A distributed hierarchical genetic algorithm for efficient optimization and pattern matching

Gautam Garai[a,*], B.B. Chaudhuri[b]

[a]Computer Division, Saha Institute of Nuclear Physics, 1/AF Bidhannagar, Kolkata 700064, India
[b]Computer Vision and Pattern Recognition Unit, Indian Statistical Institute, 203 B.T. Road, Kolkata 700108, India

Received 18 July 2005; received in revised form 16 January 2006; accepted 19 April 2006

Abstract

In this paper we propose a new approach in genetic algorithm called distributed hierarchical genetic algorithm (DHGA) for optimization and pattern matching. It is eventually a hybrid technique combining the advantages of both distributed and hierarchical processes in exploring the search space. The search is initially distributed over the space and then in each subspace the algorithm works in a hierarchical way. The entire space is essentially partitioned into a number of subspaces depending on the dimensionality of the space. This is done in order to spread the search process more evenly over the whole space. In each subspace the genetic algorithm is employed for searching and the search process advances from one hypercube to a neighboring hypercube hierarchically depending on the convergence status of the population and the solution obtained so far. The dimension of the hypercube and the resolution of the search space are altered with iterations. Thus the search process passes through variable resolution (coarse-to-fine) search space. Both analytical and empirical studies have been carried out to evaluate the performance between DHGA and distributed conventional GA (DCGA) for different function optimization problems. Further, the performance of the algorithms is demonstrated on problems like pattern matching and object matching with edge map.

Keywords: Genetic algorithm; Optimization; Coarse-to-fine; Distributed; Variable resolution; Pattern matching

1. Introduction

Genetic algorithm (GA) is a class of stochastic search methods inspired by the Darwinian's concept of survival of the fittest individual in natural selection and hence categorized as a class of evolutionary algorithms. The technique was first formalized by Holland for use in adaptive systems [1]. It has attracted a great deal of attention from researchers in numerous fields as a way of effective and efficient search for optimization in complex, multi-dimensional space. GA represents a parallel adaptive search process which is executed with modification of genetic parameters in a wide variety of problems [2–5].

Such evolutionary computation techniques try to iteratively reach the optimal solution of a problem closest to the actual or global solution. The algorithm starts with a set of individuals (called *population*) and advances towards the solution with three basic operations namely, reproduction, crossover and mutation. However, there is always a possibility that the solution vector gets stuck in a local optimum. Several modified algorithms have been proposed and implemented to jump out of this optima. One possibility is to maintain the diversity in the population and/or to distribute the individuals all over the search space as uniformly as possible. However, this usually leads to increase in the computational complexity of the system.

Among several approaches for improving performance, Cavicchio introduced a method in 1970 that preserved the best individuals by replacing the inferior parent if the offspring's fitness exceeded that of the inferior parent [6]. De Jong's *crowding* scheme [7] also aimed to retain the diversity and best individuals in the population by replacing the maximally similar strings. In 1987 Goldberg and

Richardson [8] used the sharing metaphor to induce niche and species in their new technique. In this scheme a *sharing function* is implemented in distributing the individuals over the search space, determining the neighborhood and the degree of sharing for each string in the population. Similarly, Eshelman in his CHC algorithm [9] combined a conservative selection technique that always preserved the fittest individual found with a radical recombination operator that generated offspring with maximum difference from their parents. Fogel [10] as well as Back and Schwefel [11] tested some optimizing functions for their algorithms to show how the evolutionary method with self-adaptive mutation performed better than the method without self-adaptive mutation.

The implementation of genetic and/or evolutionary algorithm for solving various complex problems often increases the computation time and the researchers try to increase the speed of the algorithm using parallel/distributed GA/EA when the computation time for a problem increases. Various implementations of parallel GAs are discussed in Refs. [12,13]. GAs are naturally suited to implementation on parallel architecture. Two approaches of parallel GAs namely, the *inland* model and the *diffusion* model, are discussed in Refs. [14,15], respectively. The inland model divides the population in a number of smaller populations. The migration of individuals among the subpopulations occurs periodically during the progress of the search. However, the number of individuals to be migrated and the occurrence of migration are important debatable problems [12]. On the other hand, in diffusion model each individual is associated with a spatial location on a low-dimensional grid. The population is considered as a system of active individuals that interacts only with neighbors. Another useful application of distributed GA is in the performance driven VLSI routing problem which is capable in handling both topological and electrical constraints associated with this problem [16].

In addition, the implementation of parallel GAs are also found in solving the well-known NP-hard Travelling Salesman problem on a cluster of nodes [17]. A *master-slave* technique is used to implement the parallel/distributed GA to obtain the optimal and/or suboptimal travelling path. Moreover, in labor scheduling problems the distributed GA is applied to determine the number of employees and their work schedule to minimize the labor expenses and expected opportunity costs [18]. Multi-objective optimization problem is also solved by parallel genetic/evolutionary algorithm. A major computational bottleneck in many contemporary multi-objective evolutionary algorithm applications as well as in other numerical or real-valued design/optimization problems is the calculation of complex non-linearmulti-objective problem functions. Such a situation implies algorithmic parallelization towards the improvement of computational complexity. The major parallel multi-objective genetic/evolutionary algorithms are discussed and some observations included in Ref. [19].

The GAs have been employed in pattern recognition and image processing problems such as medical image registration, contour recognition, geometric primitive extraction are available in Refs. [20–22]. Other applications like normalization of Chinese handwriting, classification of endothelial cells and evaluation of earthquake risk for geological structures are also reported in literature [23–25]. Moreover, the GAs have been employed in error-correcting graph isomorphism, optimization of feature extraction and dot pattern matching [26–28].

In this paper we propose a modified scheme of distributed GA named *distributed hierarchical genetic algorithm* (DHGA). We have utilized the power of distributed as well as hierarchical techniques. The distributed method strives to minimize the drawbacks of the stochastic search method in exploring the search over the entire space. Here, the whole space S is initially partitioned into a number of subspaces s_i's, $\forall i \in \{1, \ldots, m\}$ depending on the number of hyperplanes (each plane is perpendicular to each other) required to represent the function to be solved to distribute the search. The algorithm (DHGA) starts with m independent populations that are distributed in m subspaces of identical dimension. This criterion helps to run m GAs in m subspaces concurrently. The salient features of the proposed hybrid scheme are that it conducts the genetic process concurrently in various segments over the entire space and simultaneously in each segment the sequential GA is processed hierarchically in parallel. The performance of DHGA is compared with the distributed version of the conventional genetic algorithm (CGA) named DCGA. Here, the search space is similarly partitioned into subspaces as done in DHGA. Then the sequential genetic process (CGA) is spread over the space and runs in each subspace parallely.

The hierarchically processed sequential GA is a *coarse-to-fine* search technique [29]. Here, the search space resolution in each subspace is increased with the transfer of the search from one hypercube to the neighboring one. In multi-dimensional space the search shifts from one hypercube to the neighboring hypercube T_{max} times depending on the convergence of the existing population and the solution obtained so far and continues for G_{max} iterations. The GA initially starts with string length l and resolution r_1 where the search space is the entire area of a subspace. In the second stage GA jumps to the neighboring hypercube (smaller in size than the previous one) surrounding the current best solution redefining the search space. Since the new search space is now smaller in size and the string length remains the same i.e., l, the present resolution r_2 of the search space or the hypercube is more than r_1 i.e., $r_2 > r_1$ (elaborated in Section 2.4). In the subsequent stages, a similar procedure is followed and eventually the precision of the optimizing function is increased.

The rest of the paper is organized as follows. Section 2 describes the hierarchically processed GA as well as the basic steps of a conventional GA (CGA). This also discusses the variation of search space resolution and the alteration of its physical size in successive steps. The presentation and implementation of the proposed evolutionary

algorithm (DHGA) is described in Section 3. Section 4 includes the analysis of the empirical results of using DHGA on several interesting and well-known unimodal and multimodal optimization functions compared with DCGA. It also discusses the performance evaluation of the algorithms on pattern recognition problem involving dot pattern matching and object matching with edge map. Section 5 concludes the paper.

2. Function optimization by genetic algorithms

A global optimization problem can generally be formulated as a pair (S, f) where $S \subseteq R^n$ is a bounded set on R^n and $f : S \mapsto R$ is a n-dimensional real-valued function. The objective of the problem is to find a point $\mathbf{x}_{opt} \in S$ on R^n such that $f(\mathbf{x}_{opt})$ is a global optimum on S. We have to find $\mathbf{x}_{opt} \in S$ according to the following equations for minimization or maximization problems, respectively:

$$\forall \mathbf{x} \subseteq S : f(\mathbf{x}_{opt}) \leqslant f(\mathbf{x}), \tag{1}$$

$$\forall \mathbf{x} \subseteq S : f(\mathbf{x}_{opt}) \geqslant f(\mathbf{x}), \tag{2}$$

where f may not be a continuous function but it must be bounded. In this paper we consider the unconstrained function optimization only.

2.1. Conventional genetic algorithm

The CGA with self-adaptive mutation is algorithmically represented in the following steps.

Step 1: Generate randomly the initial population of μ individuals and let $g = 1$. Initialize δ and η where δ is the crossover probability and η is the mutation probability.

Step 2: Evaluate the fitness score for each individual $\mathbf{x}_i$, $\forall i \in \{1, \ldots, \mu\}$ of the population based on the objective function, $f(\mathbf{x}_i)$ where $\mathbf{x}_i$'s are objective variables.

Step 3: Select a pair of individuals $\mathbf{x}_\alpha$ and $\mathbf{x}_\beta$ at random depending on their fitness values (using roulette wheel method) from the population of μ individuals.

Step 4: Conduct crossover between the chosen individuals $\mathbf{x}_\alpha$ and $\mathbf{x}_\beta$ with δ and mutate each of their bits with adaptive mutation probability η_a such that

$$\eta_a = \eta_0 + \frac{3\eta_0 * g}{G_{max}}, \tag{3}$$

where G_{max} is the maximum number of iterations and η_0 is the initial mutation probability.

Each pair of parents $(\mathbf{x}_\alpha, \mathbf{x}_\beta)$ thus creates a pair of new individuals called offsprings $(\mathbf{x}'_\alpha, \mathbf{x}'_\beta)$ to generate a pool of individuals, $\mathbf{x}'_j$, $\forall j \in \{1, \ldots, \mu\}$ as a population of next generation.

Step 5. Terminate the process if the stopping criterion $(g > G_{max})$ is satisfied. Otherwise, $g = g + 1$ and go to Step 2.

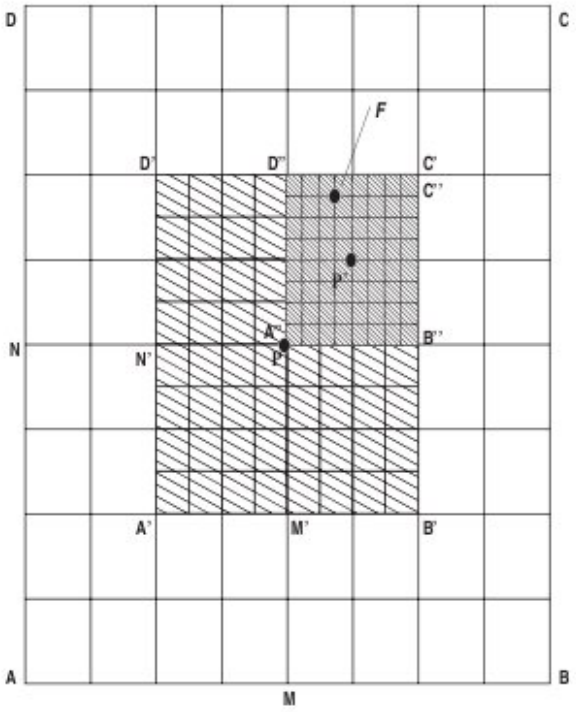

Fig. 1. Transfer of GA search from one square to the neighboring square in a space in three stages tending to increased resolution.

2.2. Hierarchically processed genetic algorithm

The search space is initially partitioned into a number of subspaces to distribute the GA over the entire space. Thus the algorithm enhances the scope of exploring each part of the space uniformly as much as possible. In each subspace a repetitive stochastic search is conducted from *coarse-to-fine* resolution hierarchically with all three basic operations of classical GA namely, selection/reproduction, crossover and mutation. The details of each basic operation is described in Ref. [30]. The genetic process starts with a population $P^t(g)$ (where $1 \leqslant t \leqslant T_{max}$ and $1 \leqslant g \leqslant G_{max}$) of μ (population size) randomly created individuals and is implemented with adaptive mutation. The hierarchical GA in a subspace, however, differs from the CGA for the convergence status of the population in the following manner:

1. The search is transferred from one hypercube to the neighboring hypercube for a specified number of times T_{max} in a search space shown in Fig. 1.
2. The search space resolution and mutation probability η are redefined (according to Eq. (5)) with the transfer of the search to the neighborhood hypercube.

Let us now define the parameters of the problem $f(\mathbf{x})$ in R^n_{SS} where $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and R^n_{SS} represents a subspace with n-dimension. The search starts with low resolution space and the algorithm finds a best solution $\mathbf{x}^t_B = (x^t_{B,1}, x^t_{B,2}, \ldots, x^t_{B,n})$ which is not changed for several

consecutive K_g generations. We have chosen $K_g = 5$ in our experiment. Since the search advances from lower to higher resolution in several (T_{max}) steps, at tth stage it is assumed to be converged if the best solution $\mathbf{x}_B^t$ remains unchanged for K_g times. Subsequently if there are no change of GA parameter values with further iterations then the algorithm cannot normally find a better solution than the solution (best at the current step) obtained so far. The search in such a situation is transferred to the neighboring hypercube. The transfer of the GA search process in successive steps is depicted in Fig. 1 and in a subspace at tth stage the hypercube with dimension n can be represented as follows:

$$R_{SS}^{t,n} = \{r_{SS}^{t,1}, r_{SS}^{t,2}, \ldots, r_{SS}^{t,n}\}, \tag{4}$$

where $1 \leqslant t \leqslant T_{max}$ and $r_{SS}^{t,j} > 0$, $j = 1, 2, \ldots, n$.

Once the search has converged at $(t-1)$th step the GA is reinvoked from the hypercube $R_{SS}^{t-1,n}$ to the new hypercube $R_{SS}^{t,n}$ with higher mutation rate and higher resolution search space (discussed in Section 2.4). The adaptive mutation probability η_a at tth stage is defined below:

$$\eta_a^t = \eta^1 * t, \tag{5}$$

where $1 < t \leqslant T_{max}$ and η^1 is the initial mutation probability at $t = 1$.

The population $P^t(g)$ at tth step is reproduced after the transfer of the search to $R_{SS}^{t,n}$ hypercube. The individuals that are located both in $R_{SS}^{t-1,n}$ and in $R_{SS}^{t,n}$ except the elitist individual in $R_{SS}^{t-1,n}$ will be destroyed from the population $P^{t-1}(g)$ where $(1 < t \leqslant T_{max})$ and $(1 < g \leqslant G_{max})$. A new population $P^t(g)$ of μ individuals is regenerated with $\mu - 1$ randomly created new individuals and the elite one of $P^{t-1}(g)$. Thus, the diversity of the population is automatically restored at each stage.

Each individual in a population is a string of bits which in two-dimensional space generally represents the values of x and y-coordinates of a probable solution for the optimization of mathematical functions. In case of pattern recognition problems for either pattern or object matching, the same process is followed. Here, an individual consists of three parameters instead of two. Since the pattern matching should be done not only by translation but also by rotation, the problem invokes one more degree of freedom. Again if an angular resolution of r_θ is expected, then the chromosome (an individual in the population) length should be increased by k according to the following relation and the angular resolution remains unaltered till the end of the algorithm:

$$2^k \geqslant \left\lceil \frac{2\pi}{r_\theta} \right\rceil.$$

2.3. Redefinition of search space

It is now understood that at tth stage $(t > 1)$ the search moves from one hypercube $R_{SS}^{t-1,n}$ to the neighboring hypercube $R_{SS}^{t,n}$ if the solution is unaltered for K_g generations. The area of $R_{SS}^{t,n}$ is subsequently reduced from that of $R_{SS}^{t-1,n}$. Fig. 1 pictorially shows the transfer of the search process in three stages in two-dimensional space.

Let us now discuss how the search space dimension is redefined with the change of the search from one hypercube to the neighboring hypercube. Initially, let the subspace be represented by the area $ABCD$ and consider the chromosome length be 6 bits where 3 bits are used to represent the x-coordinate and similarly 3 bits for the y-coordinate. The space $ABCD$ is thus partitioned into 2^3 i.e., 8 equal parts both along x and y-axes. The GA then begins its search over the entire space $ABCD$ and finds the first best solution $\mathbf{x}_B^1$ (shown by the point P in Fig. 1) at the first stage when $\mathbf{x}_B^1$ is not modified for K_g generations. At stage 2, the process shifts its attention to the new search space $A'B'C'D'$ (shown by a different shade in Fig. 1) and starts the stochastic search process with a new population $P^2(g)$ (where $t = 2$ and $1 < g \leqslant G_{max}$) over the space $A'B'C'D'$, instead of $ABCD$. From Fig. 1 it is understood that $A'B'C'D'$ is dimension-wise smaller than $ABCD$. The space $A'B'C'D'$ is formed according to the following steps.

Step 1: Consider the point P in the search space $ABCD$ of Fig. 1.

Step 2: Draw two perpendiculars from P on the lines $\overline{AB}$ and $\overline{AD}$. The perpendiculars cut $\overline{AB}$ and $\overline{AD}$ at M and N, respectively. Similarly, the perpendiculars cut $\overline{A'B'}$ and $\overline{A'D'}$ at M' and N', respectively (see Fig. 1).

Step 3: Extend the line $\overline{A'B'}$ from the point M' to both A' and B' until $\overline{M'B'} = \overline{MB}/2$ and $\overline{M'A'} = \overline{MA}/2$ so that $\overline{A'B'} = \overline{AB}/2$.

Step 4: If $\overline{M'B'} < \overline{MB}/2$ then $\overline{M'A'} = \overline{MA}/2 + (\overline{MB}/2 - \overline{M'B'})$. On the other hand, if $\overline{M'A'} < \overline{MA}/2$ then $\overline{M'B'} = \overline{MB}/2 + (\overline{MA}/2 - \overline{M'A'})$ (see Fig. 1).

Step 5: Similarly, $\overline{A'D'}$ is drawn following the Steps 3 and 4 for the formation of the rectangle $A'B'C'D'$.

Now the new search space $A'B'C'D'$ is again divided into 2^3 i.e., 8 equal parts along both x and y-axes in R^2 as before since the chromosome length remains 6-bit. Similarly, the GA finds the current best solution $\mathbf{x}_B^2$ in the new search space (depicted by a point P' in Fig. 1) when the solution is not altered for K_g generations. At the third or final stage $(t = 3)$, the GA again moves to another search space surrounding $\mathbf{x}_B^2$ following the same procedure. The current newly defined rectangular search space is $A''B''C''D''$ and the partitioning process is repeated as before. GA restarts its search over this space and finds the global or near-global solution $\mathbf{x}_B^3$ which is actually identified by $\mathbf{F}$ in Fig. 1.

2.4. Variation of search space resolution

From Section 2.3 it is noted that the search space resolution is changed with the transfer of GA from one hypercube to the neighboring hypercube. Fig. 1 illustrates how the genetic search at tth step shifts to the neighboring hypercube $R_{SS}^{t,n}$ from the hypercube $R_{SS}^{t-1,n}$ and how its resolution is modified. Variable resolution search is similar

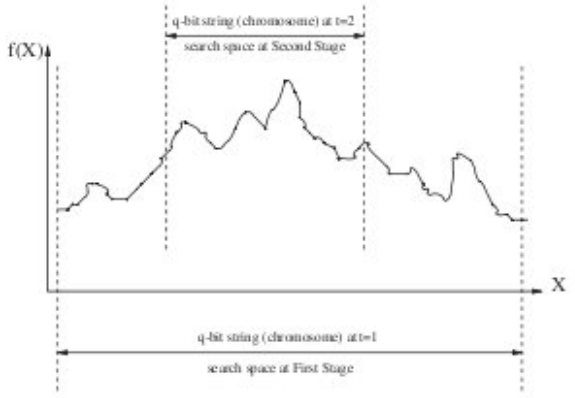

Fig. 2. Variation of resolution with constant chromosome size of q-bit in one-dimensional.

to the dynamic parameter encoding (DPE) technique [31] but the search space division scheme is completely different. We discuss the variation of search space resolution in R^2 (see Fig. 1).

We can achieve the variable resolution search space in subsequent stages (starting from lower to higher resolution) in the following way. Figs. 1 and 2 depict the change of resolution with the progress of the random genetic search. At first, the search starts with lower resolution and detects a solution faster when the solution is unaltered for K_g generations. At the second stage the search is similarly performed with a higher resolution (i.e., higher than the previous resolution of the previous search) resulting in efficient detection of a solution (may be a global optimum or near-optimum) as before but better than the previous one in value. Fig. 1 shows how the resolution of the search space is modified in successive stages.

From the discussion in Section 2.3 and Fig. 1 in R^2 it is clear that at each stage of the hierarchically processed GA, the dimension of the search space is reduced. Fig. 1 illustrates three rectangular search spaces ($ABCD$, $A'B'C'D'$ and $A''B''C''D''$) with different shades. At each stage the space is partitioned into the same number of equal parts along both x and y-axes. However, the dimension of the newly defined search space at each step t ($1 < t \leqslant T_{max}$) is reduced in size. This phenomenon of GA eventually increases the resolution of the new search space at each subsequent stage since the same number of chromosome bits represents a smaller physical space. Thus, the precision of the solution of the optimizing function is also increased.

The process is illustrated by another example in Fig. 2 which depicts an arbitrary function $y = f(x)$ in two-dimensional space. At $t = 1$ the search space is represented by a string or chromosome of q-bit and let the length of the space (along x-axis) be L. At $t = 2$ the search space length is $L/2$. Now at $t = 1$, L is divided into 2^q equal parts and the length of each partition is $L/2^q$. Similarly, the search space at $t = 2$ is divided with the same string length of q-bit and the length of each partition becomes $\frac{L/2}{2^q}$ i.e., $L/2^{q+1}$. Thus, at every stage the search space will be halved and partitioning of the new search space will be continued similarly with the same string length if the number of stages is more than two. Thus, in Fig. 2 the resolution is increased at stage 2. However, in case of CGA the initial and final search space is same with length L and it is partitioned into 2^{q+1} equal parts to maintain the identical resolution of hierarchically processed GA. As a result the chromosome length in this technique becomes $(q + 1)$ bits.

3. Distributed hierarchical genetic algorithm

In the proposed scheme the entire search space is divided into a number of segments which are either equal or different in size. The hierarchically processed sequential GA is then distributed in each segment with a population of randomly generated μ individuals. Here we have discussed how DHGA advances its parallel searching process over the entire space S.

3.1. Distribution of population

In the simple sequential GAs there is a possibility that after a few iterations the GA search may be confined to a local optimum. This may happen due to the dearth of diversity in the population for which the search cannot explore the entire space uniformly. Thus, the best solution obtained so far is not updated for several consecutive generations and the search is eventually terminated without reaching the global optimum. This nature is exhibited either for the deviation of the GA from the location of the global optimum solution or due to the inefficiency of hill climbing. The problem can, however, be either eliminated or reduced if the search is distributed uniformly as far as possible over the entire space S. The whole space is initially divided into m smaller subspaces of equal dimension where the number of subspaces s_i's, $\forall i \in \{1, \ldots, m\}$ is determined by the number of hyperplanes required to represent the optimization function. The population of equal size is then distributed in each segment. Thus, in the entire space (with segmented subspaces) the population is more distributed than the population distribution of GA over the entire single space since each s_i now contains μ individuals. This technique invokes the advantage of spreading the search over a broader space. Simultaneously in each subspace the GA progresses hierarchically from *coarse-to-fine* resolution which results in an efficient search to find the global optimum. Let us now consider m independent populations $P_i^I(g)$'s, $\forall i \in \{1, \ldots, m\}$ to be distributed in m subspaces where $P_i^I(g) \neq P_j^I(g)$, $\forall i, j \in \{1, \ldots, m\}$ and $1 \leqslant t \leqslant T_{max}$. If the number of hyperplanes is n, then we can define m as follows:

$$m = 2^n, \tag{6}$$

where $n \geqslant 2$.

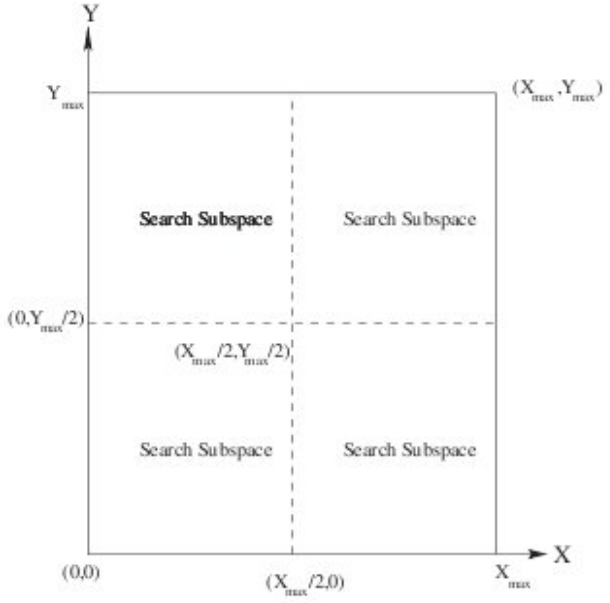

Fig. 3. The entire space partitioning on two dimensions.

The partitioning process of the entire space is shown in Fig. 3 with $n=2$ for the function, $z=f(x,y)$ in R^2. Now the GAs in each subspace run concurrently in successive steps with an independent population of μ individuals on m independent nodes. Thus, DHGA distributes equal number of individuals at each segment of the search space S from the beginning till the end of the entire search process. This technique helps to reduce the possibility of the solution being trapped in the local optima of the optimizing function since such distribution partially corresponds to uniform population distribution over the entire space S. In R^n $(n>3)$ each subspace s_i is a hypercube.

Similarly, the entire space S is divided into m subspaces of equal dimension for distributed CGA (DCGA). Now CGA with a population of randomly created μ individuals is distributed in m subspaces so that m CGAs can run concurrently like DHGA.

3.2. Progress of DHGA

The entire space S is partitioned into m subspaces and in all subspaces and/or hypercubes the hierarchically processed GA is invoked in parallel with an independent and randomly created population. Thus, the process starts in each subspace $s_i, \forall i \in \{1,\ldots,m\}$ with a population $P_i^t(1), \forall i \in \{1,\ldots,m\}$ of μ individuals which are not analogous to each other. In each subspace s_i, the hierarchical genetic process advances through variable resolution search space when at each stage the search is transferred to a neighboring hypercube depending on the convergence status of the population and the solution obtained at the current stage. Finally, the GA terminates in each subspace after a given number of generations. DHGA thus converges faster to the global or near-global optimum.

Now, in ith subspace $s_i, \forall i \in \{1,\ldots,m\}$ in R^n, a global optimization problem can be formulated as a pair (s_i, f) where $s_i \subseteq R^n_{SS,i}$ is a bounded set on $R^n_{SS,i}$ where $R^n = \{R^n_{SS,1}, R^n_{SS,2}, \ldots, R^n_{SS,m}\}$ and $f : s_i \mapsto R_{SS,i}$ is a n-dimensional real-valued function. The objective of the method is to locate a point $\mathbf{x}_{opt,i} \in s_i$ on $R^n_{SS,i}$ such that $f(\mathbf{x}_{opt,i})$ is a global optimum on s_i.

The search starts with low resolution in each subspace $s_i, \forall i \in \{1,\ldots,m\}$ and the algorithm finds a best solution $\mathbf{x}^t_{B_i} = (x^t_{B_i,1}, x^t_{B_i,2}, \ldots, x^t_{B_i,n})$ in s_i which is not changed for several consecutive K_g generations. The search in ith subspace $s_i, \forall i \in \{1,\ldots,m\}$ is then transferred to the neighboring hypercube. In ith subspace and at tth stage the hypercube can be represented as follows according to Eq. (3):

$$R^{t,n}_{SS,i} = \{r^{t,1}_{SS,i}, r^{t,2}_{SS,i}, \ldots, r^{t,n}_{SS,i}\},$$

where $i = 1,2,\ldots,m$, $1 \leqslant t \leqslant T_{max}$ and $r^{t,j}_{SS,i} > 0$, $j = 1,2,\ldots,n$.

Once the search has converged at $(t-1)$th step in s_i the GA is reinvoked from the hypercube $R^{t-1,n}_{SS,i}$ to the new hypercube $R^{t,n}_{SS,i}$ with a higher mutation rate and higher resolution of search space. The adaptive mutation probability η_a in $s_i, \forall i \in \{1,\ldots,m\}$ at tth stage $(t>1)$ is evaluated according to Eq. (5) with the transfer of the search to the neighboring hypercube. The new population in $R^{t,n}_{SS,i}$ surrounding $\mathbf{x}^t_{B_i}$ in s_i is then generated following the same procedure described in Section 2.2.

3.3. Extraction of the global optimum solution

In each subspace s_i, $\forall i \in \{1,\ldots,m\}$ the hierarchically processed sequential GA is invoked for T_{max} times and runs simultaneously in all subspaces in parallel. The GA in each s_i terminates after G_{max} iterations finding a best solution within each subspace although the best solution in the corresponding subspace may not be the global optimum of the optimizing function. Since the entire space S is divided into m subspaces such that $S = \{s_1, s_2, s_3, \ldots, s_m\}$, the number of optimal solutions are also m which are represented as $\mathbf{x}_{opt,1}, \mathbf{x}_{opt,2}, \ldots, \mathbf{x}_{opt,m}$ such that in ith subspace s_i, the optimum solution is $\mathbf{x}_{opt,i} \in s_i$ on $R^n_{SS,i}$ $\forall i \in \{1,2,\ldots,m\}$. Now, the objective of the problem is to find a point $\mathbf{x}_{opt} \in S$ on R^n so that $f(\mathbf{x}_{opt})$ is a global optimum on S. The global solution can be achieved by either of the following ways depending on the nature of optimizing function:

$$\mathbf{x}_{opt} = \mathbf{Min}(\mathbf{x}_{opt,1}, \mathbf{x}_{opt,2}, \ldots, \mathbf{x}_{opt,m}) \quad \forall \mathbf{x} \subseteq S : f(\mathbf{x}_{opt}) \leqslant f(\mathbf{x}), \tag{7}$$

$$\mathbf{x}_{opt} = \mathbf{Max}(\mathbf{x}_{opt,1}, \mathbf{x}_{opt,2}, \ldots, \mathbf{x}_{opt,m}) \quad \forall \mathbf{x} \subseteq S : f(\mathbf{x}_{opt}) \geqslant f(\mathbf{x}). \tag{8}$$

4. Experimental studies

We have considered a combination of seven unimodal and multi-modal functions (f_1–f_7) for optimization in our experiment. The detailed description of the functions is provided in Section 4.1. The functions are selected to maintain a varying number of local optima from one to a few numbers. The variation of local optima also helps to test the performance of GAs when trapped to any local optimum in the optimizing function. The analysis of the test functions shows the performance of the proposed scheme with DCGA. A comparative study on the performance of DHGA and DCGA is also discussed for optimizing functions f_5, f_6 and f_7 with different function dimensions to show the superiority of DHGA over DCGA. Moreover, we have carried out our experiment on problems like object matching with edge map as well as dot pattern matching.

4.1. Test functions used for optimization

The functions (f_1–f_7) [30,32–34] along with the parameters given in Tables 1 and 2 are considered for empirical studies. Among them, f_1 is a well known unimodal function and is popularly called *De Jong's Test Function*. The dimension of the function n is 3 and each parameter is represented by 7 bits for DHGA and 11 bits for DCGA in the range -10 to $+10$. The length of the string $l = 7 \times 3$ bits for DHGA and $l = 11 \times 3$ bits for DCGA. The function is used for finding the global maximum value $\mathbf{x}_{max} \in S$ in R^n space.

The example of another unimodal function used for maximization is f_2 with dimension $n = 5$. Each parameter of the function for encoding also requires 7 bits for DHGA which implies the string length $l = 7 \times 5$ bits and 11 bits for DCGA with string length $l = 11 \times 5$ bits. The function finds the global maximum value $\mathbf{x}_{max} \in S$ in R^n space in the range between -1 and $+1$.

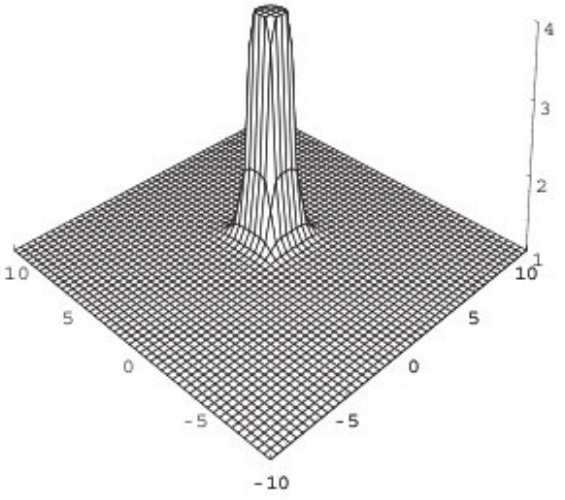

Fig. 4. Two-dimensional version of f_3.

The function f_3 is an interesting *bell-shaped* function with $n = 4$ for finding the global maximum value $\mathbf{x}_{max} \in S$ in R^n space. The function is shown in Fig. 4 in two-dimensional space. In DHGA each parameter is encoded by 7 bits with string length $l = 7 \times 4$ bits and DCGA requires 11 bits with $l = 11 \times 4$ bits for representing its parameters. The range of the function is from -10 to $+10$.

The *Six-Hump Camel-Back Function* is given in Table 2 as the minimization function f_4 with low dimension $n = 2$ compared to other optimizing functions and it has only a few local minima [32]. The number of bits required to represent a parameter and the string length l are 9 bits and 9×2 bits, respectively, for DHGA. Similarly, DCGA requires 12 bits and 12×2 bits, respectively, for encoding parameters. The range of this function is $[-5, 5]$.

In Table 2 the functions (f_5–f_7) are used for finding global minimum value $\mathbf{x}_{min} \in S$ in R^n space for three different

Table 1
Mathematical functions for maximization used in the experimental study

Function	Dimension (n)	Range	Maximum value
$f_1(\mathbf{x}) = \sum_{i=1}^{n} x_i^2$	3	$-10 \leqslant x_i \leqslant 10$	300.0
$f_2(\mathbf{x}) = \sum_{i=1}^{n} \{x_i e^{1-x_i} + (1 - x_i) e^{x_i}\}$	5	$-1 \leqslant x_i \leqslant 1$	8.30
$f_3(\mathbf{x}) = 1 + \sum_{j=1}^{n} \frac{1}{1+\sum_{i=1}^{4}(x_i-1)^6}$	4	$-10 \leqslant x_i \leqslant 10$	5.00

Table 2
Mathematical functions for minimization used in the experimental study

Function	Dimension (n)	Range	Minimum value
$f_4(\mathbf{x}) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	$-5 \leqslant x_i \leqslant 5$	-1.031628
$f_5(\mathbf{x}) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	5, 8 and 12	$-10 \leqslant x_i \leqslant 10$	0.00
$f_6(\mathbf{x}) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1$	5, 8 and 12	$-15 \leqslant x_i \leqslant 15$	0.00
$f_7(\mathbf{x}) = \sum_{i=1}^{n} [x^2 - 10\cos(2\pi x_i) + 10]$	5, 8 and 12	$-5.12 \leqslant x_i \leqslant 5.12$	0.00

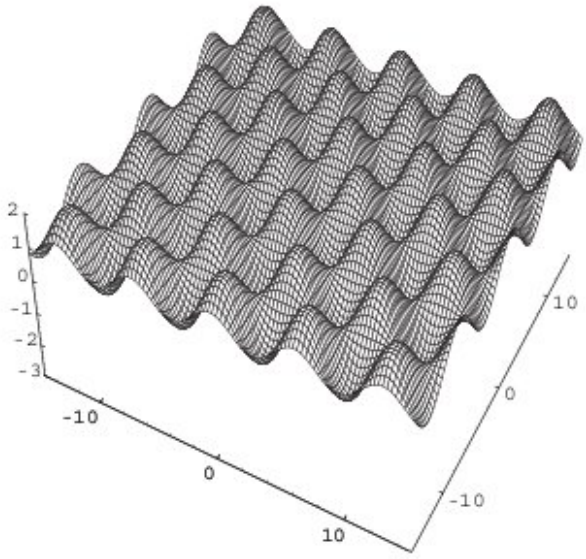

Fig. 5. Two-dimensional version of f_6.

dimensions $n = 5$, 8 and 12. The function f_5 is also a unimodal function and known as *Schwefel's problem* 2.2. Each parameter of the function is represented by 10 bits with string length $l = 10 \times n$ bits (where $n = 5/8/12$) for DHGA and by 15 bits with $l = 15 \times n$ bits (where $n = 5/8/12$) for DCGA in the range -10 to $+10$. f_6 and f_7 are two interesting multi-modal functions where the number of local minima increases exponentially with the problem dimension [33,34]. The function f_6 in two-dimensional space is illustrated in Fig. 5. f_6 is a well-known *Generalized Griewank Function* for minimization in the range $[-15, 15]$. On the other hand f_7 in Table 2 is popularly known as *Generalized Rastrigin's Function* spread over the range between -5.12 and $+5.12$. Each parameter of f_6 and f_7 is encoded by 12 bits with chromosome length $l = 12 \times n$ bits (where $n = 5/8/12$) and by 17 bits with $l = 17 \times n$ bits (where $n = 5/8/12$) for DHGA and DCGA, respectively.

4.2. Experimental setup

In our experiment the self-adaptive mutation is used for both distributed genetic methods. The population size μ is always 50 with the initial population $P_i^t(g) \neq P_j^t(g)$, $\forall i, j \in \{1, \ldots, m\}$ for DHGA and DCGA. The crossover probability δ in the range [0.5–0.9] and the initial mutation probability η in the range [0.002–0.009] have been used as input in our test cases. For mathematical functions the initial population of μ individuals is generated randomly within the specified range in Tables 1 and 2 for both GA techniques discussed here. The chromosome length in each method is different as it is dependent on the resolution of the search space and remains constant till the search process continues. However, the reduction of the search space dimension in subsequent stages for DHGA increases the resolution of the present search space or hypercube. To maintain the same resolution in both approaches, the string length in DCGA is 35–60% larger than DHGA. The total number of stages T_{max} in DHGA is always 4 in our experiment. It may be increased to a higher number for higher search space resolution with the string length of DCGA being increased accordingly. The chromosome length l defined in Section 4.1 is evaluated considering $T_{max} = 4$.

For pattern recognition problem G_{max} in each run is 100 and the population of μ individuals is also randomly generated. The chromosome length is always maintained 50% longer in DCGA to maintain equal resolution of solution space with DHGA. The resolution of the search space is altered T_{max} (=4) times for DHGA to achieve the final finer resolution for finding optimum solution. In the experiment the values of other parameters of both GAs are maintained same as the mathematical function.

4.3. Experiment on various data sets

We have performed our test of both distributed GAs on two categories of data. One test is done on several unimodal or multi-modal mathematical functions for finding optimum functional value of each function. In the other test category we have considered pattern recognition problem for dot pattern matching and object matching with edge map.

4.3.1. Experiment with mathematical functions

Two sets of optimization functions in Tables 1 and 2 are considered to compare the performance between DHGA and DCGA in our experiment. Among them the first three functions f_1–f_3 in Table 1 are tested for maximization and the remaining four functions f_4–f_7 in Table 2 are taken for minimization. Two sets find the global optimum by DHGA and DCGA according to either Eqs. (7) or (8). The average results of 50 independent runs are summarized in Tables 3 and 4. Figs. 6–12 show the progress of the mean best solution and the mean of average values of population (Figs. 6–9) found by DHGA and DCGA over 50 runs for f_1–f_7. In Figs. 6–12 *Best of a method* indicates the mean of best solution of the population found by the corresponding process at each generation over 50 runs. Similarly, *Average of a method* in the graph represents the mean of average fitness values of the population at each iteration (over 50 runs) found by the corresponding process.

The first set of experiments was aimed to compare the convergence rate between DHGA and DCGA for functions f_1–f_4. Figs. 6–9 show the progress of the mean best solutions and the mean of average values of the population found by two GAs over 50 runs for f_1–f_4. It is apparent that DHGA performs better than DCGA in terms of convergence rate although DCGAs final results is identical to DHGAs for functions f_1, f_2 and f_4. It is observed that DHGA approaches the near global optimum value faster than the other process for the said functions, except f_3. Interestingly, it is also

Table 3
Comparison between DHGA and DCGA on f_1–f_4

Function	Number of generations	DHGA		DCGA	
		Mean best	Std. dev.	Mean best	Std. dev.
$f_1(\mathbf{x})$	400	299.7071	2.9×10^{-1}	299.7071	2.9×10^{-1}
$f_2(\mathbf{x})$	500	8.243607	5.6×10^{-2}	8.243607	5.6×10^{-2}
$f_3(\mathbf{x})$	200	5.00	0.00	4.9999	3.1×10^{-4}
$f_4(\mathbf{x})$	1500	−1.031628	0.00	−1.0316	3.0×10^{-7}

All results have been averaged over 50 runs. "Mean best" indicates mean best function values found in the last generation and "std. dev." stands for standard deviation.

Table 4
Comparison between DHGA and DCGA on f_5, f_6 and f_7

Function	Dimension (n)	Number of generations	DHGA		DCGA	
			Mean best	Std. dev.	Mean best	Std. dev.
$f_5(\mathbf{x})$	5	1000	0.00	0.00	0.00	0.00
	8	1000	0.00	0.00	5.5×10^{-4}	1.1×10^{-3}
	12	1000	0.00	0.00	7.5×10^{-3}	1.1×10^{-2}
$f_6(\mathbf{x})$	5	3000	2.9×10^{-3}	5.8×10^{-3}	3.3×10^{-2}	5.2×10^{-2}
	8	3000	1.0×10^{-2}	2.3×10^{-2}	9.0×10^{-2}	1.1×10^{-1}
	12	3000	3.4×10^{-3}	1.1×10^{-2}	1.1×10^{-1}	1.3×10^{-1}
$f_7(\mathbf{x})$	5	3000	0.00	0.00	6.0×10^{-6}	2.1×10^{-5}
	8	3000	1.0×10^{-6}	1.9×10^{-6}	2.1×10^{-5}	4.4×10^{-5}
	12	3000	7.7×10^{-5}	1.2×10^{-4}	2.26	2.48

All results have been averaged over 50 runs. "Mean best" indicates mean best function values found in the last generation and "std. dev." stands for standard deviation.

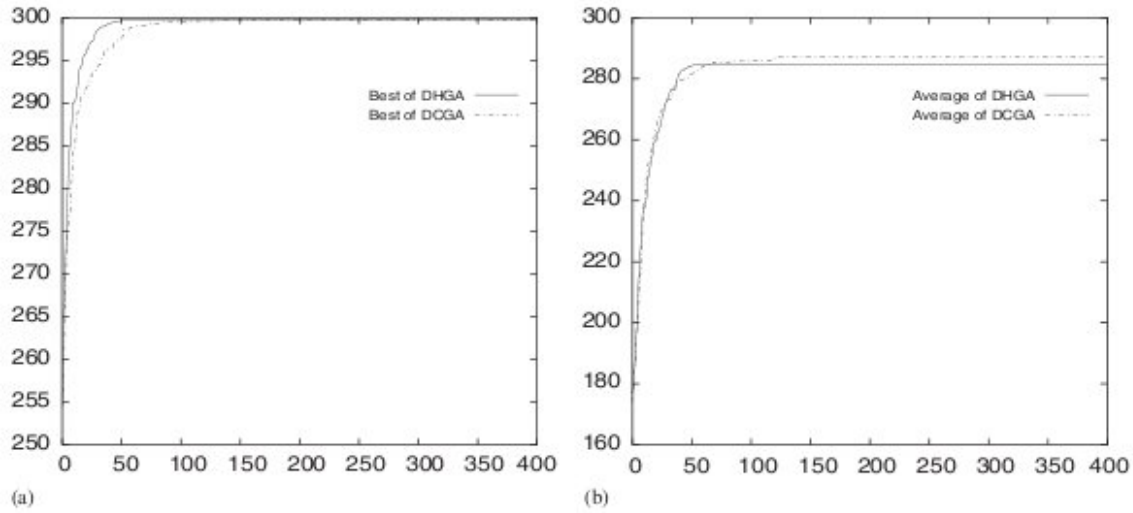

Fig. 6. Comparison between two distributed methods on f_1. The vertical axis is the function value and the horizontal axis is the number of generations. (a) shows the best results and (b) shows the average results. Both results are averaged over 50 runs.

observed that DCGA arrives earlier than DHGA to the vicinity of the global optimum for f_3, although only DHGA finally finds the global optimum. The mean of the average fitness values of a population in Figs. 6(b)–9(b) show the progress of the population for both distributed methods. It is noted that the overall population of DHGA is generally

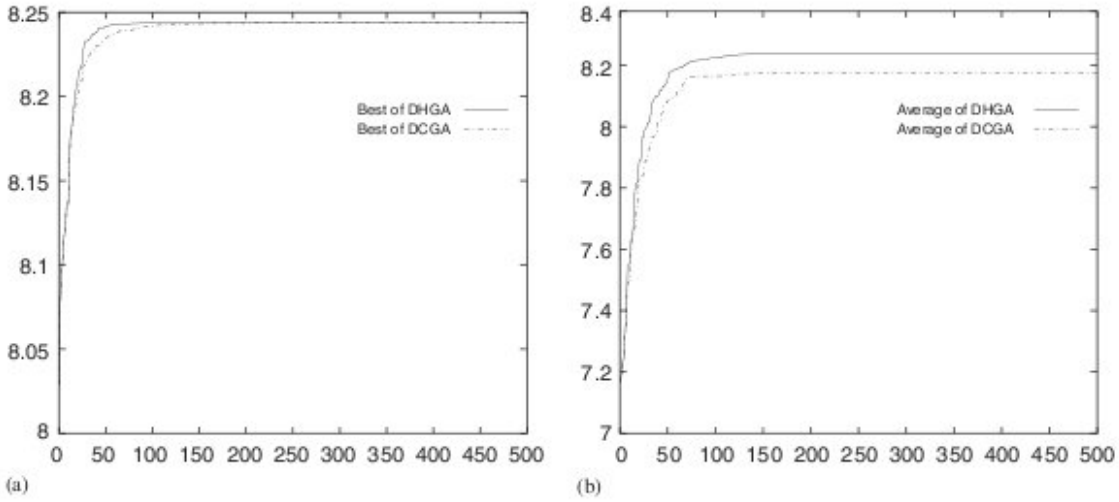

Fig. 7. Comparison between two distributed methods on f_2. The vertical axis is the function value and the horizontal axis is the number of generations. (a) shows the best results and (b) shows the average results. Both results are averaged over 50 runs.

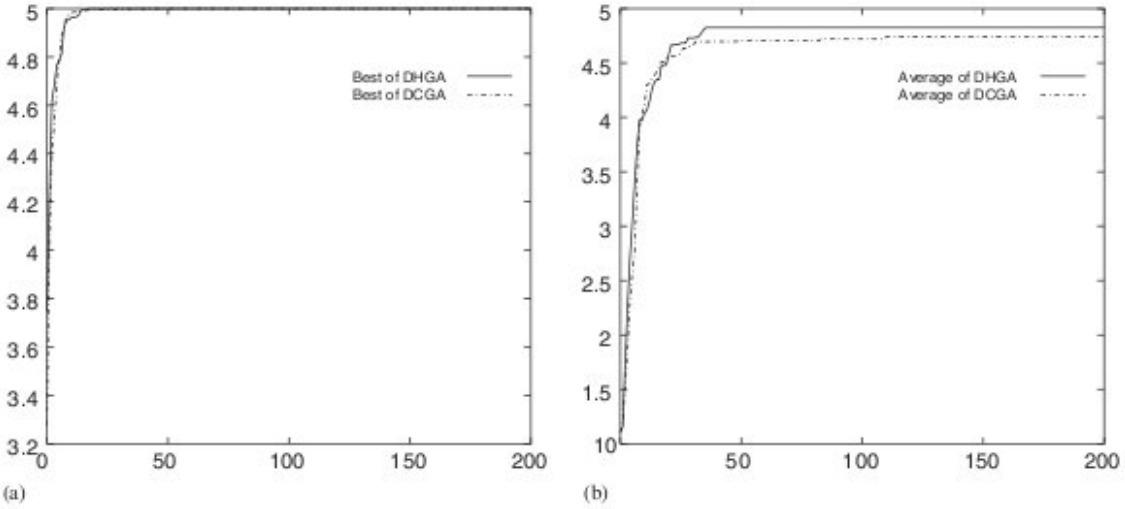

Fig. 8. Comparison between two distributed methods on f_3. The vertical axis is the function value and the horizontal axis is the number of generations. (a) shows the best results and (b) shows the average results. Both results are averaged over 50 runs.

better than DCGA except for f_1. However, the performance of DHGA is not much better than DCGA for the lower-dimensional functions except f_3.

The remaining functions f_5–f_7 in Table 2 are for higher-dimensional problems. Of these, f_5 is a unimodal function and Table 4 summarizes the performance of both DHGA and DCGA for three different dimensions $n = 5$, 8 and 12. The performance of DCGA is reduced with the increase in problem dimensionality. Fig. 10 shows that DHGA always converges faster than DCGA and reaches the global minimum value for all three dimensions of f_5. For $n = 5$, DCGA goes to the exact global minimum but it cannot perform so well for $n = 8$ and 12 (see Table 4).

We have added two other interesting and well-known functions f_6 and f_7 to verify the superiority of DHGA over DCGA. Both of them are high-dimensional multi-modal functions and they appear to be the most difficult class of problems for many optimization algorithms. Table 4 comprises of the final results of DHGA as well as DCGA averaged over 50 independent runs. For f_6 DHGA performs better than DCGA as illustrated in Fig. 11. It is observed in our experiment that both DHGAs and DCGAs performance come down with the increase of the problem dimensionality. However, DHGA outperforms DCGA in all three cases (for $n = 5$, 8 and 12) as shown in Fig. 11. Function f_7 is also tested for three different dimensions for finding whether or not the dimensionality of f_7 plays a vital role in restricting the GA to reach the global or near-global optimum. Interestingly, the superiority of DHGA persists over DCGA for all three problem dimensions. The performance of DCGA

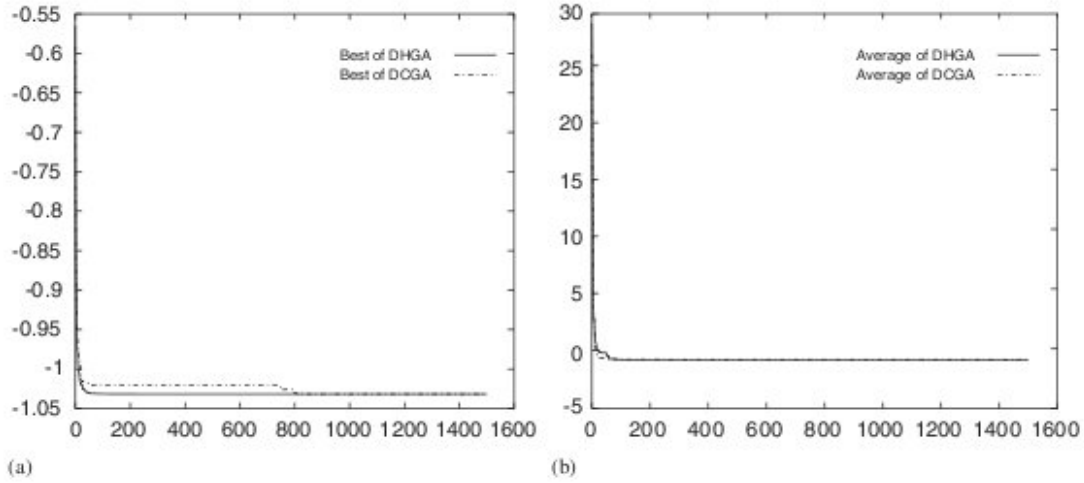

Fig. 9. Comparison between two distributed methods on f_4. The vertical axis is the function value and the horizontal axis is the number of generations. (a) shows the best results and (b) shows the average results. Both results are averaged over 50 runs.

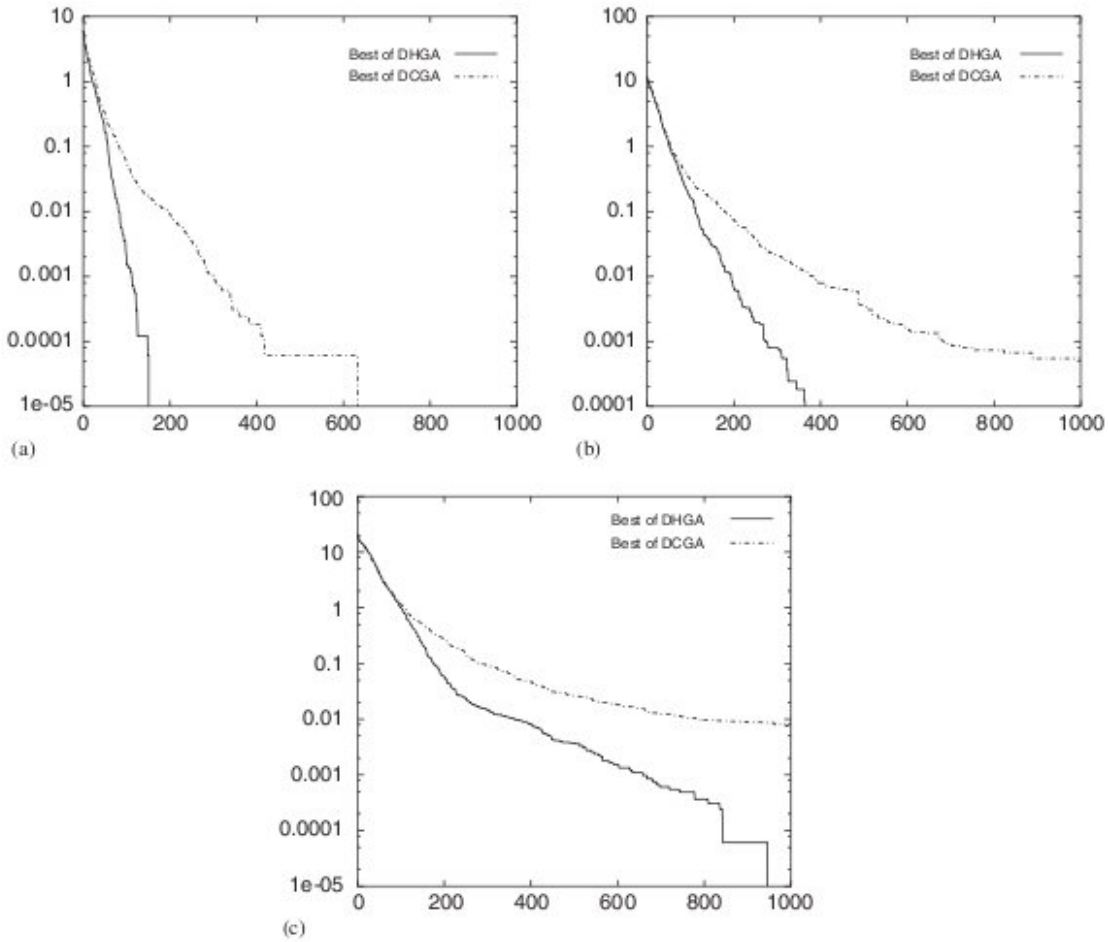

Fig. 10. Comparison between DHGA and DCGA on f_5 with three different problem dimensions. The vertical axis is the function value and the horizontal axis is the number of generations. (a) shows the best results for $n=5$, (b) shows the best results for $n=8$ and (c) shows the best results for $n=12$. All best results are averaged over 50 runs.

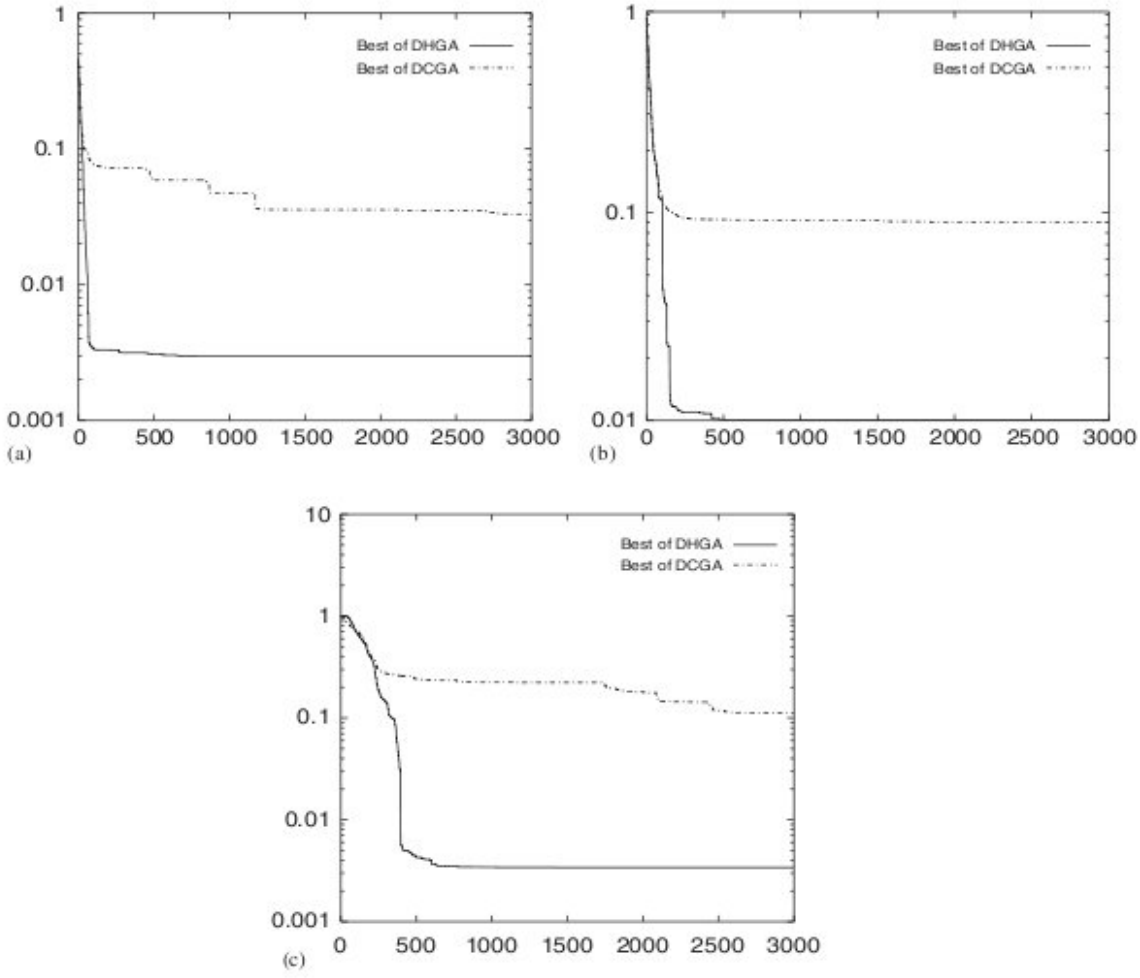

Fig. 11. Comparison between DHGA and DCGA on f_6 for three different dimensions. The vertical axis is the function value and the horizontal axis is the number of generations. (a) shows the best results for $n = 5$, (b) shows the best results for $n = 8$ and (c) shows the best results for $n = 12$. All best results are averaged over 50 runs.

deteriorates considerably for $n = 12$ compared to for $n = 5$ and 8. However, DHGA performs consistently better for f_7 compared to DCGA and reaches either the global optimum for $n = 5$ or the near-global optimum for $n = 8$ and 12 (see Table 5 and Fig. 12).

4.3.2. Pattern matching experiment

We have also conducted our experiment on two types of pattern recognition problems namely dot pattern matching and object matching with edge map. A dot pattern is a set of dots or points in two-dimensional or three-dimensional space arranged to represent some physical objects or class of objects in the feature space. Such dot patterns are encountered in astronomy and astrophysics, geographic and cartographic data, remote sensing, spatial information system, biomedical imaging, image texture analysis and some other disciplines of computer science [35–38]. The studies involving dot pattern includes shape identification and set estimation, classification and clustering, and point process parameter identification.

The dot pattern matching problem is described as follows. Given a test dot pattern T we have to identify it in an unknown scene of dot patterns S (a set of dot patterns or objects). So T should be translated and rotated to find the position of best match in S. To translate T, a reference point, say the centroid of the coordinates of the dots of T is used. Let this centroid be O. The translation of dot pattern to a point P in the space S means their centroid is translated to P and while its rotation by θ means rotation of the pattern(s) with O as origin.

The matching score of two dot patterns T and S are computed as follows. After transformation and rotation of T with respect to the origin O, the distance $d(t_i, s_j)$ between a point t_i of T and each of the points s_j of S is measured and the minimum distance is taken into consideration. If the number of points of T is α, then the sum of minimum distances D_{min} for all α points is as follows:

$$D_{min} = \sum_{i=1}^{\alpha} \mathrm{Min}[d(t_i, s_j)]_{j=1,2,\ldots,\beta}, \tag{9}$$

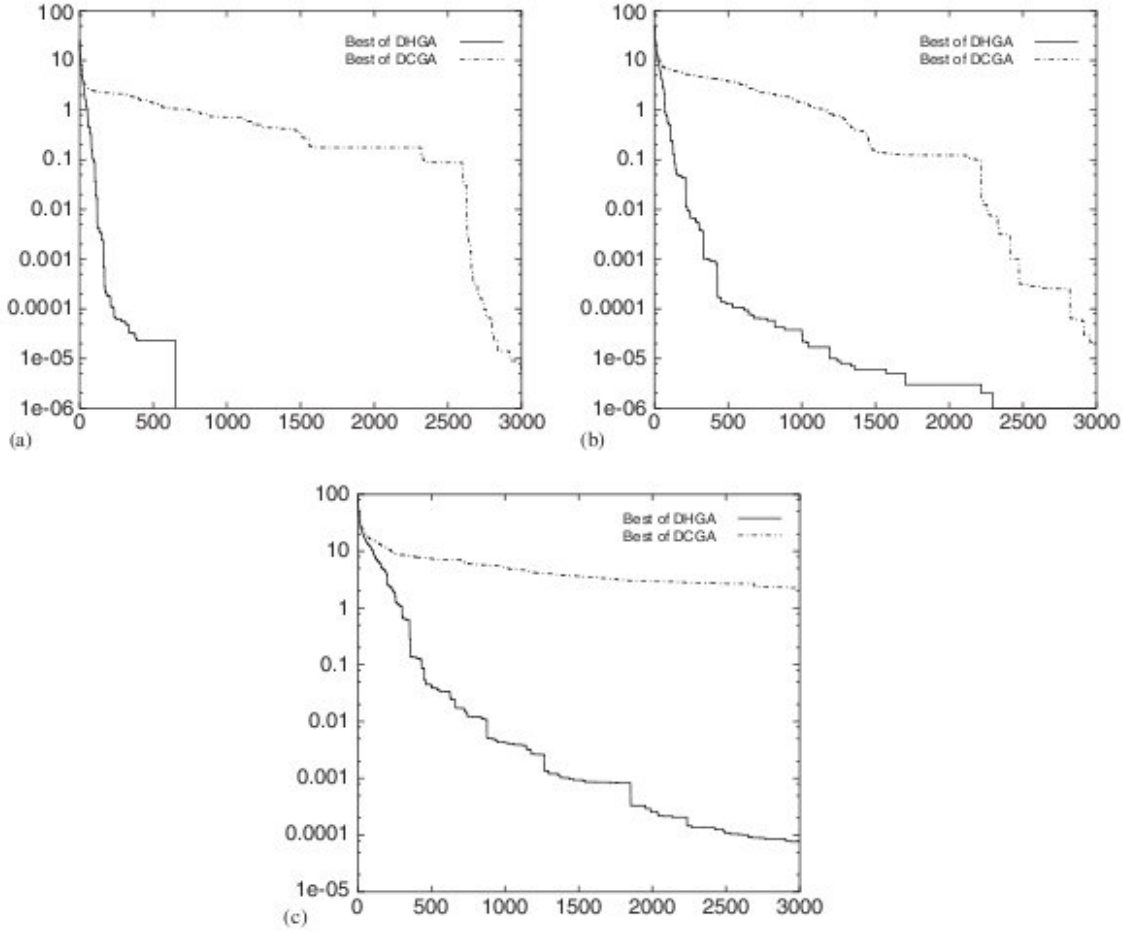

Fig. 12. Comparison between DHGA and DCGA on f_7 for three different problem dimensions. The vertical axis is the function value and the horizontal axis is the number of generations. (a) shows the best results for $n=5$, (b) shows the best results for $n=8$ and (c) shows the best results for $n=12$. All results are averaged over 50 runs.

where $t_1, t_2, \ldots, t_\alpha$ and $s_1, s_2, \ldots, s_\beta$ are the points of T and S, respectively, and $\alpha \leqslant \beta$. The value of D_{min} is the best matching score of two dot patterns or objects for a solution in a population.

In pattern recognition problem the performance of DHGA has been tested with the distributed approach of two GA based methods. One of them is DCGA and the second one is GA for affine point pattern matching (called APPMGA) [39]. We have slightly altered APPMGA to suit it with our pattern/object matching problem. The process is basically a sequential one which is converted to a distributed process so that the other two distributed approaches do not get any undue advantage. The APPMGA approach has considered the following two matrices for the affine transformation of the test data set:

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \quad \text{and} \quad [\, b_1 \quad b_2 \,]^{\mathrm{T}}.$$

The first matrix is a rotational matrix and the second one is a translation matrix. We have changed the elements of the rotational matrix according to the formulation of our proposed method as $a_{11} = \cos\theta$, $a_{12} = \sin\theta$, $a_{21} = -a_{12}$ and $a_{22} = a_{11}$ where θ is the rotational angle. Moreover, the fitness function of APPMGA is dependent on *Hausdroff distance*. However, we have used *Euclidean distance* in this experiment. In this case the distance of each point of one point set is evaluated from each point of another point set. As a result each point of both input point sets is taken into consideration during distance measurement which eventually helps to identify the decrease or increase of points in the input point sets.

In dot pattern or object matching problem S is a set of dot patterns of different shapes as shown in Figs. 13 and 14 and we have taken one of them as a test pattern T and matched it with S. The scores for matching between T and S for DHGA, DCGA and APPMGA are depicted in Tables 5 and 6. Here successful matching score is given as a ratio of the number of times the solution has been reached out to the total number of trials in percentage. We can consider two types of matching namely *perfect matching* and *imperfect* but *visual matching*. For *visually matched patterns* we see that T is superimposed over S without any visible error, although the

Table 5
Matching results of DHGA, DCGA and APPMGA on dot patterns

Dot pattern	DHGA			DCGA			APPMGA		
	Successful matching (%)		Unsuccessful matching (%)	Successful matching (%)		Unsuccessful matching (%)	Successful matching (%)		Unsuccessful matching (%)
	Perfectly matched	Visually matched	Not matched	Perfectly matched	Visually matched	Not matched	Perfectly matched	Visually matched	Not matched
DP1	57	43	0	14	73	13	15	75	10
DP2	40	60	0	10	70	20	10	75	15
DP3	83	17	0	10	67	23	10	70	20
DP4	40	60	0	10	90	0	10	90	0

The following data have been summarized over 50 runs for each DP. In each run the space is redefined 4 times.

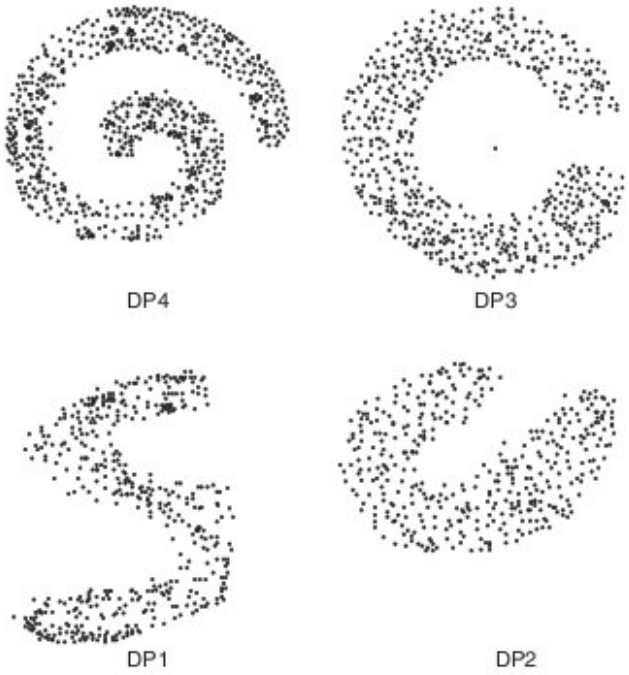

Fig. 13. A scene of multiple dot patterns in two-dimensional.

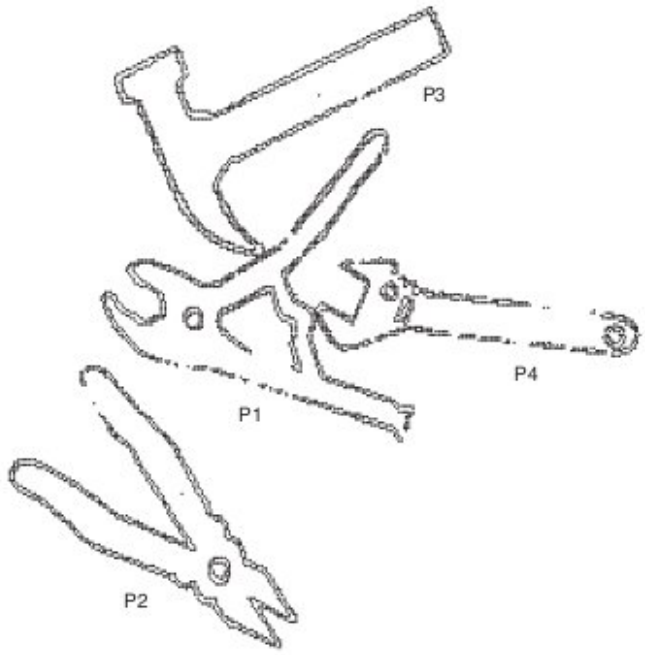

Fig. 14. A scene of multiple objects with edge map in two-dimensional.

matching error is computationally reasonable. On the other hand, in *perfectly matched situation* the computed error of matching between T and S is very close to zero. Naturally, the pattern or object is visually matched.

From Table 5 we notice that for DHGA the success rate is 100% for all four patterns by perfect and visual matching. The best performance of DHGA is achieved for DP3. On the other hand, the failure rate is at most 23% in worst case for DCGA. However, it can also achieve a success rate of 100% for matching the pattern DP4. The performance of APPMGA is almost equivalent to DCGA.

The other pattern recognition experiment done here is matching of edge maps. Here the object scene is a combination of four different objects. Initially the scene is a *gray-tone* digital image in the space of 512×512 pixels with 256 possible gray levels. It is then converted into a two-tone edge map using *Sobel Operator* [40]. The edge map may be considered as a discrete dot pattern where a dot is represented by 1 and a blank (white space) is by 0 (see Fig. 14). In the scene three objects (P1, P3 and P4) are very close and touched each other. The other one (i.e., P2) is separate.

The experimental results of object matching is tabulated in Table 6 for the approaches based on GA. It is observed that the success rate of DHGA is not always 100% as happened for dot pattern matching problem. In case of P1 and P2 the success of DHGA is 100% and its failure rate is maximum 6% for matching P3 and P4. The performance of all three techniques is best for matching P2 which is an isolated object in the scene. It is noted that DCGAs performance deteriorates when the objects are close to each other. In object matching situation the APPMGA performs similar to DCGA.

Table 6
Matching results of DHGA, DCGA and APPMGA on objects with edge map

Dot pattern	DHGA			DCGA			APPMGA		
	Successful matching (%)		Unsuccessful matching (%)	Successful matching (%)		Unsuccessful matching (%)	Successful matching (%)		Unsuccessful matching (%)
	Perfectly matched	Visually matched	Not matched	Perfectly matched	Visually matched	Not matched	Perfectly matched	Visually matched	Not matched
P1	33	67	0	0	30	70	0	35	65
P2	76	24	0	10	63	27	10	65	25
P3	27	67	6	3	10	87	5	10	85
P4	17	80	3	0	27	73	0	30	70

The following data have been summarized over 50 runs for each DP. In each run the space is redefined 4 times.

5. Conclusion

We have proposed DHGA to enhance the performance of GA and compared its performance with DCGA. Here the genetic search starts with small chromosome size. The method is a hybrid of distributed as well as hierarchical techniques. The search is first scattered over the entire space for uniform population distribution and simultaneously in each subspace the sequential GA is progressed hierarchically in parallel with coarse-to-fine resolution. Initially, the small chromosome or the string length of an individual $\mathbf{x}_i, \forall i \in \{1, \ldots, \mu\}$ represents the search space with coarse resolution. In successive stages, however, since the search space dimension is reduced (see Fig. 1) and the string length remains constant thereafter eventually increases the resolution of the search space.

DHGA and DCGA deal with multiple populations. The entire space is initially partitioned into a number of equal sized smaller subspaces like Fig. 3. In each subspace s_i, $\forall i \in \{1, \ldots, m\}$ the distributed GAs run simultaneously starting with an independent population. The process starts searching over the entire space and runs concurrently in m subspaces. Thus, DHGA reduces the possibility of the algorithm to be trapped in a local optimum of the optimizing function and is faster in terms of convergence rate of the population compared to DCGA.

We have demonstrated our approach on various optimizing functions. A set of functions f_1 to f_3 are used to find the global maximum value and the other set consisting of f_4 to f_7 are used to locate the global minimum value. The nature of the functions differ widely from one another. Some of them are unimodal whereas others are multi-modal functions. Moreover, two complicated functions f_6 and f_7 are included in our experiment for which the number of local optima increases exponentially with the problem dimensionality. In f_6 and f_7 the performance of DHGA as well as DCGA deteriorates with the increase of the problem dimension, although DHGA always outperforms DCGA (see Figs. 11 and 12). For the remaining functions (f_1–f_4) the performance is comparable with DCGA.

Next, we have taken the matching problem between two dot patterns or objects with edge map. The match between two dot patterns/objects is obtained by first translating the test pattern to a point P and then rotating by an angle θ with respect to the mean point O as origin. In dot pattern matching DHGA always outperforms DCGA and APPMGA for all dot patterns with success rate 100% except for DP4.

In object matching with edge map a digital image with 256 possible gray levels in a space of 512×512 pixels is a combination of four separate objects which are very close and touching each other. It is then converted to a two-tone edge map using *Sobel Operator* [40] and transformed to a dot pattern where the edge is indicated by a series of dots. The matching between two objects is done in a manner similar to the dot pattern matching problem. In this experiment DHGA performs better than DCGA as well as APPMGA but it cannot always achieve a success rate of 100% except for objects P1 and P2. Similarly, the performance of DCGA and APPMGA are best among all four results for the isolated object P2 in the scene.

Acknowledgments

The authors would like to thank Mr. Rahul Banerjee of C&MB Division and Mr. N. Sanpui of Computer Division, SINP for their help. One of the authors would like to thank Jawaharlal Nehru Memorial Fund for providing support in the form of a Jawaharlal Nehru Fellowship.

References

[1] J. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, MI, 1975.
[2] S.F. Smith, A learning system based on genetic algorithms, Ph.D. Dissertation, University of Pittsburgh, PA, 1980.
[3] S.A. Harp, T. Samad, Genetic synthesis of neural network architecture, in: L. Davis (Ed.), Handbook of Genetic algorithms, New York, 1992, pp. 202–221.

[4] D.E. Goldberg, K. Deb, B. Korb, Messy genetic algorithms: motivation, analysis and first results, Complex Systems 3 (1989) 493–530.
[5] D. Kim, S. Ahu, A MS-GS VQ codebook design for wireless image communication using genetic algorithms, IEEE Trans. Evol. Comput. 3 (1999) 35–52.
[6] D.J. Cavicchio, Adaptive Search using simulated evolution, Ph.D. Dissertation, University of Michigan, Ann Arbor, 1970.
[7] K.A. De Jong, An analysis of behavior of a class of genetic adaptive system, Doctoral Dissertation, University of Michigan, 1975.
[8] D.E. Goldberg, J. Richardson, Genetic algorithms with sharing for multimodal function optimization, Proceedings of the Second International Conference on Genetic Algorithms, 1987, pp. 44–49.
[9] L. Eshelman, The CHC adaptive search algorithm. How to have safe search when engaging in nontraditional genetic recombination, in: G. Rawlins (Ed.), FOGA-I, Morgan Kaufmann, 1991, pp. 256–283.
[10] D.B. Fogel, Evolutionary Computation: Toward a New Philosophy of Machine Intelligence, IEEE Press, Piscataway, NJ, 1995.
[11] T. Back, H.P. Schwefel, An overview of evolutionary algorithm for parameter optimization, Evol. Comput. 1 (1993) 1–23.
[12] E. Cantú-Paz, A summary of research on parallel genetic algorithms, Technical Report 950076, Illinois Genetic Algorithm Laboratory, University of Illinois Urbana-Champaign, Urbana, IL, July 1995.
[13] M. Tomassini, Parallel and distributed evolutionary algorithms: a review, in: K. Miettinen, M. Mkel, P. Neittaanmki, J. Periaux (Eds.), Evolutionary Algorithms in Engineering and Computer Science, Wiley, New York, 1999, pp. 113–133.
[14] J.K. Hao, R. Dome, A new population-based method for satisfiability problems, in: Proceedings of the 11th European Conference on Artificial Intelligence, Wiley, New York, 1994, pp. 135–139.
[15] J.H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, MI, 1975.
[16] J. Lienig, A parallel genetic algorithm for performance-driven VLSI routing, IEEE Trans. Evol. Comput. 1 (1997) 29–39.
[17] G.A. Sena, D. Megherlu, G. Isern, Implementation of a parallel genetic algorithm on a cluster of workstations: traveling salesman problem, a case study, Future Generation Computer Systems 17 (2001) 477–488.
[18] F.F. Easton, N. Mansour, A distributed genetic algorithm for deterministic and stochastic labor scheduling problems, Eur. J. Oper. Res. 118 (1999) 505–523.
[19] D.A.V. Veldhuizen, J.B. Zydallis, G.B. Lamont, Considerations in engineering parallel multiobjective evolutionary algorithms, IEEE Trans. Evol. Comput. 7 (2002) 144–173.
[20] A. Hill, C.J. Taylor, Model-based image interpretation using genetic algorithm, Image Vision Comput. 10 (1992) 295–300.
[21] G. Roth, M.D. Levine, Geometric primitive extraction using a genetic algorithm, IEEE Trans. Pattern Anal. Mach. Intell. 16 (1994) 901–905.
[22] A. Toet, W.P. Hajema, Genetic contour matching, Pattern Recognition Lett. 16 (1995) 849–856.
[23] D.S. Lin, J.J. Leou, A genetic algorithm approach to Chinese handwriting normalization, IEEE Trans. Syst. Man Cybern.—Part B 27 (1997) 999–1007.
[24] S.M. Yamany, K.J. Khiani, A.A. Farag, Application of neural networks and genetic algorithms in the classification of endothelial cells, Pattern Recognition Lett. 18 (1997) 1205–1210.
[25] G. Giacinto, P. Paolucci, F. Roli, Application of neural networks and statistical pattern recognition algorithms to earthquake risk evaluation, Pattern Recognition Lett. 18 (1997) 1353–1362.
[26] Y.K. Wang, K.C. Fan, J.T. Horng, Genetic-based search for error-correcting graph isomorphism, IEEE Trans. Syst. Man Cybern.—Part B 27 (1997) 588–596.
[27] Nirwan Ansari, M.H. Chen, E.S.H. Hou, Point pattern matching by a genetic algorithm, in: Proceedings of the 16th Annual Conference of IEEE Industrial Electronic Society (IECON'90), vol. II, Pacific Grove, 1990, pp. 1233–1238.
[28] M. Mirmehdi, P.L. Palmer, J. Kittler, Genetic optimization of the image feature extraction process, Pattern Recognition Lett. 18 (1997) 355–365.
[29] G. Garai, B.B. Chaudhuri, A cascaded genetic algorithm for efficient optimization and pattern matching, Image Vision Comput. 20 (2002) 265–277.
[30] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, New York, 1989.
[31] N.N. Schrandolph, R.K. Belew, Dynamic parameter encoding for genetic algorithms, Mach. Learn. 9 (1992) 9–21.
[32] D.B. Fogel, System Identification Through Simulated Evolution: A Machine Learning Approach to Modeling, Ginn, Needham Heights, MA, 1991.
[33] A. Torn, A. Zilinskas, Global optimization, Lecture Notes in Computer Science, vol. 350, Springer, Berlin, Germany, 1989.
[34] H.P. Schwefel, Evolution and Optimization Seeking, Wiley, New York, 1995.
[35] P.M. Griffin, C. Alexopoulos, Point pattern matching using centroid bounding, IEEE Trans. Syst. Man Cybern. 19 (5) (1989) 1274–1276.
[36] D. Lavine, B.A. Lambird, L.N. Kanal, Recognition of spatial point patterns, Pattern Recognition 16 (3) (1983) 289–295.
[37] J. Sprinzak, M. Werman, Affine point matching, Pattern Recognition Lett. 15 (4) (1994) 337–339.
[38] L. Zhang, W. Xu, Point-pattern matching using irreducible matrix and relative invariant, Tsinghua Sci. Technol. 4 (4) (1999) 1602–1605.
[39] L. Zhang, W. Xu, Genetic algorithm for point pattern matching, Pattern Recognition Lett. 24 (2003) 9–19.
[40] A.K. Jain, Fundamentals of Digital Image Processing, Prentice-Hall, Englewood Cliffs, 1989.

About the author—GAUTAM GARAI received the B.Tech. degree from Calcutta University in 1987 and the M.Tech. degree from Jadavpur University in 1991. He worked as an Assistant Engineer in M.N. Dastur, Kolkata from 1987 to 1988. He is currently a Scientist in Computer Division, Saha Institute of Nuclear Physics. His research interests include Image Processing, Pattern Matching, Evolutionary Algorithms, Data Mining and Optimization.

About the author—PROF. B.B. CHAUDHURI received his B.Sc. (Hons), B.Tech. and M.Tech. degrees from Calcutta University, India in 1969, 1972 and 1974, respectively and Ph.D. Degree from Indian Institute of Technology, Kanpur in 1980. He joined Indian Statistical Institute in 1978 where he served as the Project Coordinator and Head of National Nodal Center for Knowledge Based Computing funded by DoE and United Nations Development Program. Currently, he is the Head of Computer Vision and Pattern Recognition Unit of the Institute. His research interests include Pattern Recognition, Image Processing, Computer Vision, Natural Language Processing (NLP), Speech Processing, Digital Document Processing and OCR. He pioneered the first Indian language Bharati Braille system for the blind as well as developed the first workable OCR for Bangla, Devnagari, Assamese and Oriya scripts. Also, he developed a successful Bangla speech synthesis system with various applications. In NLP area, robust spell-checker, powerful morphological processor, multi-word expression parser and statistical analyzer were pioneered by him. The OCR technologies have been transferred to several organizations like CDAC, Pune and Noida, IIT Guwahati and WEBEL Kolkata for commercialization. He had also developed some software for Bio-medical Image Processing that has been used in pathological institutes. He has published about 300 research papers in reputed international journals, conference proceedings and edited books. Also, he has authored three books entitled *Two Tone Image Processing and Recognition* (Wiley Eastern, 1993), *Object Oriented Programming: Fundamentals and Applications* (Prentice Hall, 1998) and *Computer and Software Technology Dictionary*

(Ananda Publishers, 2002). He was awarded Sir J. C. Bose Memorial Award for best engineering science oriented paper in 1986, M.N. Saha Memorial Award (twice) for best application oriented paper in 1989 and 1991, Homi Bhabha Fellowship award in 1992 for OCR of the Indian Languages and computer communication for the blind, Dr. Vikram Sarabhai Research Award in 1995 for his outstanding achievements in the fields of Electronics, Informatics and Telematics, C. Achuta Menon Prize in 1996 for computer-based Indian Language Processing, Homi Bhabha Award: Applied Sciences in 2003 and Ram Lal Wadhwa Gold Medal award in 2005. Also, he has been chosen for the prestigious Jawaharlal Nehru Fellowship to conduct research on Document processing during 2004–2006. He has worked as a Leverhulme visiting fellow at Queen's University, UK in 1981–82, as a visiting scientist at GSF, Munich, and as a guest faculty at the Technical University of Hannover during 1986–88 and 1990–91. He is a member secretary (Indian Section) of International Academy of Sciences, Fellow of National Academy of Sciences (India), Fellow of Institution of Electronics and Telecommunication Engineering, and Fellow of the Indian National Academy of Engineering. In 1998 and 2000, respectively, he was elected as a Fellow of International Association of Pattern Recognition (IAPR) and a fellow of IEEE (Institute of Electrical and Electronics Engineers). Presently, he is serving as an associate editor of *Pattern Recognition, VIVEK, International Journal of Pattern Recognition and Artificial Intelligence, International Journal of Computer Vision and International Journal of Document Analysis and Recognition.* He has served as a guest editor of special issue of several journals.