex landscapes usually span across multiple dimensions and assume very complex structures that are difficult for any conventional evolutionary technique or search strategy to get the global minima (Yao 1993; Kala et al. 2009a, 2009b). Complex landscapes may be explored to a reasonably small extent due to the finiteness of time. The highly sensitive nature displayed by these algorithms makes it difficult to get the global minima. As a result the algorithms may usually converge to a local minima or any point far from the vicinity of the global minima. The conventional algorithms may behave more random due to the complexity.

EAs are conventional and widely used means of problem solving for the optimization problems. They use many 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 has of a lot of importance for the optimal working of the algorithm. A major limitation of these algorithms 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 effort exists to make these algorithms free of parameter 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) theorems (Wolpert and Mcready 1997).

Crossover and mutation rate are two commonly used parameters for control of the EA as per the fitness landscape and scenario of the EA (Syswerda 1989; Jong and Spears 1991, 1992; Eberhart and Shi 2006; Shukla, Tiwari and Kala 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 EA. A premature convergence may mean a decrease in crossover and increase in mutation rate and vice versa. In this manner the EA is able to search for optima in the fitness landscape.

The ES further make extensive use of parameters that contribute towards the effective working of these algorithms. λ, μ and ρ are the fundamental parameters that are usually kept constant in the program run. The other important parameters would involve the step size (σ), the total number of generations for which the algorithm is executed and the initial individuals of the algorithm. All these have an impact on the effective working of the algorithm in some or the other way that is analogous to the EAs.

The failure or inability of a single EA to solve the optimization problem in case of complex fitness landscape results in the use of multiple algorithms or hybridization of algorithms. Here we wish to combine algorithms in such a manner that the advantage of one overcomes the disadvantage of the latter. Hybridization usually results in effective performance boost especially in problems where all simple methods fail. The increasing complexity has resulted in a lot of use of hybrid algorithms over various spheres. The hybridization in Evolutionary Algorithms is no exception which sees the fusion of evolutionary techniques for effective problem solving.

This work reports the control of ES by a master control technique. We hence implement the algorithm in two levels of hierarchies, the master and the slave. The slave is a simple ES. The slave ES needs control and coordination. For this purpose we implement a master algorithm that works over the slave ES. The master has multiple instances of different slave ESs working at various positions of the fitness landscape. All these ESs work in isolation of each other and search for some local minima in their near vicinity. The master tries to do parameter control of the slave ESs by fixing their number of generations, position of individuals and mutation strengths. The whole algorithm is built over the local to global approach. This means that at the first few time steps we try to find the various local minima at various locations of the fitness landscape. As the algorithm proceeds, we intend to search or the global minima. Hence the numbers of ES instances reduce with time to enable a single ES take charge of the entire landscape based on the observations of the earlier ES instances and runs. Also the mutation strength and number of generations are increased by the master for more exploration and search for global minima.

This paper is organized as follows. In section 2 we present the related works. The CMA Evolutionary strategy used in this paper is introduced in section 3. Section 4 gives the general algorithmic framework. This includes the slave and master. Section 5 would present the various parameters of the ES. In section 6 we discuss some of the simulation results. We give the conclusions in section 7.

2. RELATED WORK

Various concepts of hierarchies are implemented in Evolutionary Strategies (ES). Rudolph (1997) used a nested evolutionary strategy where the step size of the lower ES was modified after a few iterations. Lohmann (1992) used nested ES for discrete and continuous variable problems. Here the inner generations were for the optimization of continuous variables. After a few iterations, the discrete variables were changed. 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.

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. The individuals in a hierarchy correspond to many modules. Mutation and crossover were proposed to act upon this structure. 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. Wang et al. (2002) used another hierarchical representation for the problem of robotic path planning using HGA. Yen and Lu (2000) used hierarchical representation for evolving a neural network.

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. The sub-populations operate in isolation to each other. The migration of individual in-between sub-populations is carried out using migration strategies. Using a similar structure Antonio (2006) suggested a HGA with age based structure. Here he also used controlled mutation in order to improve 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). Here we associate every individual with a class and then every class undergoes a fair competition amongst its members. The migration is carried out between the various hierarchies of populations based on these scores. 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. 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).

3. CMA EVOLUTIONARY STRATEGIES

CMA-ES or the Covariance Matrix Adaptation Evolutionary Strategy (Hansen 2008) is a classical implementation of ES that uses a Covariance Matrix for the computation of the mutation strengths of the individuals of the population.

The basic concept behind the algorithm is the creation of a higher of generation individuals by sampling a multivariate normal distribution. The basic equation for this generation from the generation i to the generation i+1 is given by equation (1).

$$x_k^{i+1} \approx m^i + \sigma^i N(0, C^i) \tag{1}$$

Here x^{i+1}_k denotes the k^{th} individual in a population of λ individuals for the generation i+1.

m^i is the mean value of the search distribution

σ^i is the overall standard deviation or the step size. This ultimately leads to the mutation strength in the population.

$N(0,C^i)$ is a multivariate normal distribution with zero mean and covariance matrix C^i

The new mean of the generation i+1 from a generation i is the weighted mean of the population that has already been generated for the new generation using equation (1). This is given by equation (2).

$$m^{i+1} = \sum_{i=1}^{\mu} w_j x_{j:\lambda}^{i+1} \tag{2}$$

Here w_j are the weight coefficients for recombination. $w_{j=1, 2, 3 \ldots \mu} = 1/\mu$ for equal weightage strategy.

$x_{j:\lambda}^{i+1}$ is the j^{th} best individual out of x_1^{j+1}, x_2^{j+1} , x_3^{j+1}..., x_λ^{j+1} where the individuals are sorted by their fitness.

The weights w_i have been normalized and hence obey the equation (3). Further w_j are always sorted in descending order to enable higher contribution of fitter individuals.

$$\sum_{i=1}^{\mu} w_i = 1 \tag{3}$$

Assigning of different weights w_j for the various individuals is referred as the selection mechanism of the ES as the different individuals give different contributions to the final individual in accordance with their assigned weights. Variance effective selection mass (μ_{eff}) is given by equation (4). This parameter always lies between 1 and μ and may normally be assigned a value of λ/4.

$$\mu_{eff} = \left(\sum_{i=1}^{n} w_i^2 \right)^{-1} \tag{4}$$

The next task is to adapt the covariance matrix C^i. This is done as a combination of two procedures called as rank 1 update and rank μ update. The resultant equation may be given by equation (5).

$$c^{i+1} = (1 - C_{\text{cov}})C^g + \frac{c_{\text{cov}}}{\mu_{\text{cov}}} p_c^{i+1} p_c^{i+1^{\tau}} + c_{\text{cov}}(1 - \frac{1}{\mu_{\text{cov}}}) X \sum_{j=1}^{\mu} w_j y_{j:\lambda}^{i+1} (y_{j:\lambda}^{i+1})^{\tau} \tag{5}$$

Here

$\mu_{cov} \geq 1$. Choosing $\mu_{cov} = \mu_{eff}$ is most appropriate

$c_{cov} \approx \min(\mu_{cov}, \mu_{eff}, n^2)/n^2$

$y_{j:\lambda}^{i+1} = (x_{j:\lambda}^{i+1} - m^i)/\sigma^2$

The next step to be performed is the adaptation of the step length or σ^{i+1}. This value denotes the step length and must be optimal; in accordance with the length of the path. The adaptation of this factor is carried out with the comparison between the length of the path with that of the expected length of path under random conditions. The resultant equation is given by (6).

$$\sigma^{i+1} = \sigma^i \exp\left(\frac{c_\sigma}{d_\sigma} \left(\frac{\left\| p_\sigma^{i+1} \right\|}{E\left\| N(0,1) \right\|} - 1 \right) \right) \tag{6}$$

Here E||N(0,I)|| is the expected path length on a random distribution

d≈1 is the damping parameter that scales the change magnitude of ln σ^i.

p_σ^{i+1} is the conjugation evolution path given by equation (7)

$$p_\sigma^{i+1} = (1 - c_\sigma) p_\sigma^i + \sqrt{c_\sigma (2 - c_\sigma) \mu_{eff}}\, C^{i^{-1/2}} \frac{m^{i+1} - m^i}{\sigma^i}$$

(7)

A detailed description of the CMA-ES is available in (Hansen 2008).

4. ALGORITHM

The algorithm works at two hierarchies of master and slave. The master algorithm is supposed to carry forward the task of coordination and control over the slave algorithm which is a standard CMA-ES. In this manner the algorithm chiefly incorporates the concept of controlled ES over a complex fitness landscape. The slave ES is given a particularly small portion of the complex fitness landscape to solve. As the algorithm proceeds, the effective area increases which denotes a more global nature of the algorithm. The general master-slave framework of the algorithm is given in figure 1.

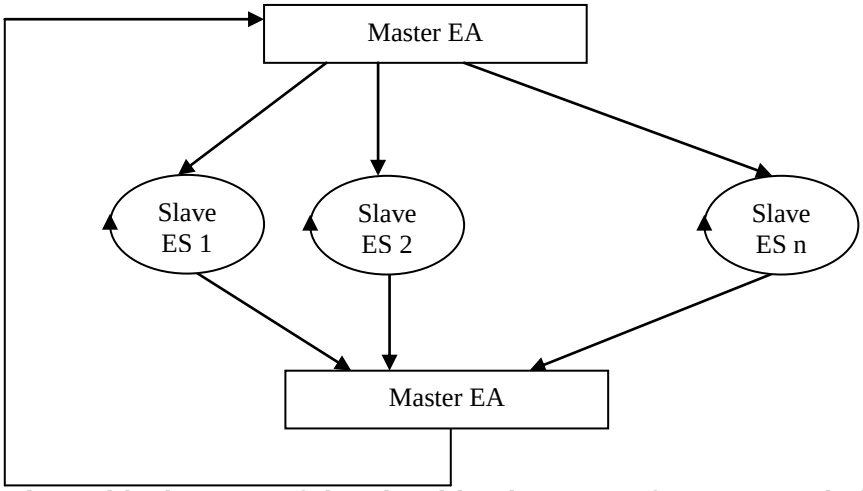

Figure 1: Hierarchical nature of the algorithm in terms of master Evolutionary Algorithm and slave Evolutionary Strategy

The number of slave ESs instances is kept large at the first little iterations. As the algorithm proceeds with generations, we keep reducing the number of ESs instances. 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 ES instances is unity. Now the ES 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.

4.1 Slave CMA-ES

The slave is simply an instance of CMA-ES as discussed in section 3. The purpose of the slave ES is to search for the optima in the restricted search space. In other words the slave ES searches for the local optima. The basic purpose of the slave ES 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 ES are of a high importance to the master controller that tries to control the slave ES in a manner that the best solution is found within the time restrictions. The algorithm for the slave ES is shown in figure 2.

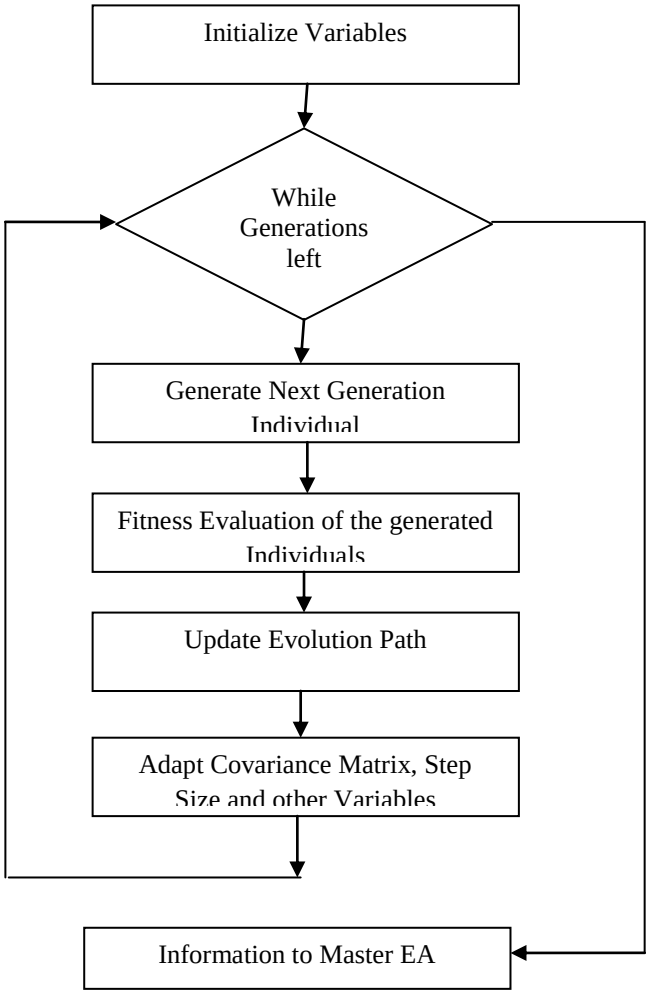

Figure 2: Slave Evolutionary Strategy

We know that the input given to the slave ES is almost always a simple fitness landscape with limited search space. Hence we adapt the various operators and parameters to match this need of the slave ES. The number of generations in a local ES may also be called as the isolation period. This is the time when all local ESs operate in isolation from each other. The number of generations in ES 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 ES 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.

The step size (σ) in a CMA-ES decides its exploratory nature that ultimately decides the basic region in which the ES would work. The larger the value of this parameter means that the algorithm is allowed to make large mutations and hence spread the individuals across a comparatively larger area. As the generations increase, the movement further increases and covers the entire fitness landscape. This is contradictory to the ES which has a small value of this parameter, where the steps restrict individual movements to a very small area. At the initial iterations the step size is kept low, as we are primarily interested in searching for local optima. This also restricts the ES from entering fitness space that is in possession with some other local ES. As the algorithm reaches higher generations, the step size 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.

4.2 Master Controller

While the slave CMA-ES works in a restricted fitness space, the task of the master controller is to define and distribute this fitness space among the slave ESs. The other task of the master ES is the parameter control and the coordination of the slave ESs. The algorithm for the master controller is shown in figure 3. The entire algorithm behaves like a mini evolutionary algorithm with its own operators that are used to control the slave ESs. Each slave ES instance acts like an individual of this evolutionary algorithm. The algorithm at every generation first uses a selection strategy to select the best few ES instances. It then adds new instances of ES to this pool. All the ES instances undergo a mutation operation where the mean positions of these ES individuals are mutated and moved in the fitness landscape by an amount depending upon the fitness landscape. The various parameters of the ES increase or decrease with time with the parameter control logic. We discuss the various concepts in the following subsections.

4.2.1 Representation: Each individual p of the master evolutionary algorithm is an instance of the slave ES. The slave ES is represented by its mean individual X_p. Further each ES has parameters that affect its performance. Most of the CMA-ES parameters may be kept constant or computed from the best practices (Hansen 2008), the parameters that need to be fixed are the stopping criterion and the step size (σ). Here we use the number of iterations (G) or generations as the stopping criterion. Both these parameters are constant for all instances of ESs.

4.2.2 Number of Individuals: Each individual of the master EA or the number of ES instances are variable in number and change along with time. This is done to maintain a local to global strategy of the algorithm. The number of individuals (n^i) at generation i decrease along with time following equation (8).

$$n^i = (n^0 - 1)\left(1 - \frac{i}{G}\right) + 1 \qquad (8)$$

Here G is the total number of generations for master EA

n^0 is the maximum number of individuals

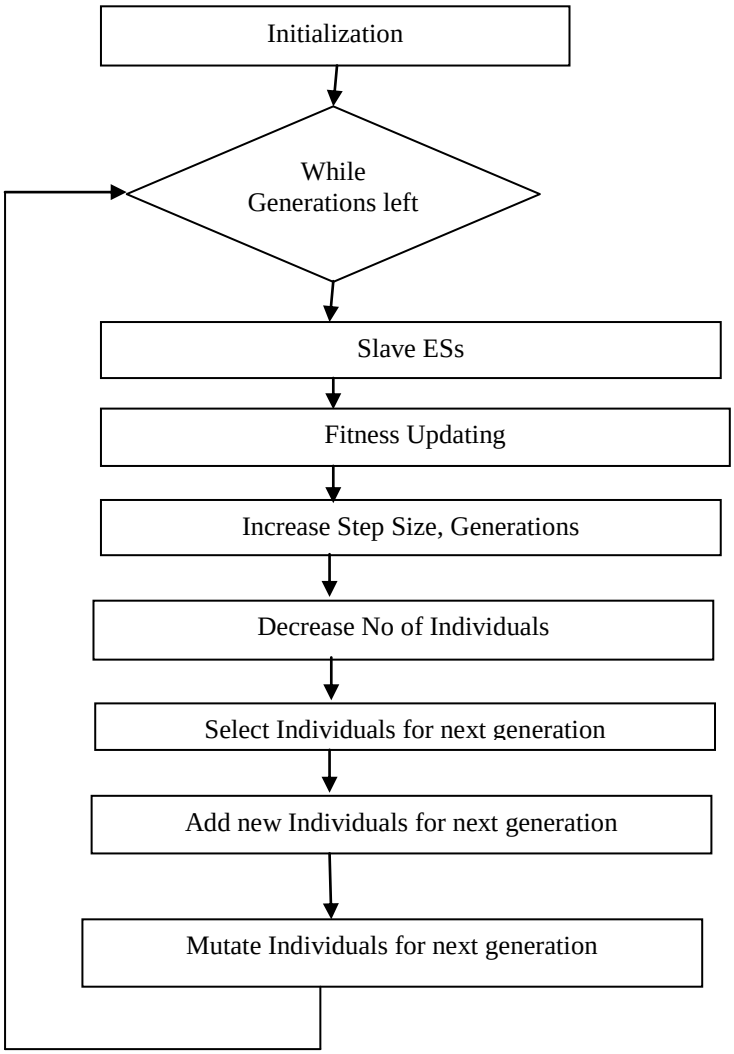

Figure 3: Master Evolutionary Algorithm

4.2.3 Selection: Only the best c% individuals required by the next generation go from the previous generation. Here c is the relative contribution of previous generation to the next generation. The fitness of the mean solution of the ES is used as a measure of the goodness of the individual of the master EA. However we need to pay attention towards diversity as well to avoid individuals with same or close position in the fitness landscape to work in the next iteration. This is important as for every iteration of the master, there is a complete cycle of slave ES that gives it enough time to search for the optima in the close vicinity depending upon the step size and number of generations fixed. Similar ESs may merge if allowed to pass from one generation to the other of the master EA. Here we pass an individual to the next generation only when it is located at some threshold distance (ή) in the fitness landscape from the other individuals.

4.2.4 New Individuals: This is an operator that adds new ES instances or individuals into the population pool of the master EA. In all (100-c)% of individuals required for the next generation are added by this operator. They are located at completely random positions in the fitness landscape.

Since the landscape is complex, there might be an incentive to invest computation at random places as well from time to time. This is implemented by this operator.

4.2.4 Mutation: All individuals of the master EA undergo mutation which changes their position by a small amount in the fitness landscape depending upon the mutation rate (m) which is fixed. Any individual X_p after mutation changes its position given by equation (9).

$$Xp' = Xp'(1+rand*m) \quad (9)$$

Here *rand* is a random number on the range -1 to 1.

4.2.5 Evolutionary Strategy Parameters: The two ES parameters Step Size (σ^i) and Generations (G^i) increase with time that enable a more exploratory nature needed to explore the global optima along with time. These are given by equation (10) and (11)

$$\sigma^{i+1} = \sigma^i + \frac{\sigma_{inc}}{G} \quad (10)$$

$$G^{i+1} = G^i + \frac{G_{inc}}{G} \quad (11)$$

Here σ_{inc} is the maximum increase in step size from initial step size.

G_{inc} is the maximum increase in number of generations from initial number of generations.

5 VARIOUS GENETIC PARAMETERS

In section 4 we introduced various parameters that make up the new Evolutionary Algorithm implemented by the master controller. The parameters that we have added and would like to study are the initial number of clusters, initial number of ES instances or individuals, total generations of master EA, mutation rate, contribution of previous generation to next generation (c), increase in step size and increase in generations.

The initial number of ES instances denotes the localized nature of the algorithm. The more the number of instances, the more is the approach of the algorithm not to converge at the optima assuming a limited constant computation. It may however be able to escape the local optima and get towards global optima. 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. The more the number of instances, the more is the randomness or convergence to local optima which may be desirable in real life scenarios.

The total generations of the master EA is similar in nature to the role of number of generations in any EA. The more generations require more computation. In many cases the higher generations may result in working over an already converged result and hence may not be very useful. Suppose we keep the computation time constant, higher generation would mean a shorter computation per iteration or small number of individuals. This is an indication of higher global optima search trends, escaping from the local optima.

The step size is kept sufficiently low for easy search for the local optima by the local ES. A high initial step size 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.

The mutation rate decides the displacement of the mean individuals of the slave ES in the algorithm. A high mutation might place them at completely random place and the algorithm would behave more or less random. At the same time keeping very small values may make them converge at more or less same place at every run of the slave ES. This would be a waste of computation since the results may not improve significantly over generations.

Similar is the case with the factor denoting contribution of previous generation to next generation (c). A small value of this factor denotes complete randomness of the algorithm with every run of ES starting at some random point. A larger value may lead to no incentive for exploration in the complex space which is a need of the algorithm.

The increase in step size is a similar factor in its behaviour in algorithmic working. A higher value of this factor would normally mean that we intend to get large changes in step size. Hence the algorithm makes rapid changes in step size per iteration and in the process quickly goes to attain the global trends. This is unlike the case with smaller value of change in step size where the step size behaves constant in nature and behaves as per the set step size. Here we make specific strategy that is more or less passive in nature. This has an advantage of the algorithm able to perform well with larger computation time if suitable value of step size can be guessed. The increase in generation follows similar trends.

6 RESULTS

The algorithm was implemented and tested using MATLAB platform. The algorithm made the master EA. The slave ESs was taken by the implementation of CMA-ES available at (Hansen 2008). The parameters of the slave ES were passed by the master EA in its execution.

For the purpose of testing we used 11 different test functions used in literature (Garai and Chaudhuri 2007, 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 15 master generations. The mutation rate was fixed as 0.03 and c as 0.8. The initial number of iterations of the slave ES was 2500 which could increase by 25%. The step size could vary from an initial of 0.1 and increase by 0.7. All simulations took 2 to 4 seconds running time as per the set parameters except the last function (F11) which took approximately 5 to 7 seconds for the different dimensions.

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	$F = \sum_{i=1}^{3} 100(x_i^2 - x_{i+1})^2 + (1 - x_i)^2$	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

We used Conventional CMA-ES, 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. Step size of conventional CMA-ES was fixed to 0.5 and the generation was fixed to 55000. The mutation rate of GA 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: Comparative analysis of working of the proposed algorithm with GA, PSO and CMA-ES on the objective functions

S. No.	Optimal Value	Proposed Algorithm	GA	PSO	CMA-ES
F1	3	3.0000	142.0500*	3.0000	3.0000
F2	0	0.0000	0.0000	0.0000	0.0000
F3	0	0.0000	0.0000	0.0000	0.0000
F4	0	0.0000	0.0000	0.0002	0.0000
F5	0	0.0000	0.0000	0.0000	0.0000
F6	-10	-10.0000	-10.0000	-10.0000	-10.0000
F7	-300	-300.0000	-269.4620	-299.917	-300.0000
F8	-8.2436	-8.2436	-8.24361	-8.2436	-8.2436
F9	-4	-4.0000	-4.0000	-4.0000	-4.0000
F10	-1.0316	-1.0316	-0.1123#	-1.0316	-1.0316
F11(i) (ii) (iii)	0 0 0	0.0000 0.0000 0.0000	-0.9572$ 6.5080 31.3468	0.0146 0.07240 1.0511	0.0000 0.0000 0.0000

Table 2 showcases the performances of the proposed algorithms compared to standard CMA-ES, GA and PSO. The 11 functions given to the algorithm to optimize represent a range of simple to complex functions. We separately discuss the results in separate 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 4 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. 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 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. The CMA-ES gave very good results that were at par with the proposed algorithm as compared to all the scenarios presented.

In order to effectively compare the proposed algorithm with CMA-ES we use another function that represents a much more complex landscape than the above presented algorithms. This is the rastringin's function whose equation is given by (12).

$$F(x) = 20n - \sum_{i=1}^{n} \left(x_i^2 - 10\cos 2\pi x_i \right) \tag{12}$$

In order to visually analyze the complexity of the function we draw the rastringin's function in 2-dimensions given in figure 4. This forms the fitness landscape over which the algorithms work and find the optima. It may again be seen that this extended over multiple dimensions pose a very complex problem because of which many algorithms fail.

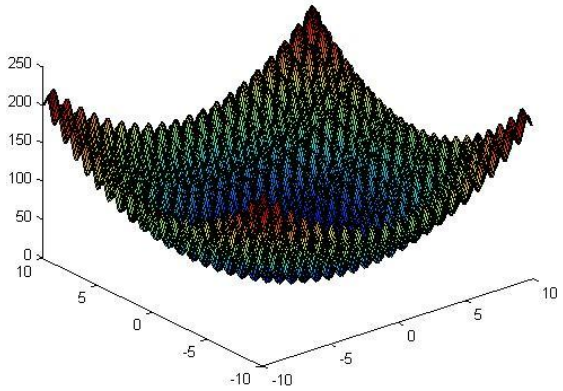

Figure 4: Fitness landscape (in 2 dimensions) of Rastrigin's function

The function was used in a domain of -10 to 10 for all axes. The conventional CMA-ES generations were kept as 200000 for these runs which was the only change made from the previous settings. The proposed algorithm had same settings from the previous run. The simulations took around 7 seconds which was kept equal for both algorithms. Table 3 summarizes the results for various dimensions (n) of the function. It may be clearly seen that initially the performance of the two algorithms were somewhat at par with each other with decent performances. But as we kept increasing the dimensionality which denotes complexity, the proposed algorithm kept exceeding the conventional CMA-ES in optimality.

Table 3: Comparative analysis of working of the proposed algorithm with CMA-ES on the rastrigin's functions

S. No.	Dimensionality	Proposed Algorithm	CMA-ES
1	1	0.0000	0.3980
2	2	0.0000	1.04472
3	5	0.4477	5.72099
4	10	3.2903	14.2278
5	15	6.4273	18.0584

Based on the simulations, it may be extrapolated that all algorithms would pose problems of high complexity as we keep increasing the dimensionality. However, ES, GA and PSO in higher dimensionality 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.

7 CONCLUSIONS

In this work a hierarchical implementation was followed consisting of master and slave. The slave was a simple instance of CMA-ES that helped in optimization in the allocated space in the fitness landscape. The master consisted of an evolutionary technique to control the slave algorithm. This was to fix the parameters of the slave ES and to coordinate the various instances of ESs running in parallel. The slave ES helped in convergence to some local minima. The master EA helped in addition of global traits to the entire algorithm as the algorithm proceeds. In this manner we get a contrast of both global and local characteristics. Not only the algorithm is able to attain the global optima, but it is able to give the optimal outputs to even complex problems presented in finite amount of time. The master EA was programmed with special evolutionary operators that aid in the generation of individuals that are able to further improve themselves in the further ES runs as per the strategy of the master EA. This is a unique property of the algorithm that can be controlled by the discussed parameters to maintain tradeoffs between time and local and global optima. The algorithm is especially designed for the fitness landscapes that are spread across multiple dimensions and have multiple modalities that make the optimization task very difficult. The proposed algorithm is able to perform well while most conventional algorithms fail due to the sensitive nature of the algorithm and converge at some other point than the optima.

The testing of the algorithm was done using the benchmark functions available in literature (Garai and Chaudhuri 2007; Shi et al 2005). These functions ranged in their complexity and dimensionality. We tested the performance of the algorithm in comparison to the conventional ES, GA and PSO. The results revealed that the conventional ES, GA and PSO were able to give a good performance in most of the objective functions.

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 EA as a means of parameter setting of the local ES. This is a much complex relationship between parameters that require a much formal modelling 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.

8 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] Hansen, N. (2008), The CMA Evolutionary Strategy: A Tutorial, Available at http://www.bionik.tu-berlin.de/user/niko/cmatutorial.pdf

[12] 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.

[13] 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

[14] 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

[15] 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

[16] 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

[17] 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,

[18] 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

[19] 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

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

[21] 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

[22] 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

[23] Mitchell, Melanie (1998), *An Introduction to Genetic Algorithms*, MIT Press

[24] Muhlenbein, Heinz (1992), 'How genetic algorithms really work I. Mutation and Hill Climbing', Parallel Problem Solving from Nature,

[25] 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

[26] Rechenberg, I. (1973), *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipiender biologischen Evolution*, Frommann-Holzboog, Stuttgart.

[27] Rudolph, Gunter (1997), Evolutionary Strategis, In Evolutionary Algorithms and Their Standard Instances*, Handbook on Evolutionary Computation*, IOP Publishing Ltd and Oxford University Press

[28] Schwefel, H.P. (1995), *Evolution and Optimum Seeking*, John Wiley.

[29] 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

[30] 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

[31] Shukla, Anupam, Tiwari, Ritu, and Kala, Rahul (2010), *Real Life Applications of Soft Computing*, CRC Press

[32] 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

[33] Syswerda, Gilbert (1989), 'Uniform Crossover in Genetic Algorithms', In *Proceedings of the 3rd International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, pp 2 - 9

[34] 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

[35] 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

[36]Yao, Xin (1993), 'Evolutionary Artificial Neural Networks', In *International Journal of Neural Systems*, Vol 4, No 3, pp 203-222

[37] 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

About the authors

Rahul Kala

Rahul Kala is an Integrated Post Graduate (BTech and MTech in Information Technology) student in Indian Institute of Information Technology and Management Gwalior. His areas of research are hybrid soft computing, robotic planning, biometrics, artificial intelligence, and soft computing. He has published about 35 papers in various international and national journals/conferences and is the author of 2 books. He also takes a keen interest toward free/open source software. He secured All India 8th position in Graduates Aptitude Test in Engineeging-2008 Examinations and is the winner of Lord of the Code Scholarship Contest organized by KReSIT, IIT Bombay and Red Hat.

Dr. Ritu Tiwari

Dr. Ritu Tiwari is serving as an Assistant Professor in 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 over 50 research 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.

Prof. Anupam Shukla

Prof. Anupam Shukla is serving as a Professor in Indian Institute of Information Technology and Management Gwalior. He heads the Soft Computing and Expert System Laboratory at the Institute. He has 20 years of teaching experience. His research interest includes Speech processing, Artificial Intelligence, Soft Computing, Biometrics and Bioinformatics. He has published over 100 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.