Information Sciences 221 (2013) 28–48

Contents lists available at SciVerse ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

A novel hybrid genetic algorithm with Tabu search for optimizing multi-dimensional functions and point pattern recognition

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

[a] Computational Science Division, Saha Institute of Nuclear Physics, 1/AF Bidhannagar, Kolkata 700 064, India
[b] Computer Vision & Pattern Recognition Unit, Indian Statistical Institute, 203 B.T. Road, Kolkata 700 108, India

ARTICLE INFO

Article history:
Received 29 November 2011
Received in revised form 27 August 2012
Accepted 14 September 2012
Available online 25 September 2012

Keywords:
Hybrid optimization approach
Memetic algorithm
Genetic algorithm
Tabu search
Grid pattern matching
Point pattern recognition

ABSTRACT

Hybrid evolutionary algorithms are drawing significant attention in recent time for solving numerous real world problems. This paper presents a new hybrid evolutionary approach for optimizing mathematical functions and Point Pattern Recognition (PPR) problems. The proposed method combines a global search genetic algorithm in a course-to-fine resolution space with a local (Tabu) search algorithm. Such hybridization enhances the power of the search technique by virtue of inducing hill climbing and fast searching capabilities of Tabu search process. The approach can reach the global or near-global optimum for the functions in high dimensional space. Tests have been successfully made on several benchmark functions in up-to 100 dimensions. The performance of the proposed algorithm has been compared with other relevant algorithms using non-parametric statistical approaches like Friedman test, multiple sign-test and contrast estimation. Also, the hybrid method with grid based PPR technique has been applied for solving dot pattern shape matching and object matching represented as edge maps. The performance of proposed method compares favorably with relevant approaches reported in the article.

1. Introduction

Evolutionary Algorithms (EAs) are a class of search and optimization techniques that work on the principle inspired by Darwinian Evolution. EAs are used in solving a wide variety of combinatorial optimization problems in diverse fields as machine vision, astronautics, document processing, biometrics, computational biology and computational chemistry [1,2,19,26].

In recent years the hybrid genetic algorithms have gained significant interest and are being increasingly used in solving various real world problems. It is now well known that *pure* evolutionary algorithms are not suited for a fine tuned search in complex combinatorial spaces. Instead hybridization with other techniques can improve the efficiency of the search process [11]. The combination of *Evolutionary Algorithm* with *Local Search* approach is known as "Memetic" or "Hybrid" algorithm. Memetic or hybrid algorithms are extensions of evolutionary algorithms which apply separate processes of hill climbing to refine individual chromosome.

In recent past, we proposed a CAscaded Genetic Algorithm (CAGA) for optimization problems [7] which is a *coarse-to-fine* search method. The process starts in low resolution over the entire space and in subsequent stages the space is reduced and the resolution is increased. Recently we have upgraded this algorithm by introducing varying degree of search space reduction so that the global solution can be better attained. This approach is described in Section 2.3. However, there have been

* Corresponding author. Fax: +91 33 23374637.
E-mail address: gautam.garai@saha.ac.in (G. Garai).

0020-0255/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ins.2012.09.012

occasions where the global optimum has been missed by CAGA. In this paper CAGA with Tabu search has been explored as a remedy to this problem.

The hybridization overcomes the limitation of the *coarse-to-fine* search based genetic algorithm by reducing the possibility of the search process getting trapped in local optima. Since the search (*coarse-to-fine*) process jumps from one hypercube to another in a multi-dimensional space, there is a possibility of missing the global solution in this movement. The inclusion of a local search process with the Genetic Algorithm (GA) helps to search in the neighborhood of the solution (when the solution is converged in a hypercube in an intermediate stage) when the GA jumps to the new hypercube (see Figs. 1 and 2 and description in Section 2.3). Thus, the hybridization process largely (but not entirely) reduces the possibility of missing the global solution. As a result, the proposed method can work on high dimensional spaces. The user also has the option to control the rate of reduction of search space. The reduction in slower pace may be necessary to reach the global optimum solution. Tests have been made on the benchmark functions in up-to 100 dimensions as described in Section 4.2.

The effective use of genetic algorithm in point pattern recognition is reported in various literatures [10,11,26]. Given two sets of points in d-dimensional space, one has to determine whether there is a transformation that maps the first set onto or close to the second set of points. In general, pattern matching can be of two categories namely, *complete matching* and *approximate matching*. Complete matching usually occurs in an idealistic situation while in practice there exist spurious or lost points due to image degradation and binarization error, so that exact match is not possible. Depending on additional information of an image (such as color, intensity etc.) other than pixel coordinates, the point pattern recognition can be done on labeled points or unlabeled points [26]. The unlabeled point matching is more difficult than its labeled counterpart but such a situation is encountered more often.

Among other hybrid approaches, in [20] the GA is combined with two neural networks namely, a feature extraction network and a neural classifier. The hybrid GA is used to select the receptive field parameters to improve the classification performance and is applied for handwritten digit classification and face recognition. Another genetic approach hybridized with fuzzy logic has been proposed by Ishibuchi et al. [14] for designing fuzzy rule based systems for pattern classification. Also, the memetic evolutionary method with Tabu search has been used for cell image segmentation [16].

Among the different approaches proposed for point pattern matching, a heuristic algorithm has been used by Denton and Beveridge [4] to minimize the effect of spurious points for matching two point patterns. A meta-heuristic particle swarm optimization algorithm [25] has also been proposed in the recent past. Zhang et al. [26] employed genetic algorithm for point pattern recognition. They used the reference triplet points as the chromosome representation in order to reduce the search space significantly. In a noise-free environment Caetano et al. [1] proposed point pattern matching as a weighted graph matching problem where the weights correspond to Euclidean distance between nodes. Carcassoni and Hancock [2] have applied the spectral graph theory to compute the point pattern correspondence. Li et al. [19] used the point pattern matching method in palm pattern recognition. The matching is performed in two-phases based on the local structure of the minutiaes and the global feature.

Usually, GAs require a large number of iterations in order to converge to a solution. In pattern recognition or similar problems, generally one starts with chromosome length dependent on the accuracy of the required solution. The length then

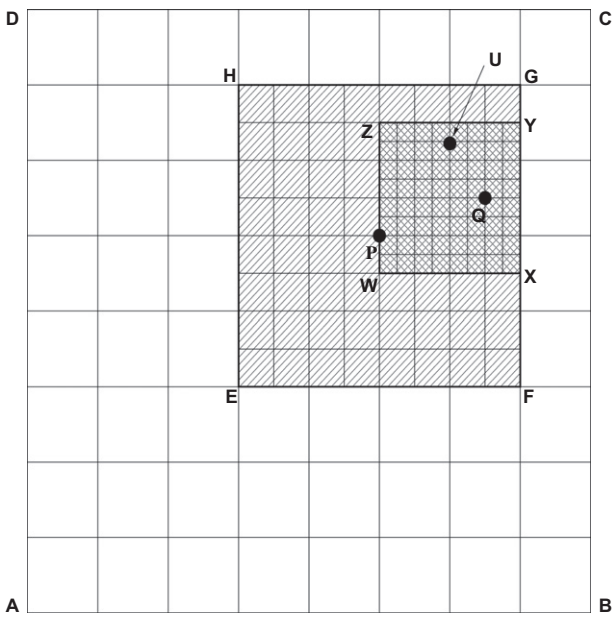

Fig. 1. Transfer of GA search in three stages in 2-D space with fast partitioning process.

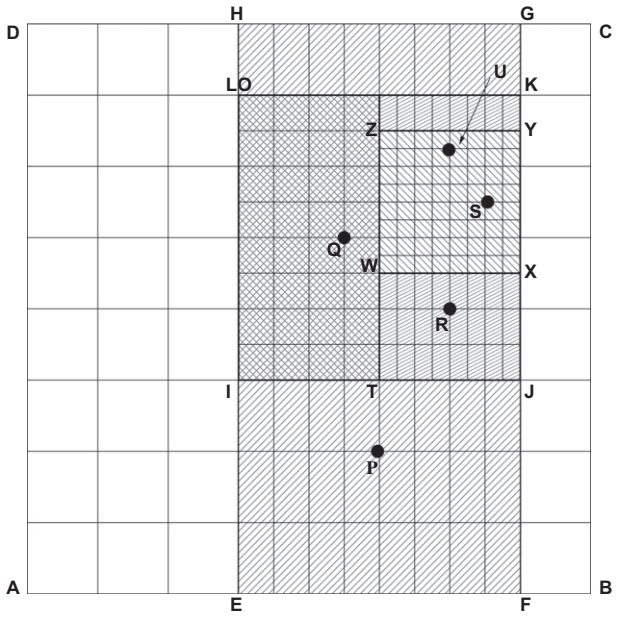

Fig. 2. Transfer of GA search in three stages in 2-D space with slow partitioning process.

remains constant throughout the execution of the algorithm. Thus, the chromosome length will be increased with the increase in accuracy of the required solution, which eventually will increase the execution time of the GA. The benefit of the proposed approach is the improvement in the efficiency of the system by hybridization of a local search technique with *coarse-to-fine* search strategy based GA. Such combination can work with smaller chromosome length and fast convergence to the solution.

In this paper our contribution has two parts. The first one is a cascaded hybrid genetic approach with Tabu search and the second is a grid based point pattern matching method where the match is optimized by the proposed hybrid genetic algorithm. The performance of the Hybrid CAscaded Genetic Algorithm (H-CAGA) is tested on *twenty* benchmark functions. They are used to compare the optimization capability of the hybrid method with several relevant genetic algorithms in 30 dimensional space. The performance of H-CAGA is also evaluated for the functions in higher (100) dimensional space with fast and slow reduction of search space. We also performed the set of non-parametric procedures (like Friedman test, multiple sign-test, contrast estimation) for doing statistical comparison between H-CAGA and the relevant algorithms considered in the experiment. In solving pattern recognition problems, we have partitioned the pattern space with a grid structure. The proposed grid based Point pattern Recognition (PPR) method with H-CAGA is named as Grid based Optimized Pattern Matching (GOPM) technique. The performance of GOPM is also evaluated on different pattern sets and compared with related pattern matching algorithms.

The rest of the paper is organized as follows. The next section describes different optimization techniques presented in this paper. Section 3 depicts the hybridization schemes of the sequential optimization methods discussed in Section 2. Analysis of empirical results of H-CAGA on several standard unimodal and multimodal functions [23,24] which are compared with the results of other relevant algorithms is included in Section 4. The performance of various hybrid genetic algorithms based pattern recognition is also elaborated in this section. The paper is concluded in Section 5.

2. Optimization techniques

In this section we present the global and the local optimization algorithms used in the hybridization process.

2.1. Conventional genetic algorithm

The binary coded Conventional GA (CGA) with *one-point crossover* is described below.

Step 1. Generate randomly the initial population of N individuals (each is a bit string chosen from $\{0,1\}$) and let generation $g = 1$. Initialize p_c as the crossover probability, p_m as the mutation probability and G_{max} as the maximum number of generation/iterations.

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

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 N individuals.

Step 4. Conduct crossover between the chosen individuals $\mathbf{x}_\alpha$ and $\mathbf{x}_\beta$ with p_c and mutate each of their bits with mutation probability p_m. Each pair of parents $(\mathbf{x}_\alpha, \mathbf{x}_\beta)$ thus creates a pair of new individuals called offsprings $\left(\mathbf{x}'_\alpha, \mathbf{x}'_\beta\right)$ to generate a pool of individuals, $\mathbf{x}'_j,\ \forall j \in \{1,\ldots,N\}$ 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.

2.2. Tabu Search (TS) method

TS is a meta heuristic local search that can be used to solve combinatorial optimization problems. Compared to hill climbing local search techniques, it is less likely to get trapped into a local optimal solution [9]. Also, TS has a higher execution speed than GA since it (TS) does not revisit the already explored solutions, considering them *taboo*. This is possible because each move in TS is recorded to prevent revisit to the old solution.

The iterative search procedure starts with a set of probable or feasible solutions. Each solution is a string of bits, chosen from {1,0}, which is called the array. Let $\mathbf{A}_t$, $\mathbf{A}_c$ and $\mathbf{A}_b$ denote the trial, current and best array(s) and O_t, O_c and O_b be the corresponding trial, current and best objective function value(s), respectively. The process assigns the current solution $\mathbf{A}_c$ for starting its operation. Then the trial solutions $\mathbf{A}_t$s are generated through some moves. For each iteration a best solution A_b is found. With the progress of the search process if it is found that the best solution is in the forbidden or tabu list but satisfies the *aspiration criteria*, then it is considered to be the new current solution. An aspiration criterion is a condition when a tabu move has an attractive evaluation where it results a better solution than any visited so far, then its tabu classification is overridden. The TS algorithm is stated by the following steps.

Step 1: Initialize the parameters MTLS (Maximum Tabu List Size), the number of trial solutions λ, the maximum number of iterations I_{max}. Let $\mathbf{A}_c$ be an arbitrary solution and O_c be the corresponding objective function value. Initially, let $\mathbf{A}_b = \mathbf{A}_c$, $O_b = O_c$, TLL (Tabu List Length) = 0 and iteration $I = 1$.

Step 2: Using $\mathbf{A}_c$, generate λ trial solutions $\mathbf{A}_t^1, \mathbf{A}_t^2, \ldots, \mathbf{A}_t^\lambda$ and evaluate their corresponding objective function values $O_t^1, O_t^2, \ldots, O_t^\lambda$. Given a current solution $\mathbf{A}_c$, one can generate a trial solution using several strategies. In our case, given $\mathbf{A}_c$, we have flipped a bit of $\mathbf{A}_c$ if the probability threshold is higher than a randomly generated value between 0 and 1. Otherwise, the corresponding bit is kept unchanged.

Step 3: Arrange the objective function values $O_t^1, O_t^2, \ldots, O_t^\lambda$ in ascending order and denote them as $O_t^{1'}, O_t^{2'}, \ldots, O_t^{\lambda'}$. If $O_t^{1'}$ is not tabu or if it is tabu but $O_t^{1'} < O_b$ (in case of minimization) then make $\mathbf{A}_c = \mathbf{A}_t^{1'}$ and $O_c = O_t^{1'}$. Next, go to step 4. Otherwise, let $\mathbf{A}_c = \mathbf{A}_t^L$ and $O_c = O_t^L$ where O_t^L is the best objective function values of $O_t^{1'}, O_t^{2'}, \ldots, O_t^{\lambda'}$ that is not tabu and go to step 4. If $O_t^{1'}, \ldots, O_t^{2'}, O_t^{\lambda'}$ are all tabu, then go to step 2.

Step 4: Insert A_c at the bottom of the tabu list and increment TLL by 1. If TLL = MTLS + 1 then delete the first element in the list and TLL = TLL − 1. If $O_b > O_c$ then $\mathbf{A}_b = \mathbf{A}_c$ and $O_b = O_c$. Terminate the process if $I = I_{max}$ with $\mathbf{A}_b$ as the best solution and O_b as the corresponding best objective function. Otherwise, make $I = I + 1$ and go to step 2.

2.3. CAGA in multi-resolution search space

In the multi-dimensional data space the CAGA [7] initiates a repetitive stochastic search process that goes from coarse-to-fine resolution, with all three basic operations of the conventional GA namely, selection/reproduction, crossover and mutation.

2.3.1. Increase of search space resolution in stages

For simplicity, consider a 2-D space represented by the area ABCD (see Fig. 1) and the solution chromosome length be 6 bits where 3 bits are used to represent the x-coordinate and 3 bits for the y-coordinate. The GA then begins its search over the entire space ABCD and finds the best solution $\mathbf{x}_B^1$ (shown by the point P in Fig. 1) at the first stage. At stage 2, the process shifts its attention to the reduced search space EFGH around P (shown by a different shade in Fig. 1) where $\overline{EF} = \frac{1}{2}\overline{AB}$ and $\overline{FG} = \frac{1}{2}\overline{BC}$ and starts the stochastic search process with a new population $\mathcal{P}^2(g)$ (where stage number $\tau = 2$ and $1 < g \leqslant G_{max}$). From Fig. 1 it is understood that EFGH is $\frac{1}{2^2}$ i.e., 1/4 times of ABCD in size (area).

Since the chromosome length still remains 6-bit, the search space resolution in EFGH is increased by a factor of four. Let the GA now find the current best solution $\mathbf{x}_B^2$ (depicted by a point Q in Fig. 1) in the new search space EFGH. Then at the third or final stage, the GA again moves to another search space WXYZ surrounding $\mathbf{x}_B^2$ following the same procedure. Again, the GA restarts its search over the reduced space WXYZ (with higher resolution) and finds the global or near-global solution $\mathbf{x}_B^3$ which is identified by **U** in Fig. 1. Thus, the search space resolution is altered T_{max} times (here, $T_{max} = 3$).

At each stage the space is partitioned into the same number of equal parts along both x and y-axes. However, the area of the newly defined space at each stage is reduced in size. This phenomenon of GA eventually increases the resolution of the new space at each subsequent stage since the same number of bits (of chromosome) represents a smaller physical space. Thus, the precision of the solution of the optimizing function is also increased.

If a function to be optimized is in high dimensional space, then partitioning by a factor of 2 along all dimensions can drastically reduce the space at each stage. For example, a 2-D space will be reduced by $1/2^2$, a 10-dimensional space will be re-

duced by $1/2^{10}$ and so on. It is likely to miss the global optimum by such drastic reduction (which has been noticed for the optimization problem in 100 dimensions depicted in Table 6). Instead, we can partition the search space at a slower rate by considering fewer dimensions at a time. The slower partitioning process in 2-D space is explained through Fig. 2, where it is done along only one dimension at a time.

The domain of initial space is ABCD and the chromosome length of solutions remains same as discussed above for Fig. 1. After the initial best solution is obtained at P, let the search space reduction be done along x-direction only. The new search space is now EFGH (shown with different shade in Fig. 2). At stage 2, the search starts with the new population in EFGH and finds the best solution Q (see Fig. 2). The partitioning is then done along y-direction only. The reduced search space is now IJKL (depicted by another shade in Fig. 2) enclosing Q. Let the best solution obtained in this space (IJKL) be R. Now the process at stage 2 is complete and the ultimate rectangle obtained is IJKL. Next, the stage 3 process is started in a similar manner and the final global or near-global solution found is **U** in the rectangular space WXYZ as shown in Fig. 2.

The results of finding the optimum for the mathematical functions in 100 dimensional space using fast and slow change of search space resolution are presented in Section 4.2.

2.3.2. Cascaded genetic approach

The genetic process starts with a population $\mathcal{P}^{\tau}(g)$ (where $1 \leqslant \tau \leqslant T_{max}$ and $1 \leqslant g \leqslant G_{max}$) of N (population size) randomly created individuals. The cascaded GA, however, differs from the conventional GA in the following manner:

1. The search is transferred from a larger hypercube to a smaller hypercube for a specified number of times T_{max}.
2. The search space resolution is redefined with the transfer of the search process to the newly defined hypercube.

Let us now define the parameters of the function $f(\mathbf{x})$ in R^n where $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and R^n represents the n-dimensional space. The searching process starts with low resolution and the algorithm finds a best solution $\mathbf{x}_B^{\tau} = \left(x_{B,1}^{\tau}, x_{B,2}^{\tau}, \ldots, x_{B,n}^{\tau}\right)$ which has not changed for several consecutive K_g generations. The search is then transferred to a smaller hypercube around $\mathbf{x}_B^{\tau}$ as depicted in Figs. 1 and 2 in 2-D space. At the τth stage the hypercube on n-dimensional space can be represented as

$$R^{\tau,n} = \{r^{\tau,1}, r^{\tau,2}, \ldots, r^{\tau,n}\} \tag{1}$$

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

Once the search process converges at $(\tau - 1)$th step, the GA is reinvoked in the new hypercube $R^{\tau,n}$ with same mutation rate and higher space resolution. The population $\mathcal{P}^{\tau}(g)$ at τth step is reproduced after transfer of the search process to $R^{\tau,n}$ hypercube. The individuals that are located both in $R^{\tau-1,n}$ and in $R^{\tau,n}$ except the elitist individual in $R^{\tau-1,n}$ are removed from the population $\mathcal{P}^{\tau-1}(g)$ where $(1 < \tau \leqslant T_{max})$ and $(1 < g \leqslant G_{max})$. A new population $\mathcal{P}^{\tau}(g)$ of N individuals is regenerated with $N - 1$ randomly created new individuals and the elite one of $\mathcal{P}^{\tau-1}(g)$. Thus, the diversity of the population is restored at each stage.

3. Hybridization of optimization approaches with TS process

The genetic methods discussed in Section 2 are hybridized with the local TS process described in Section 2.2. The hybridization of each genetic method eventually enhances the performance of the respective hybrid genetic approach.

3.1. Hybridized CAscaded Genetic Algorithm (H-CAGA)

We propose a hybrid genetic algorithm where the CAGA works along with the TS process. From the discussions in Section 2.3 it is known that the CAGA advances its search process hierarchically in multiple stages. At τth (initially $\tau = 1$) stage, it starts with a population $\mathcal{P}^{\tau}(g)$ (where $1 \leqslant \tau \leqslant T_{max}$ and $1 \leqslant g \leqslant G_{max}$) of N individuals and finds the best solution $\mathbf{x}_B^{\tau}$ in the initial hypercube in the multi-dimensional space.

In the hybrid process the TS technique is invoked at this point at the same hypercube without the change of resolution. The TS algorithm starts processing with the best solution $\mathbf{x}_B^{\tau}$ found so far by CAGA as its current solution $\mathbf{A}_c$ and generates randomly a pool of trial solutions consisting of λ individuals where $\lambda < N$. The trial solutions are denoted by $\mathbf{A}_t^j$, $\forall j \in \{1, 2, \ldots, \lambda\}$. Each $\mathbf{A}_t^j$ consists of the same number of bits of $\mathbf{x}_B^{\tau}$. The fitness values of λ trial solutions are calculated and the best solution $\mathbf{A}_b$ is picked, depending on the fitness values. If $\mathbf{A}_b$ satisfies the conditions of TS algorithm to be considered as the new current solution for the next generation, then $\mathbf{A}_c = \mathbf{A}_b$. The process is continued I_{max} times ($I_{max} \ll G_{max}$) to achieve the final best solution $\mathbf{A}_b^{\tau}$ (at τth stage) of the TS process. At the end, $\mathbf{A}_b^{\tau}$ may or may not be better (but not worse) than $\mathbf{x}_B^{\tau}$. The CAGA is then reinvoked where the search domain jumps from the hypercube $R^{\tau-1,n}$ to the new smaller hypercube $R^{\tau,n}$ with $\mathbf{A}_b^{\tau}$ as its best solution. The GA generates a new population of $\mathcal{P}^{\tau}(g)$ of N individuals including $\mathbf{A}_b^{\tau}$, following the approach described in Section 2.3. In doing so the search space resolution is redefined, as discussed in Sections 2.3. The cascaded GA restarts with a new population and it may again converge to a solution. As before, TS is again initiated at that point. Thus, to find the global optimal solution the hybrid genetic process is continued with the transfer of the search location from one hypercube to the neighboring hypercube for at most T_{max} times. The changeover from GA to TS is continued T_{max} times.

The algorithmic steps of the hybrid process are briefly discussed below where the steps of the CAGA and the TS process remain as before. Both algorithms are executed equal number of times for a problem.

Step 1: Set $\tau = 1$. Initialize the CAGA as well as the TS parameters.
Step 2: Start CAGA with a pool of N individuals and continue the process.
Step 3: If the best solution of CAGA has not changed during K_g iterations, invoke TS.
Step 4: Consider the best solution of the CAGA as the current solution of TS at the respective stage.
Step 5: Continue the TS process for I_{max} times.
Step 6: If $\tau > T_{max}$, terminate the hybrid algorithm. Otherwise, $\tau = \tau + 1$ and consider the best solution of TS as the best solution of CAGA for the next generation.
Step 7: Redefine the new hypercubic search space with higher resolution. Generate a pool of N individuals including the best solution found in Step 6 and go to Step 2.

3.2. Hybridization of other methods

The CGA is also hybridized with TS similar to the hybridization of CAGA. The performance of the hybrid CGA is compared with the hybrid CAGA along with other relevant optimization algorithms for function optimization as well as pattern recognition involving point pattern matching. Similarly, the technique APPMGA [26] used in PPR, is also hybridized with TS process. The pattern matching in APPMGA is based on genetic approach and it is relevant to compare with the proposed method after its hybridization. The performance comparison of the hybrid methods on PPR is discussed in Section 4.4.

4. Experimental studies

We have evaluated the performance of the proposed hybrid method in two parts. In the first part, efficiency of the H-CAGA is compared with several relevant genetic approaches on some mathematical functions for establishing it as a good optimization algorithm. The pattern recognition problem is considered in part two to examine how well the proposed optimization technique performs for dot pattern shape matching and object matching with edge map.

4.1. Parameter initialization

Values of the parameters used in the test cases are based on literatures [12,17,18,23,24]. In the present experiment auto-adaptive mutation is used for the sequential genetic methods with the population size μ (=50). The crossover probability p_c in the range [0.5–0.9] and the initial mutation probability p_m in the range [0.002–0.009] have been considered. For function optimization the initial population of μ individuals is generated randomly within the specified range in Tables 1 and 2. T_{max} is always 4 in our experiment. It may be increased for higher search space resolution. The number of test runs is 50 and the considered value of K_g is 5. A test *run* represents the completion of an algorithm for a set of inputs after G_{max} iterations.

Since TS process accepts the best solution obtained by the genetic algorithm, its string length of the individual is the same as that of the genetic method. In each run, the TS process is continued for $I_{max} = 10$ times. The maximum tabu list size MTLS is 10 and the population size of the trial solution λ is 20.

For the point pattern recognition problem G_{max} in each run is 100 and the population of μ (=50) individuals is randomly generated. The values of other parameters of the GA and TS process are the same as in the case of function optimization.

4.2. Results on mathematical function optimization

A set of 20 test functions are chosen from [23] and the proposed hybrid algorithm is tested on them. Some basic characteristics of these functions are listed in Tables 1 and 2. The functions in Table 1 belong to high (30 or 100) dimensional space while the functions in Table 2 are in low dimensional space and these are a combination of unimodal and multimodal functions.

The listed benchmark functions (in Tables 1 and 2) have been evaluated by various evolutionary techniques [23]. Among them, ALEP (Adaptive Levy Evolutionary Programming) [17] uses evolutionary programming with adaptive Levy mutation in order to generate offspring for each new generation. It is also noted in [17] that the non-adaptive algorithm can never outperform ALEP. The orthogonal genetic algorithm with quantization (OGA/Q) [18] uses an orthogonal design to construct a crossover operator. Another evolutionary approach based method FEP (Fast Evolutionary Programming) [24] uses evolutionary programming with Cauchy mutation to generate offspring for each new generation. Hong and Quan [12] proposed a theoretical approach called the mean-value-level-set method (M-L method) by improving the mean of the objective function value on the level set. Wang and Dang [23] designed the level-set evolutionary algorithm (LEA) for global optimization with Latin squares. Its application leads to an effective crossover operator. Tsai et al. [22] proposed the hybrid algorithm (HTGA) combining the Taguchi method and a genetic algorithm. The Taguchi technique is a robust experimental design approach which uses both the orthogonal design method and the *signal-to-noise* ratio. Another hybrid technique named as CGA-TS, is a combination of the Conventional GA and the TS method.

Table 1
Basic characteristics of high dimensional functions tested for optimization.

Function	Dimension, n	Range	Optimum value
$f_1(\mathbf{x}) = \sum_{i=1}^{n} -x_i sin(\sqrt{\lvert x_i \rvert})$	30	$-500 \leqslant x_i \leqslant 500$	−12569.5
$f_2(\mathbf{x}) = \sum_{i=1}^{n} (x_i^2 - 10cos(2\pi x_i) + 10$	30	$-5.12 \leqslant x_i \leqslant 5.12$	0.00
$f_3(\mathbf{x}) = -20exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - exp\left(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)\right) + 20 + e$	30	$-32 \leqslant x_i \leqslant 32$	0.00
$f_4(\mathbf{x}) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	$-600 \leqslant x_i \leqslant 600$	0.00
$f_5(\mathbf{x}) = \frac{\pi}{n}\left\{10\ sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2 \times [1 + 10\ sin^2(\pi y_{i+1})] + (y_n - 1)^2\right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$ where $y_i = 1 + \frac{1}{4}(x_i + 1)$ and $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a \leqslant x_i \leqslant a \\ k(-x_i - a)^m & x_i < -a \end{cases}$	30	$-50 \leqslant x_i \leqslant 50$	0.00
$f_6(\mathbf{x}) = \frac{1}{10}\left\{sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i - 1)^2 \times [1 + sin^2(3\pi x_{i+1})] + (x_n - 1)^2[1 + sin^2(2\pi x_n)]\right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	30	$-50 \leqslant x_i \leqslant 50$	0.00
$f_7(\mathbf{x}) = -\sum_{i=1}^{n} sin(x_i) sin^{20}\left(\frac{i \times x_i^2}{\pi}\right)$	100	$0 \leqslant x_i \leqslant \pi$	−99.2784
$f_8(\mathbf{x}) = \sum_{i=1}^{n}\left[\sum_{j=1}^{n}(\chi_{ij} sin\omega_j + \psi_{ij} cos\omega_j) - \sum_{j=1}^{n}(\chi_{ij} sinx_j + \psi_{ij} cosx_j)\right]$	100	$-\pi \leqslant x_j \leqslant \pi$ $-100 \leqslant \chi_{ij}, \psi_{ij} \leqslant 100$ $\& -\pi \leqslant \omega_j \leqslant \pi$	0.00
$f_9(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}(x_i^4 - 16x_i^2 + 5x_i)$	100	$-5 \leqslant x_i \leqslant 5$	−78.33236
$f_{10}(\mathbf{x}) = \sum_{i=1}^{n-1}\left[100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2\right]$	100	$-5 \leqslant x_i \leqslant 10$	0.00
$f_{11}(\mathbf{x}) = \sum_{i=1}^{n} x_i^2$	30	$-100 \leqslant x_i \leqslant 100$	0.00
$f_{12}(\mathbf{x}) = \sum_{i=1}^{n} x_i^4 + random[0, 1)$	30	$-1.28 \leqslant x_i \leqslant 1.28$	0.00
$f_{13}(\mathbf{x}) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	$-10 \leqslant x_i \leqslant 10$	0.00
$f_{14}(\mathbf{x}) = \sum_{i=1}^{n}\left(\sum_{j=1}^{i} x_j\right)^2$	30	$-100 \leqslant x_j \leqslant 100$	0.00
$f_{15}(\mathbf{x}) = max\{\lvert x_i \rvert, i = 1, 2, \ldots, n\}$	30	$-100 \leqslant x_i \leqslant 100$	0.00

Tables 1 and 2 contain the set of test functions with their optimization ranges and optimum values. Tables 3 and 4 show the results of evolutionary algorithms tested on the mathematical functions in Tables 1 and 2. These evolutionary hybridized and non-hybridized algorithms are compared with our proposed method (H-CAGA). The computation results provided in Tables 3 and 4 for the optimization methods (except H-CAGA and CGA-TS) are taken from Ref. [23]. From Tables 3 and 4 it is observed that the proposed genetic approach (H-CAGA) performs better or similar to the other listed optimized algorithms for the mathematical test functions. It is noted that none of the techniques can reach the exact global optimum (given in Table 1) for the functions f_1, f_5–f_{10} and f_{12} (see Tables 3 and 4). However, for f_5–f_8, f_{10} and f_{12} none of the listed techniques including H-CAGA can attain the global or near-global optimum, although the optimum value achieved by H-CAGA is better than the other optimized methods. On the other hand, for functions f_3, f_4, f_{12} and f_{13} H-CAGA converges to the same or better (for f_{12}) optimum value with more number of iterations compared with the hybridized method HTGA. In case of the remaining functions H-CAGA reaches the exact global optimum or the optimum value reached by the listed methods with lesser number of iterations. For low dimensional functions (f_{15} and f_{18}) H-CAGA can achieve the exact global optimum with very little number of iterations compared with the other relevant algorithms in Table 4. For f_1 and f_9, none of the optimized techniques including H-CAGA can reach the exact global optimum. However, the optimum values achieved by other optimized techniques are better than H-CAGA for f_9 and for f_1 H-CAGA can achieve the 4th highest optimum value. Finally, it was observed that the overall performance of H-CAGA is better than other evolutionary methods (listed in Tables 3 and 4) in terms of achieving optimum value for most of the functions except f_1 and f_9. It is also noted that the performance of the CGA-TS is better than M-L optimization technique except for f_5, f_8 and f_{15} but worse than the top four optimization techniques (namely, H-CAGA, HTGA, LEA and OGA/Q) listed in Tables 3 and 4.

To see how well H-CAGA performs for problems with 100 dimensions we have tested the functions (f_2–f_{15}) of Table 1 for 100-dimension. We did not consider other functions since the optimum value would be different and we did not have any gold standard for them. It was noted that the functions were unable to come within the vicinity of the global optimum if the search space is rapidly reduced by partitioning along all dimensions at each stage (see Table 6). In such a situation, we reduce the search space gradually or slowly, as discussed in Section 2.3, to further improve the performance of the proposed method. It was then observed that most of the functions could reach the global optimum as depicted in Table 5. It was also noted that if the space was partitioned simultaneously along more than 7 dimensions at each stage then the test functions (f_2–f_{15})

Table 2
Basic characteristics of low dimensional functions tested for optimization.

Function	Dimension, n	Range	Optimum value
$f_{16}(\mathbf{x}) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	$-5 \leqslant x_i \leqslant 5$	−1.0316285
$f_{17}(\mathbf{x}) = \left(x_2 - \frac{5.1}{4\pi_2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10(1 - \frac{1}{8\pi})cosx_1 + 10$	2	$-5 \leqslant x_1 \leqslant 10$ and $0 \leqslant x_2 \leqslant 15$	0.398
$f_{18}(\mathbf{x}) = \left[1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\right] \times \left[30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\right]$	2	$-2 \leqslant x_i \leqslant 2$	3.00
$f_{19}(\mathbf{x}) = \sum_{i=1}^{11}\left[a_i - \frac{x_1(b_i^2 + b_ix_2)}{b_i^2 + b_ix_3 + x_4}\right]^2$ where $a_1, \ldots, a_{11}$ = [0.1957, 0.1947, 0.1735, 0.1600, 0.0844, 0.0627, 0.0456, 0.0342, 0.0323, 0.0235, 0.0246] and $[b_1, \ldots, b_{11}] = \left[4, 2, 1, \frac{1}{2}, \frac{1}{4}, \frac{1}{6}, \frac{1}{8}, \frac{1}{10}, \frac{1}{12}, \frac{1}{14}, \frac{1}{16}\right]$	4	$-5 \leqslant x_i \leqslant 5$	0.0003075
$f_{20}(\mathbf{x}) = -\sum_{i=1}^{4}c_i exp\left[-\sum_{j=1}^{6}a_{ij}(x_j - p_{ij})^2\right]$ where $[c_1,c_2,c_3,c_4]$ = [1,1.2,3,3.2] $[a_{ij}]_{4\times6} = \begin{bmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{bmatrix}$ $[p_{ij}]_{4\times6} = \begin{bmatrix} 0.1312 & 0.1696 & 0.5569 & 0.0124 & 0.8283 & 0.5886 \\ 0.2329 & 0.4135 & 0.8307 & 0.3736 & 0.1004 & 0.9991 \\ 0.2348 & 0.1415 & 0.3522 & 0.2883 & 0.3047 & 0.6650 \\ 0.4047 & 0.8828 & 0.8732 & 0.5743 & 0.1091 & 0.0381 \end{bmatrix}$	6	$0 \leqslant x_j \leqslant 1$	−3.32

could not reach the target solution. The situation for simultaneous reduction of search space along all dimensions (i.e., 100) is shown in Table 6 where the results are far away from the optimum. We noted that other high dimensional functions follow similar nature. It is to be noted too that the results of the functions (f_7–f_{10}) of 100 dimensions, shown in Tables 3 and 4, were also achieved by slow reduction of search space (as discussed in Section 2.3). For the proposed method the slow reduction of search space is necessary for 100 dimensional problems for the best performance.

4.3. Statistical comparison of H-CAGA with relevant optimizing algorithms

We have employed some non-parametric statistical procedures to obtain the quantitative performance differences between H-CAGA and the relevant optimizing algorithms. The performance of the algorithms has been measured across the same set of problems (mathematical functions). The procedures used for performance measurement are Friedman test, Multiple Sign-test and the median based Contrast Estimation. The ranking done by these statistical tests eventually helps to identify the best performing algorithm.

4.3.1. Friedman test and Iman–Davenport extension

The Friedman Test [6] is a non-parametric analog of the parametric two-way analysis of variance [8]. For this test the rank of the algorithms for each problem is separately computed from the results in Tables 3 and 4. The ranks are given as follows. The best performing algorithm will get the rank of 1, the second best will have rank 2 and so on, as shown in Table 7. In case of ties, the average rank will be calculated. We have considered the *mean best* value of the algorithms for ranking. However, if the *mean best* values of two algorithms are the same, we have broken the tie using the *Mean Func. Eval.* values of the corresponding algorithms.

In the experiment we have considered k (=4) number of algorithms (best four algorithms namely, OGA/Q, HTGA, LEA and H-CAGA obtained by consulting optimum values in Tables 3 and 4) and n (=15) number of functions. The remaining five functions (f_{16}–f_{20}) have not been considered because we have not received data (as given in Tables 3 and 4) of the algorithms OGA/Q and HTGA for those functions. Let r_i^j be the rank of the jth algorithm on the ith function. Now, the average rank of the jth algorithm is $R_j = \frac{1}{n}\sum_i r_i^j$. Under the null hypothesis, which states that all the algorithms behave similarly and thus their ranks R_j should be equal, the Friedman statistics

$$\chi_F^2 = \frac{12n}{k(k+1)}\left[\sum_{j=1}^{k}R_j^2 - \frac{k(k+1)^2}{4}\right] \tag{2}$$

is distributed according to χ_F^2 with $(k-1)$ degrees of freedom, when n and k are reasonably big.

In [13], Iman and Davenport noted that Friedman χ_F^2 presents a pessimistic behavior. They proposed a statistic

Table 3
Comparative results of H-CAGA with relevant genetic methods.

Function	Algorithms	Mean func-eval	Mean best	Std. dev.
	ALEP	150,000	−11469.2	58.2
	FEP	900,000	−12554.5	52.6
	OGA/Q	302,116	−12569.4537	6.4×10^{-4}
$f_1(\mathbf{x})$	M-L	655,895	−5461.826	275.15
	CGA-TS	245,500	−12561.7	7.8
	HTGA	163,468	−12569.46	0.00
	LEA	287,365	−12569.4542	4.8×10^{-4}
	H-CAGA	53,050	−12569.393	9.0×10^{-1}
	ALEP	150,000	5.85	2.07
	FEP	500,000	0.046	0.012
	OGA/Q	224,710	0.00	0.00
$f_2(\mathbf{x})$	M-L	305,899	121.75	7.75
	CGA-TS	98,900	13.78	13.53
	HTGA	16,267	0.00	0.00
	LEA	223,803	2.1×10^{-18}	3.3×10^{-18}
	H-CAGA	15,658	0.00	0.00
	ALEP	150,000	0.019	0.0010
	FEP	150,000	0.018	0.021
	OGA/Q	112,421	4.4×10^{-16}	3.9×10^{-17}
$f_3(\mathbf{x})$	M-L	121,435	2.59	0.094
	CGA-TS	242,700	0.00	0.00
	HTGA	16,632	0.00	0.00
	LEA	105,926	3.2×10^{-16}	3.0×10^{-17}
	H-CAGA	18,900	0.00	0.00
	ALEP	150,000	0.024	0.028
	FEP	200,000	0.016	0.022
	OGA/Q	134,000	0.00	0.00
$f_4(\mathbf{x})$	M-L	151,281	0.1189	0.0104
	CGA-TS	245,600	4.7×10^{-4}	7.0×10^{-4}
	HTGA	20,999	0.00	0.00
	LEA	130,498	6.1×10^{-16}	2.5×10^{-17}
	H-CAGA	27,825	0.00	0.00
	ALEP	150,000	6.0×10^{-6}	1.0×10^{-6}
	FEP	150,000	9.2×10^{-6}	3.6×10^{-6}
	OGA/Q	134,556	6.0×10^{-6}	1.1×10^{-6}
$f_5(\mathbf{x})$	M-L	146,209	0.21	0.036
	CGA-TS	440,500	1.85	1.857
	HTGA	66,457	1.0×10^{-6}	0.00
	LEA	132,642	2.4×10^{-6}	2.3×10^{-6}
	H-CAGA	65,450	1.0×10^{-6}	9.9×10^{-12}
	ALEP	150,000	9.8×10^{-5}	1.2×10^{-5}
	FEP	150,000	1.6×10^{-4}	7.3×10^{-5}
	OGA/Q	134,143	1.9×10^{-4}	2.6×10^{-5}
$f_6(\mathbf{x})$	M-L	147,928	1.50	2.25
	CGA-TS	243,250	6.2×10^{-6}	7.5×10^{-6}
	HTGA	59,033	1.0×10^{-4}	0.00
	LEA	130,213	1.7×10^{-4}	1.2×10^{-4}
	H-CAGA	29,050	1.4×10^{-6}	2.1×10^{-11}
	OGA/Q	302,773	−92.83	0.03
	M-L	329,087	−23.97	0.62
$f_7(\mathbf{x})$	CGA-TS	496,650	−87.56	11.75
	HTGA	265,693	−92.83	0.00
	LEA	289,863	−93.01	0.02
	H-CAGA	244,600	−93.86	0.02
	OGA/Q	190,031	4.7×10^{-7}	1.3×10^{-7}
	M-L	221,547	2.5×10^{4}	1.7×10^{3}
$f_8(\mathbf{x})$	CGA-TS	237,550	5.3×10^{7}	5.3×10^{7}
	HTGA	186,816	5.9×10^{-5}	8.3×10^{-5}
	LEA	189,427	1.6×10^{-6}	6.5×10^{-7}
	H-CAGA	150,350	5.2×10^{-7}	2.7×10^{-12}

Table 3 (*continued*)

Function	Algorithms	Mean func-eval	Mean best	Std. dev.
$f_9(\mathbf{x})$	OGA/Q	245,903	−78.30	6.3×10^{-3}
	M-L	251,199	−35.80	0.89
	CGA-TS	495,850	−74.95	3.43
	HTGA	216,535	−78.30	0.00
	LEA	243,895	−78.31	6.1×10^{-3}
	H-CAGA	47,650	−78.19	0.53

[For H-CAGA, space partitioning process was done along all dimensions simultaneously for functions with 30 dimensions. "Mean Func-Eval" denotes the average number of function evaluations to reach the optimum value. "Mean Best" indicates *Mean Best Function Values* found in the last generation and "Std. Dev." stands for *Standard Deviation*.]

$$F_F = \frac{(n-1)\chi_F^2}{n(k-1)-\chi_F^2} \tag{3}$$

which is distributed according to the F-distribution with $(k-1)$ and $(k-11)(n-11)$ degrees of freedom.

From Table 7 (which compares the four algorithms) it is noticed that H-CAGA ranked first with an average score of 1.666, HTGA ranked second with the score 2.0 and so on. Now, it is required to show by the Friedman test whether the measured average ranks are significantly different from the mean rank R_j (= 2.5) expected under the null hypothesis. Here, we have

$$\chi_F^2 = \frac{12\times 15}{4\times 5}\left[(1.666^2+2.0^2+2.466^2+3.666^2)-\frac{4\times 5^2}{4}\right] = 11.666 \tag{4}$$

and

$$F_F = \frac{14\times 11.666}{14\times 3-11.666} = 5.384 \tag{5}$$

Now, for 4 algorithms and 15 functions, F_F is distributed according to the F distribution with $(4-1)=3$ and $(4-1)\times(15-1)=42$ degrees of freedom. The p-value computed by using the $F(3, 42)$ distribution is 3.2×10^{-4} [source – http://graphpad.com/quickcalcs/PValue1.cfm]. Therefore, the null hypothesis is rejected at a high level of significance.

4.3.2. Multiple sign-test

The procedure proposed by [21] is called multiple sign-test, which is basically an extension of the Sign test [3]. It compares the algorithm with control label 1 (see Table 8) with all other algorithms considered in the statistical experiment. The procedure is described by the following steps.

Step 1: Find the signed differences $d_{ij} = x_{i1} - x_{ij}$ where x_{i1} and x_{ij} be the performance of the control and that of the jth optimizer in the ith problem, respectively.

Step 2: If M_1 is the median value of a sample of results of the control method and M_j is the median value of a sample of results of the jth algorithm, then apply either of the following decision rules.

(a) For testing the hypothesis H_0: $M_j \geqslant M_1$ against H_1: $M_j < M_1$, reject H_0 if the number of positive signs is less than or equal to the critical value appearing in the Table (critical value computation table) of [21] (same as Table A.1 of [8]) for $k-11$ (number of algorithms excluding control), n and the chosen experiment-wise error rate.

(b) For testing the hypothesis H_0: $M_j \leqslant M_1$ against H_1: $M_j > M_1$, reject H_0 if the number of negative signs is less than or equal to the critical value appearing in the Table of [21] for $k-11$, n and the chosen experiment-wise error rate.

Table 8 depicts the computation of the above steps. Following [8] we have also considered a level of significance $\alpha = 0.05$. Now, our hypothesis be H_0: $M_j \geqslant M_0$ and H_1: $M_j < M_0$. Thus, according to the hypothesis the control algorithm H-CAGA is better than the remaining three optimizing algorithms. From the Table of [21] for $(k-11)=3$ and $n=15$, we get the critical value 3. Since the number of positive signs in the pairwise comparison between the control and LEA and OGA/Q algorithms are each less than or equal to 3, the performance of H-CAGA is better than them. However, the null hypothesis cannot be rejected in the pairwise comparison between H-CAGA and HTGA. Thus, we conclude that they perform almost equally well.

4.3.3. Median based contrast evaluation

Table 9 contains the performance differences of a pair of algorithms from a set of four optimizing algorithms over multiple functions. A procedure proposed in [5] makes an estimation of the expected differences. In this approach it is assumed that the expected differences between the performances of algorithms are identical across functions. Also, it is assumed that the performance is reflected by the magnitudes of the differences between the observed performances of the algorithms [8]. The approach is based on computing the contrast based on medians of each set of differences.

Table 4
Comparative results of H-CAGA with relevant genetic methods.

Function	Algorithms	Mean func-eval	Mean best	Std. dev.
	OGA/Q	167,863	0.75	0.11
	M-L	137,100	2935.93	134.81
$f_{10}(\mathbf{x})$	CGA-TS	997,900	1072.80	1155.88
	HTGA	60,737	0.70	0.00
	LEA	168,910	0.56	0.11
	H-CAGA	234,250	1.4×10^{-2}	3.2×10^{-3}
	ALEP	150,000	6.3×10^{-4}	7.6×10^{-5}
	FEP	150,000	5.7×10^{-4}	1.3×10^{-4}
	OGA/Q	112,559	0.00	0.00
$f_{11}(\mathbf{x})$	M-L	162,010	3.19	0.29
	CGA-TS	146,900	3.6×10^{-4}	5.9×10^{-4}
	HTGA	20,844	0.00	0.00
	LEA	110,674	4.7×10^{-16}	6.2×10^{-17}
	H-CAGA	15,850	0.00	0.00
	OGA/Q	112,652	6.3×10^{-3}	4.1×10^{-4}
	M-L	124,982	1.70	0.52
$f_{12}(\mathbf{x})$	CGA-TS	207,450	2.4×10^{-3}	2.5×10^{-3}
	HTGA	20,065	1.0×10^{-3}	0.00
	LEA	111,093	5.1×10^{-3}	4.4×10^{-4}
	H-CAGA	32,650	2.8×10^{-6}	1.5×10^{-10}
	FEP	200,000	8.1×10^{-3}	7.7×10^{-4}
	OGA/Q	112,612	0.00	0.00
$f_{13}(\mathbf{x})$	M-L	120,176	9.74	0.46
	CGA-TS	146,900	3.6×10^{-4}	5.9×10^{-4}
	HTGA	14,285	0.00	0.00
	LEA	110,031	4.2×10^{-19}	4.2×10^{-19}
	H-CAGA	17,050	0.00	0.00
	ALEP	150,000	0.04	0.06
	FEP	500,000	0.02	0.01
	OGA/Q	112,576	0.00	0.00
$f_{14}(\mathbf{x})$	M-L	155,783	2.21	0.504
	CGA-TS	145,500	7.8×10^{-4}	1.4×10^{-3}
	HTGA	26,469	0.00	0.00
	LEA	110,604	6.8×10^{-18}	5.4×10^{-18}
	H-CAGA	12,650	0.00	0.00
	FEP	500,000	0.3	0.5
	OGA/Q	112,893	0.00	0.00
$f_{15}(\mathbf{x})$	M-L	125,439	0.55	0.039
	CGA-TS	242,600	0.79	0.801
	HTGA	21,261	0.00	0.00
	LEA	111,105	2.6×10^{-16}	6.3×10^{-17}
	H-CAGA	725	0.00	0.00
	ALEP	3,000	−1.031	0.00
	FEP	10,000	−1.03	4.9×10^{-7}
$f_{16}(\mathbf{x})$	M-L	13,592	−1.02662	5.3×10^{-3}
	CGA-TS	4,800	−1.03	1.6×10^{-3}
	LEA	10,823	−1.03108	3.4×10^{-7}
	H-CAGA	1,685	−1.03	2.7×10^{-5}
	FEP	10,000	0.398	1.5×10^{-7}
	M-L	12,703	0.403297	8.8×10^{-3}
$f_{17}(\mathbf{x})$	CGA-TS	2,570	0.398	0.00
	LEA	10,538	0.398	2.7×10^{-8}
	H-CAGA	5,985	0.398	0.00
	ALEP	3,000	3.00	0.00
	FEP	10,000	3.02	0.11
$f_{18}(\mathbf{x})$	M-L	16,325	3.048	0.0603
	CGA-TS	1,415	3.00	0.00
	LEA	11,721	3.00003	6.2×10^{-5}
	H-CAGA	295	3.00	0.00
	FEP	400,000	5.0×10^{-4}	3.2×10^{-4}
	M-L	186,768	1.3×10^{-3}	2.9×10^{-4}
$f_{19}(\mathbf{x})$	CGA-TS	192,350	3.1×10^{-4}	1.2×10^{-5}
	LEA	55,714	3.5×10^{-4}	7.4×10^{-5}
	H-CAGA	51,500	3.1×10^{-4}	2.5×10^{-12}

Table 4 (*continued*)

Function	Algorithms	Mean func-eval	Mean best	Std. dev.
	FEP	20,000	−3.27	0.059
	M-L	92,516	−3.12696	0.067397
$f_{20}(\mathbf{x})$	CGA-TS	38,050	−3.32	0.00
	LEA	28,428	−3.301	7.8×10^{-3}
	H-CAGA	3,350	−3.32	0.00

For H-CAGA, space partitioning process was done along all dimensions simultaneously. The terms of the table are described in Table 3.

Table 5
Results of H-CAGA for the functions with higher dimension and slower reduction of the search space as discussed in Section 2.3.

Function	Dimension	Mean func-eval	Mean best	Std. dev.
$f_2(\mathbf{x})$	100	162,100	0.00	0.00
$f_3(\mathbf{x})$	100	192,552	0.00	0.00
$f_4(\mathbf{x})$	100	207,290	0.00	0.00
$f_5(\mathbf{x})$	100	198,500	1.6×10^{-3}	2.8×10^{-5}
$f_6(\mathbf{x})$	100	59,900	1.0×10^{-6}	9.9×10^{-12}
$f_7(\mathbf{x})$	100	244,600	93.86	0.02
$f_8(\mathbf{x})$	100	150,350	5.2×10^{-7}	2.7×10^{-12}
$f_9(\mathbf{x})$	100	47,650	78.19	0.53
$f_{10}(\mathbf{x})$	100	234,250	1.4×10^{-2}	3.2×10^{-3}
$f_{11}(\mathbf{x})$	100	180,050	0.00	0.00
$f_{12}(\mathbf{x})$	100	130,250	1.8×10^{-4}	4.5×10^{-7}
$f_{13}(\mathbf{x})$	100	132,770	0.00	0.00
$f_{14}(\mathbf{x})$	100	161,175	0.00	0.00
$f_{15}(\mathbf{x})$	100	4,785	0.00	0.00

Partitioning is done in four directions at a time. The terms of the table are described in Table 3.

Table 6
Results of H-CAGA for the functions with higher dimension and fast reduction of the search space as discussed in Section 2.3.

Function	Dimension	Mean func-eval	Mean best	Std. dev.
$f_3(\mathbf{x})$	100	1,444,450	3.0×10^{-2}	1.0×10^{-3}
$f_4(\mathbf{x})$	100	1,434,270	8.5×10^{-2}	9.4×10^{-3}
$f_{11}(\mathbf{x})$	100	1,215,440	1.1×10^{-1}	1.6×10^{-2}
$f_{13}(\mathbf{x})$	100	1,367,440	1.8×10^{-1}	3.2×10^{-2}
$f_{14}(\mathbf{x})$	100	1,446,160	$6.9 \times 10^{+1}$	$5.5 \times 10^{+3}$

Partitioning is done along all 100 dimensions simultaneously. The terms of the table are described in Table 3.

The performance measure can be obtained as follows [8]:

Step 1: For each pair of k algorithms considered, the difference between the performances of two algorithms for each of the n functions, the performance differences $D_{i(uv)} = x_{iu} - x_{iv}$ where $i = 1, \ldots, n$; $u = 1, \ldots, k$; $v = 1, \ldots, k$ and $u < v$.

Step 2: Let Z_{uv} be the median of each set of differences and also be the *unadjusted estimator* of $M_u - M_v$. Since $Z_{vu} = -Z_{uv}$ and $Z_{uu} = 0$, we have to calculate only Z_{uv} for $u < v$. Total number of medians to be computed is $k(k-11)/2$.

Step 3: The mean of each set of unadjusted medians (m_u) is computed as follows.

$$m_u = \frac{\sum_{j=1}^{k} Z_{uj}}{k}, \quad u = 1, \ldots, k \tag{6}$$

Step 4: Now, $M_u - M_v = m_u - m_v$ where $u, v = 1, \ldots, k$.

From Table 9, we find that the six medians are $Z_{12} = 0.0$, $Z_{13} = 0.0$, $Z_{14} = -4.7 \times 10^{-16}$, $Z_{23} = 0.0$, $Z_{24} = 1.2 \times 10^{-16}$ and $Z_{34} = -2.6 \times 10^{-16}$. We now calculate the following averages for M_1 and M_2, i.e.,

$$m_1 = \frac{0 + 0.0 + 0.0 + (-4.7 \times 10^{-16})}{4} = -1.2 \times 10^{-16} \tag{7}$$

$$m_2 = \frac{0.0 + 0 + 0.0 + 1.2 \times 10^{-16}}{4} = 3.0 \times 10^{-17} \tag{8}$$

Table 7
Comparison of optimum values among best four algorithms selected from Tables 3 and 4. The numbers in the parenthesis represent ranks. They are used in the Friedman test.

Function	H-CAGA	OGA/Q	HTGA	LEA
$f_1(\mathbf{x})$	−12569.393(4)	−12569.453(3)	−12569.460(1)	−12569.454(2)
$f_2(\mathbf{x})$	0.00(1)	0.00(4)	0.00(2)	2.1×10^{-18}(3)
$f_3(\mathbf{x})$	0.00(2)	4.4×10^{-16}(4)	0.00(1)	3.2×10^{-16}(3)
$f_4(\mathbf{x})$	0.00(2)	0.00(4)	0.00(1)	6.1×10^{-16}(3)
$f_5(\mathbf{x})$	1.0×10^{-6}(1)	6.0×10^{-6}(4)	1.0×10^{-6}(2)	2.4×10^{-6}(3)
$f_6(\mathbf{x})$	1.4×10^{-6}(1)	1.9×10^{-4}(4)	1.0×10^{-4}(2)	1.7×10^{-4}(3)
$f_7(\mathbf{x})$	−93.86(1)	−92.83(4)	−92.83(3)	−93.01(2)
$f_8(\mathbf{x})$	5.2×10^{-7}(2)	4.7×10^{-7}(1)	5.9×10^{-5}(4)	1.6×10^{-6}(3)
$f_9(\mathbf{x})$	−78.19(4)	−78.30(3)	−78.30(2)	−78.31(1)
$f_{10}(\mathbf{x})$	1.4×10^{-2}(1)	0.75(4)	0.70(3)	0.56(2)
$f_{11}(\mathbf{x})$	0.00(1)	0.00(4)	0.00(2)	4.7×10^{-16}(3)
$f_{12}(\mathbf{x})$	2.8×10^{-6}(1)	6.3×10^{-3}(4)	1.0×10^{-3}(2)	5.1×10^{-3}(3)
$f_{13}(\mathbf{x})$	0.00(2)	0.00(4)	0.00(1)	4.2×10^{-19}(3)
$f_{14}(\mathbf{x})$	0.00(1)	0.00(4)	0.00(2)	6.8×10^{-18}(3)
$f_{15}(\mathbf{x})$	0.00(1)	0.00(4)	0.00(2)	2.6×10^{-16}(3)
Average rank	1.666	3.666	2.0	2.466

Table 8
Comparison of optimum values between the control algorithm H-CAGA and the rest of the algorithms selected for statistical analysis. The signs in the parenthesis are used in the computation of the multiple Sign test.

Function	H-CAGA 1 (Control)	OGA/Q 2	HTGA 3	LEA 4
$f_1(\mathbf{x})$	−12569.393	−12569.453(+)	−12569.460(+)	−12569.454(+)
$f_2(\mathbf{x})$	0.00	0.00(−)	0.00(−)	2.1×10^{-18}(−)
$f_3(\mathbf{x})$	0.00	4.4×10^{-16}(−)	0.00(+)	3.2×10^{-16}(−)
$f_4(\mathbf{x})$	0.00	0.00(−)	0.00(+)	6.1×10^{-16}(−)
$f_5(\mathbf{x})$	1.0×10^{-6}	6.0×10^{-6}(−)	1.0×10^{-6}(−)	2.4×10^{-6}(−)
$f_6(\mathbf{x})$	1.4×10^{-6}	1.9×10^{-4}(−)	1.0×10^{-4}(−)	1.7×10^{-4}(−)
$f_7(\mathbf{x})$	−93.86	−92.83(−)	−92.83(−)	−93.01(−)
$f_8(\mathbf{x})$	5.2×10^{-7}	4.7×10^{-7}(+)	5.9×10^{-5}(−)	1.6×10^{-6}(−)
$f_9(\mathbf{x})$	−78.19	−78.30(+)	−78.30(+)	−78.31(+)
$f_{10}(\mathbf{x})$	1.4×10^{-2}	0.75(−)	0.70(−)	0.56(−)
$f_{11}(\mathbf{x})$	0.00	0.00(−)	0.00(−)	4.7×10^{-16}(−)
$f_{12}(\mathbf{x})$	2.8×10^{-6}	6.3×10^{-3}(−)	1.0×10^{-3}(−)	5.1×10^{-3}(−)
$f_{13}(\mathbf{x})$	0.00	0.00(−)	0.00(+)	4.2×10^{-19}(−)
$f_{14}(\mathbf{x})$	0.00	0.00(−)	0.00(−)	6.8×10^{-18}(−)
$f_{15}(\mathbf{x})$	0.00	0.00(−)	0.00(−)	2.6×10^{-16}(−)
Number of (−)ve signs		12	10	13
Number of (+)ve signs		3	5	2

Table 9
Differences of performance in each function for different pairs of algorithms listed in Table 8.

Function	$D_{i(12)}$	$D_{i(13)}$	$D_{i(14)}$	$D_{i(23)}$	$D_{i(24)}$	$D_{i(34)}$
$f_1(\mathbf{x})$	0.06	0.067	0.061	0.007	0.001	−0.006
$f_2(\mathbf{x})$	0.00	0.00	-2.1×10^{-18}	0.00	-2.1×10^{-18}	-2.1×10^{-18}
$f_3(\mathbf{x})$	-4.4×10^{-16}	0.00	-3.2×10^{-16}	4.4×10^{-16}	1.2×10^{-16}	-3.2×10^{-16}
$f_4(\mathbf{x})$	0.00	0.00	-6.1×10^{-16}	0.00	-6.1×10^{-16}	-6.1×10^{-16}
$f_5(\mathbf{x})$	-5.0×10^{-6}	0.00	-1.4×10^{-6}	5.0×10^{-6}	3.6×10^{-6}	-1.4×10^{-6}
$f_6(\mathbf{x})$	-1.9×10^{-4}	-9.8×10^{-5}	-1.7×10^{-4}	9.0×10^{-5}	2.0×10^{-5}	-7.0×10^{-5}
$f_7(\mathbf{x})$	−1.03	−1.03	−0.85	0.00	0.18	0.18
$f_8(\mathbf{x})$	0.5	-5.8×10^{-5}	-1.1×10^{-6}	-5.9×10^{-5}	-1.1×10^{-6}	5.7×10^{-5}
$f_9(\mathbf{x})$	0.11	0.11	0.12	0.00	0.01	0.01
$f_{10}(\mathbf{x})$	−0.74	−0.69	−0.55	0.05	0.19	0.14
$f_{11}(\mathbf{x})$	0.00	0.00	-4.7×10^{-16}	0.00	-4.7×10^{-16}	-4.7×10^{-16}
$f_{12}(\mathbf{x})$	-6.3×10^{-3}	-9.9×10^{-4}	-5.1×10^{-3}	5.3×10^{-3}	1.2×10^{-3}	-4.1×10^{-3}
$f_{13}(\mathbf{x})$	0.00	0.00	-4.2×10^{-19}	0.00	-4.2×10^{-19}	-4.2×10^{-19}
$f_{14}(\mathbf{x})$	0.00	0.00	-6.8×10^{-18}	0.00	-6.8×10^{-18}	-6.8×10^{-18}
$f_{15}(\mathbf{x})$	0.00	0.00	-2.6×10^{-16}	0.00	-2.6×10^{-16}	-2.6×10^{-16}

Table 10
Contrast estimation based on medians among four best algorithms considered.

	H-CAGA	OGA/Q	HTGA	LEA
H-CAGA	0.000	-1.5×10^{-16}	-5.5×10^{-17}	-2.7×10^{-16}
OGA/Q	1.5×10^{-16}	0.000	9.5×10^{-16}	-1.2×10^{-16}
HTGA	5.5×10^{-17}	-9.5×10^{-16}	0.000	-2.2×10^{-16}
LEA	2.7×10^{-16}	1.2×10^{-16}	2.2×10^{-16}	0.000

Now, $M_1 - M_2 = m_1 - m_2 = -1.2 \times 10^{-16} - 3.0 \times 10^{-17} = -1.5 \times 10^{-16}$. In other words, the difference in optimum value between H-CAGA and OGA/Q algorithms computed over the median in multiple functions is equal to -1.5×10^{-16}. Table 10 shows the results of the four algorithms.

From Table 10, we can conclude that H-CAGA always obtains a negative difference value (for minimization) with respect to the other three methods compared, which indicates that it is the best performing method. HTGA is the second best. However, the performance of LEA is not better than OGA/Q as per Table 10 although LEA is the third best and OGA/Q is the fourth best if the average rank of Table 7 is considered.

4.4. Experiment on grid point pattern matching

To test the suitability of H-CAGA for pattern recognition, we have conducted our second experiment on two types of problems namely, point pattern matching and object recognition with edge map matching. A point pattern is a set of points in 2-D or 3-D space arranged to represent some physical objects or class of objects in the feature space.

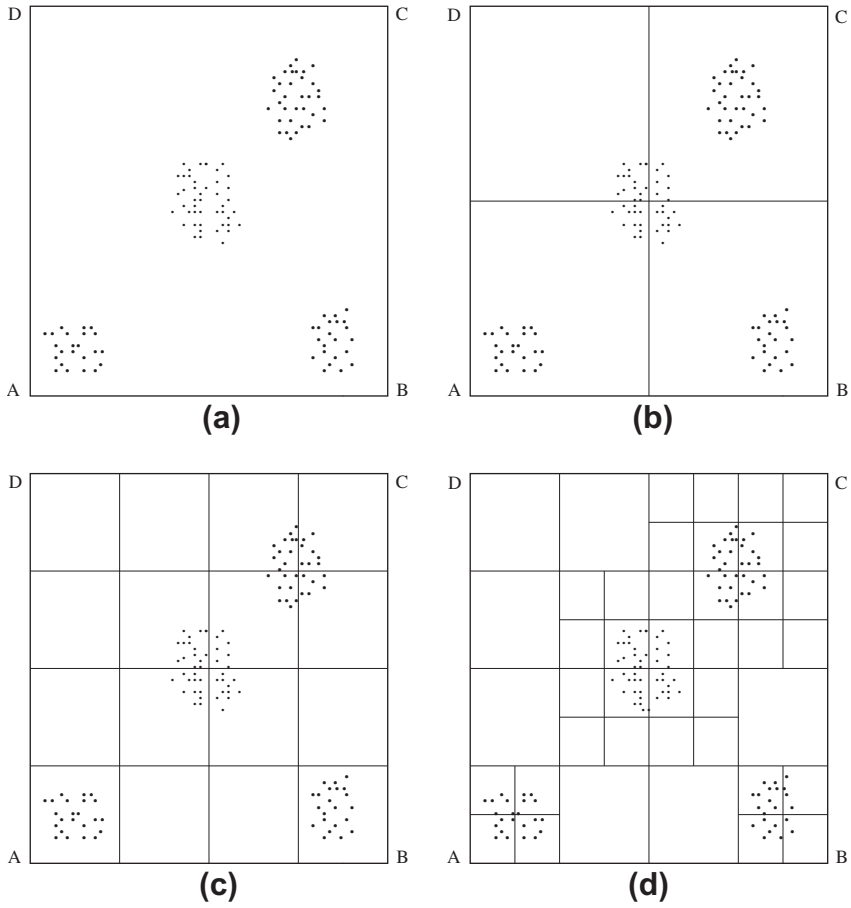

Fig. 3. Progress of the partitioning process using grid structure in 2-D space. (a) Pattern space before start of the partitioning process, (b) initial partitioning of the space, (c) repartition of the non-empty blocks in the intermediate stage of the process and (d) complete partitioning after checking all non-empty blocks.

Here, the proposed PPR algorithm (GOPM) is mainly dependent on H-CAGA for optimized pattern matching. Moreover, the GOPM is grid based pattern matching performing faster than other PPR techniques since less number of comparisons are required. Other two non-grid based PPR processes are dependent on the hybrid technique of the conventional GA and TS process for optimized pattern matching. While doing literature survey we have not encountered any grid based optimized PPR technique. The objective to compare these three PPR methods is to show their relative performance on point pattern recognition.

The point pattern matching problem can be posed as follows. Given a test point pattern T we have to identify its replica among a set of point patterns S in a scene. To do so, T is translated and rotated to find the position of best match in S. To translate T, a reference point, say the centroid O of the coordinates of the point of T is used. The translation and rotation is done with respect to O as origin.

Now, minimum Euclidean distance is used for matching of T with S. After transformation and rotation of T, 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 their minimum is taken. Finally, the sum of all such minimum distances is computed as follows.

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

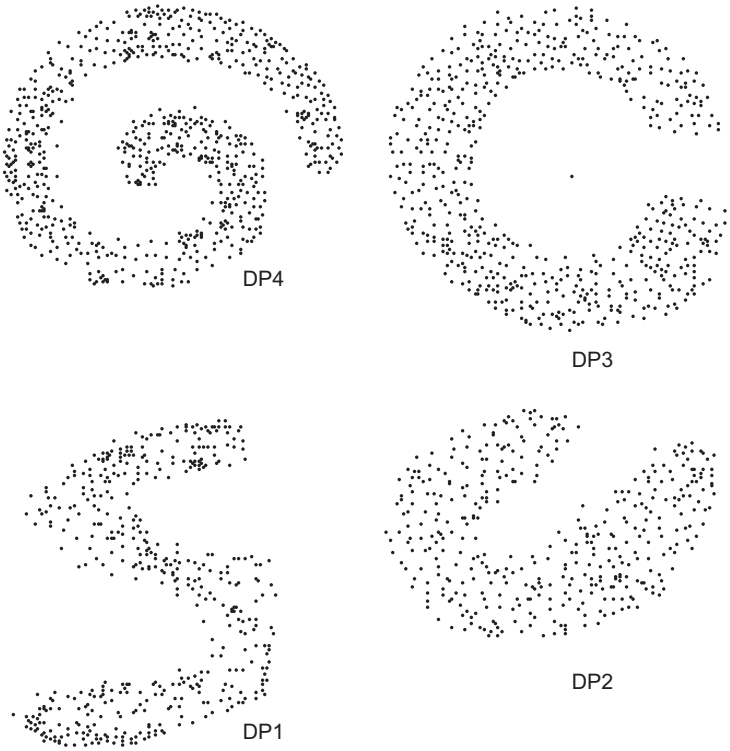

Fig. 4. A scene of multiple point patterns in 2-D space.

Table 11
Matching results of GOPM, OPM and OAPPM on Point Patterns in Fig. 4.

Point pattern	GOPM (%)			OPM (%)			OAPPM (%)		
	SM		UM	SM		UM	SM		UM
	PM	VM	NM	PM	VM	NM	PM	VM	NM
DP1	53	47	0	14	76	10	23	67	10
DP2	60	40	0	15	67	18	21	65	14
DP3	40	60	0	20	60	20	25	65	10
DP4	83	17	0	21	79	0	27	73	0

The above data have been summarized over 50 runs for each point pattern. In each run the space is redefined four times. SM: successful matching, UM: unsuccessful matching, PM: perfectly matched, VM: visually matched, NM: not matched.

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 point patterns or objects for a solution in a population.

In the grid pattern matching problem the space is initially partitioned into 2^d blocks or grid cells by a grid structure where d is the dimension of the space. Each cell contains either a finite number of points or remains empty. Each block is then examined and if any block contains more points than a prespecified value (10% of total points in a scene of pattern), then the same partitioning process is run on the corresponding block. Otherwise, the process is moved to other untested blocks until all blocks are encountered. The partitioning process advances in a hierarchical way which is depicted in Fig. 3. At the end, the non-empty grid cells (say, δ) are only considered for pattern matching problem.

In conventional PPR approaches the time complexity for pattern matching according to Eq. (9) is $O(\alpha\beta)$. However, in grid pattern matching the computation time is reduced to a great extent since GOPM does not compare each point of T with each point of S. Instead, each t_i of T is first compared with δ non-empty grid cells to first detect its position in a grid cell. Once it is done, t_i is compared with each point of the identified grid cell to find its nearest neighbor. If each grid cell contains an average of η points, then the time complexity to evaluate the matching score for GOPM is $O(\alpha(\delta + \eta))$. Since both δ and $\eta \ll \beta$, it is now easily understood that the computation time of pattern matching for GOPM is better than that for other relevant pattern matching methods.

The performance of GOPM has been tested with two hybrid GA based methods only on matching score and not on the computation time since other methods are not based on grid partitioning. One of them is Optimized Patten Matching with CGA (named as OPM) and the second one is Optimized method for Affine Point Pattern Matching (called OAPPM). We have slightly altered APPMGA [26] to suit it with our pattern matching problem. The process is basically a sequential one which is

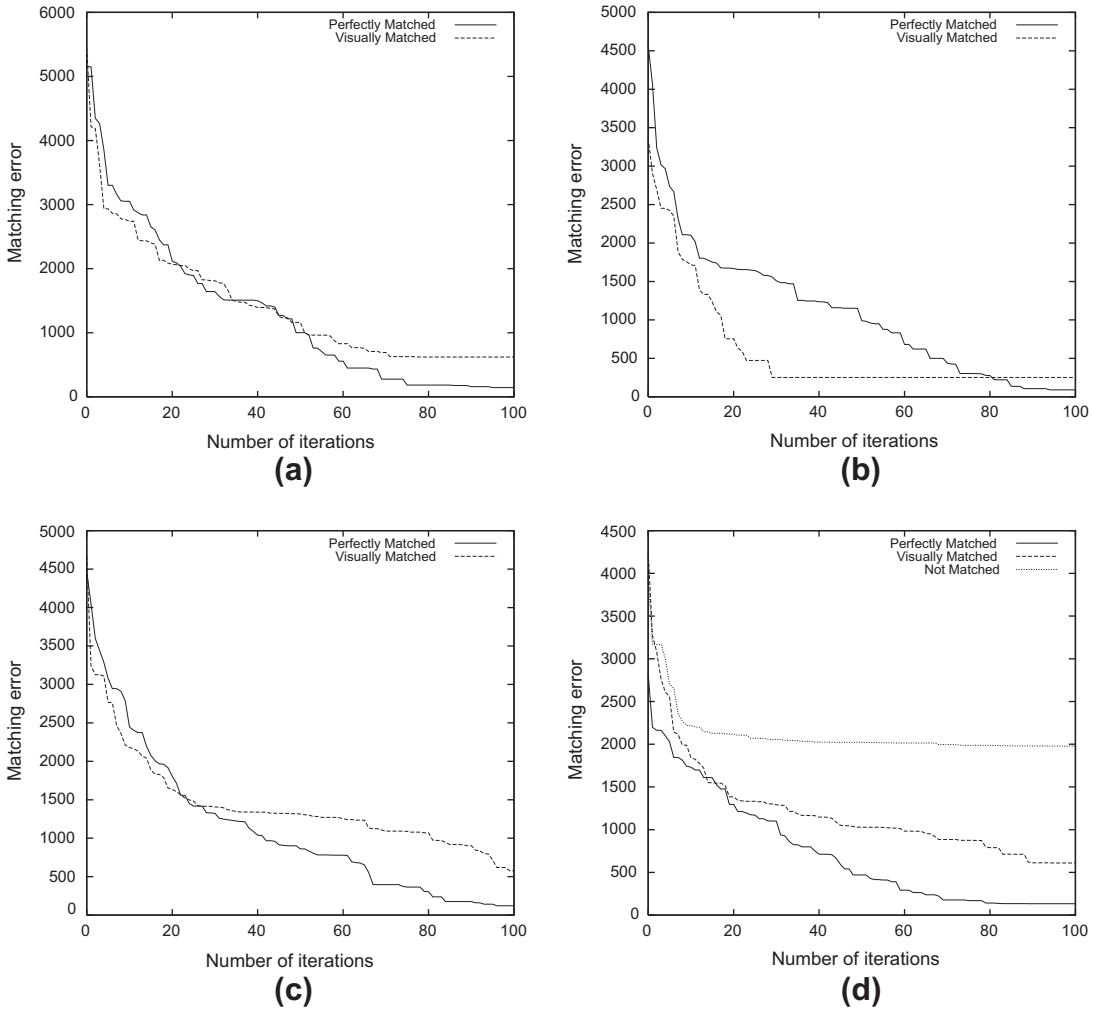

Fig. 5. Matching error vs. number of iterations for the patterns in Fig. 4. (a) Best case matching results of GOPM for pattern DP4, (b) worst case matching results of GOPM for pattern DP3, (c) best case matching results of OPM for pattern DP4 and (d) worst case matching results of OPM for pattern DP3. All matching results are averaged over 50 runs.

Table 12
Matching results of GOPM, OPM and OAPPM on non-overlapped Objects with edge map in Fig. 7.

Point pattern	GOPM (%)			OPM (%)			OAPPM (%)		
	SM		UM	SM		UM	SM		UM
	PM	VM	NM	PM	VM	NM	PM	VM	NM
P1	82	13	5	6	30	64	9	40	51
P2	47	50	3	3	13	84	15	35	50
P3	47	48	5	0	15	85	0	25	75
P4	70	27	3	7	27	66	17	33	50

The above data have been summarized over 50 runs for each point pattern. In each run the space is redefined four times. The terms SM, UM, PM, VM and NM are same as described in Table 11.

Fig. 6. Edge map of the image, Lena.

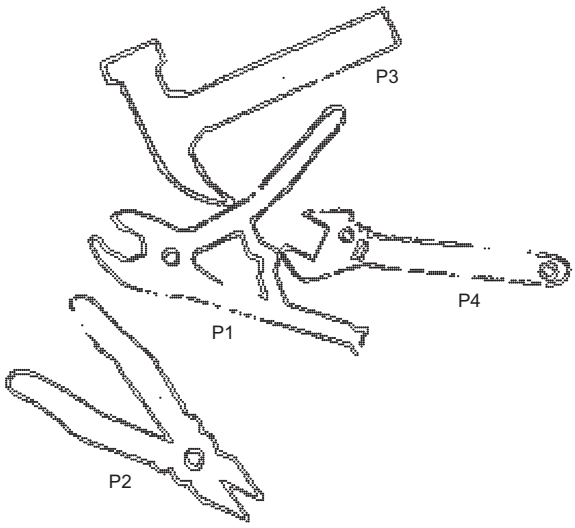

Fig. 7. A scene of multiple non-overlapped objects with edge map in 2-D space.

converted to a hybrid process so that the other two hybrid approaches do not get any undue advantage. The OAPPM approach considers the following rotation and translation matrices for the affine transformation of the test data set.

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

We have changed the elements of the rotation 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 angle of rotation. Moreover, the fitness function of APPMGA is dependent on *Hausdorff distance*. However, we have changed to *Euclidean distance* in the experiment. We have chosen this distance since it collects contribution from each and every point of the test pattern data. On the other hand, *Hausdorff* distance is based on *max* of *min* distance between points of one set to the other. It is a single contribution that may be drastically different if a single outlier is present.

In point pattern or object matching problem S is a set of point patterns of different shapes as shown in Figs. 4, 7 and 9 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 GOPM, OPM and OAPPM are depicted in Tables 11–13, respectively. 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 consider two types of matching namely *perfect matching* and *imperfect* but *visually acceptable matching*. For *visually matched patterns* we should observe that T is superimposed over S without any visible error, although the matching error is computationally reasonable. On the other hand, in *perfectly matched situation* the computed error is very close to zero.

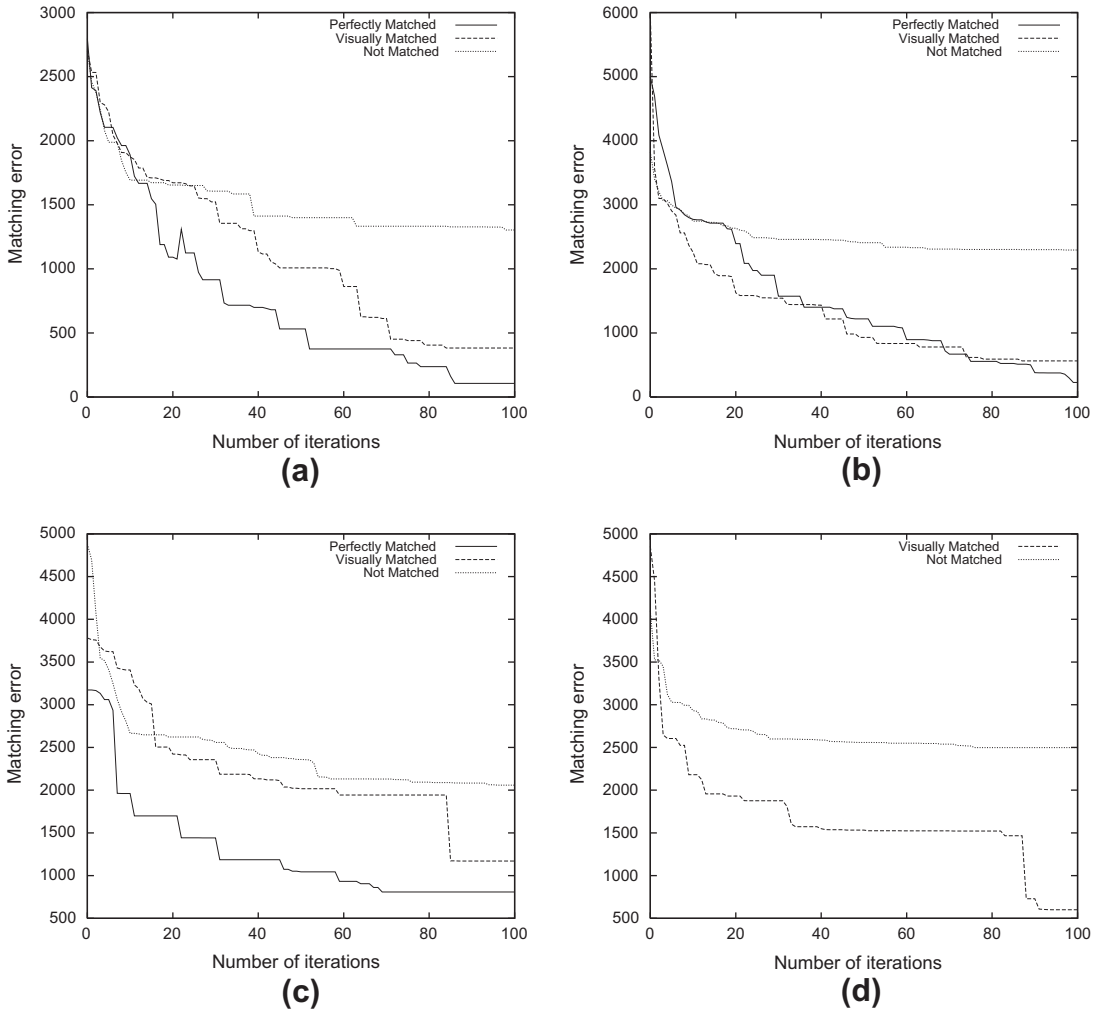

Fig. 8. Matching error vs. number of iterations for the patterns in Fig. 7. (a) Best case matching results of GOPM for pattern P1, (b) worst case matching results of GOPM for pattern P3, (c) best case matching results of OAPPM for pattern P1 and (d) worst case matching results of OAPPM for pattern P3. All matching results are averaged over 50 runs.

From Table 11 we notice that for GOPM the success rate is 100% for all four patterns considering perfect and visual matching. The best performance of GOPM is achieved for DP4 although the unsuccessful matching score for the remaining three patterns is 0. On the other hand, the failure rate is at most 20% in the worst case for OPM. However, it can also achieve a success rate of 100% while matching the pattern DP4. The performance of OAPPM is almost equivalent to or better than OPM. Fig. 5 shows the change of matching error of GOPM and OPM with respect to the number of iterations. It is noted from Fig. 5 that the error difference between perfect matching and visual matching is not too large for GOPM. Since the performance of OAPPM is similar to that of OPM, we have not included it in Fig. 5.

The other experiment done here is matching of edge maps. Here the object in Fig. 6 is a gray-tone facial image of Lena and in Figs. 7 and 9 the scenes are a combination of four different objects. Using *Sobel Operator* [15] all images are converted into two tone edge maps from the original gray-tone images. The edge map may be considered as a discrete point pattern where a point is represented by *1* and a blank (white space) by *0* (see Figs. 6, 7 and 9). Fig. 6 is the edge map of the Lena image. In the edge map of the multi-object scene (see Fig. 7) the objects (P1, P3 and P4) are very close and touch one another. In Fig. 9 all four objects (P1, P2, P3 and P4) are overlapped.

In case of Fig. 6 the object S is the original one and T is the test pattern (a displaced form of S). The pattern T is displaced by transformation and/or translation of the original object S. In the experiment the test result of the proposed method (GOPM) shows that the success rate of matching is 93% considering perfectly as well as visually matched patterns. The unsuccessful matching score is 7%. On the other hand, for other two genetic methods (OPM and OAPPM) the successful matching score is 72% and 75%, respectively.

Table 13
Matching results of GOPM, OPM and OAPPM on overlapped Objects with edge map in Fig. 9.

Point pattern	GOPM (%)			OPM (%)			OAPPM (%)		
	SM		UM	SM		UM	SM		UM
	PM	VM	NM	PM	VM	NM	PM	VM	NM
P1	70	30	0	9	40	51	15	40	45
P2	53	47	0	7	27	66	15	35	50
P3	60	40	0	13	15	72	20	25	55
P4	70	30	0	7	37	56	17	45	38

The above data have been summarized over 50 runs for each point pattern. In each run the space is redefined four times. The terms SM, UM, PM, VM and NM are same as described in Table 11.

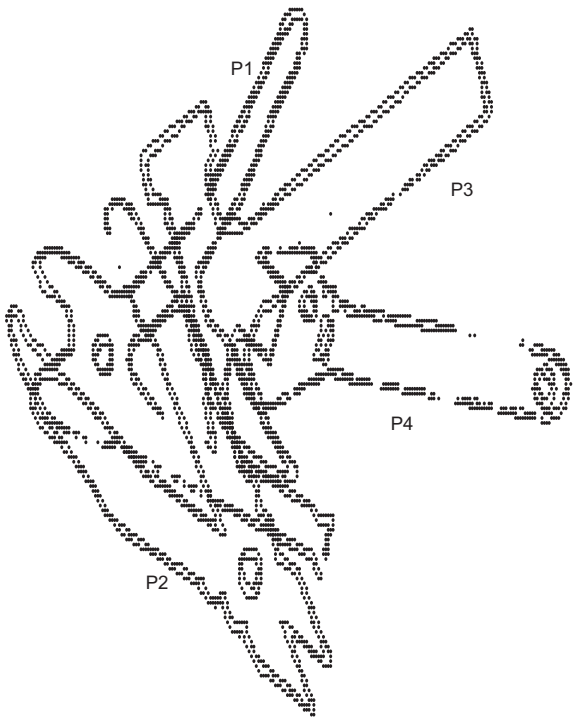

Fig. 9. A scene of multiple overlapped objects with edge map in 2-D space.

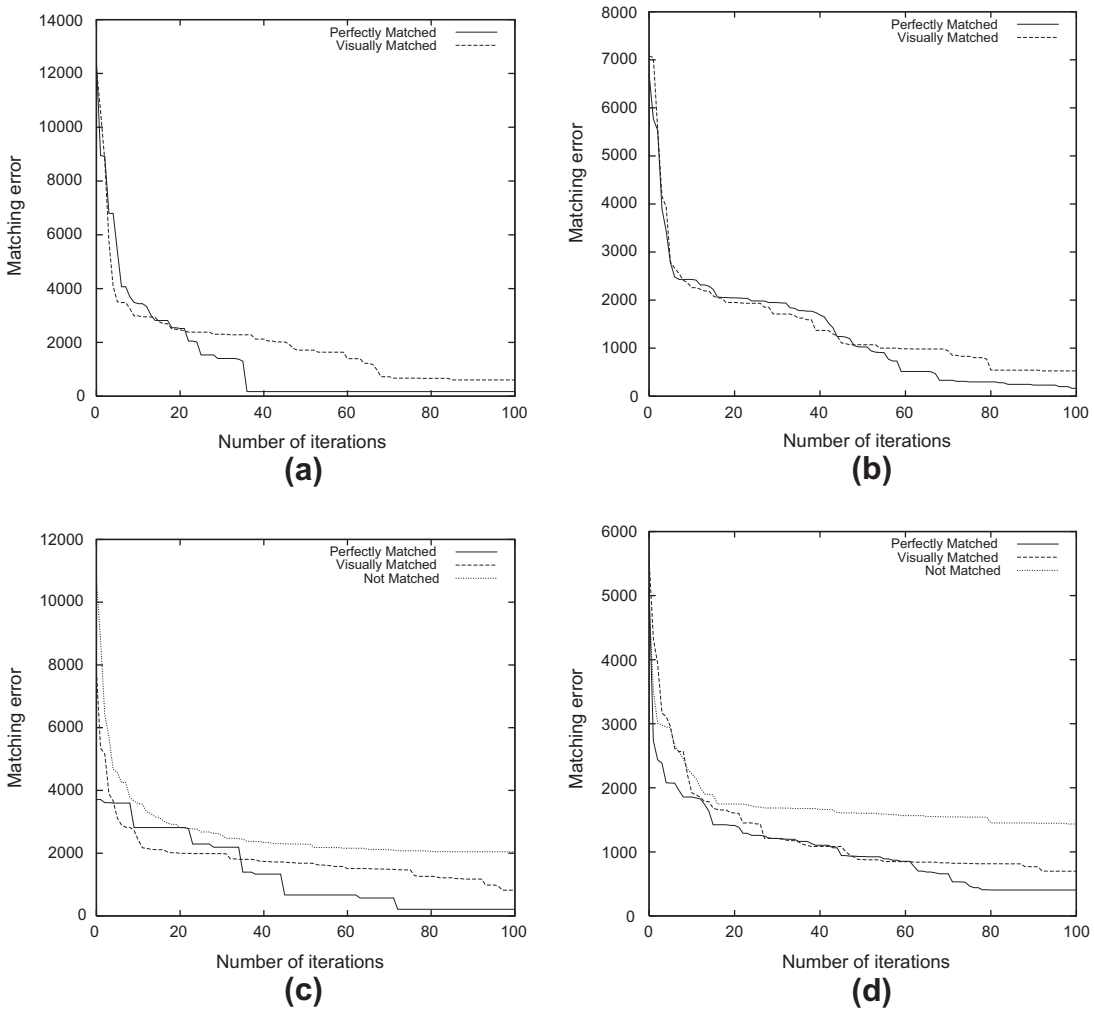

Fig. 10. Matching error vs. number of iterations for the patterns in Fig. 9. (a) Best case matching results of GOPM for pattern P1, (b) worst case matching results of GOPM for pattern P2, (c) best case matching results of OPM for pattern P1 and (d) worst case matching results of OPM for pattern P3. All matching results are averaged over 50 runs.

The matching results of the objects in Figs. 7 and 9 are tabulated in Tables 12 and 13, respectively. It is observed that the success rate of GOPM is not always 100% for pattern in Fig. 7. The proposed approach achieves best performance for P4 with perfect match 70 times out of 100 runs. However, the performance of other two genetic techniques is best for matching pattern P1. Fig. 8 shows the matching error vs. number of iterations for GOPM and OAPPM. Here we have not included OPM in Fig. 8 as its performance is similar to OAPPM. Also, in the worst case of OAPPM, there is no perfect match for P3 like the OPM.

The pattern in Fig. 9 is more complex than that in Fig. 7 but GOPM achieves 100% successful matching. GOPM provides the best result for patterns P1 and P4. Similarly, OPM obtains best result for P1 and OAPPM gets the best for P4. It is also noted that the performance of GOPM is worse when P2 is isolated. The overlapped patterns although complex enhances the performance of the grid based approach. Fig. 10 depicts the matching error vs. number of iterations for GOPM and OPM. Here, we have also not included the graphical results of OAPPM for similar reason described before.

5. Discussions with conclusion

The performance of H-CAGA on *twenty* high and low dimensional benchmark functions shows its improvement over other relevant methods. It has the capability to reach the global optimum for most of the problems having dimension 100 by slowing down the partitioning process. On the other hand, the non-parametric test for statistical analysis also shows that H-CAGA performs better than the other optimizing methods. The point pattern matching approach depends on the matching score of two point patterns by finding an optimal transformation. The efficiency of the proposed pattern matching technique (GOPM)

is enhanced with the dual applications of hybrid cascaded GA and grid based approach. The grid based approach helps to match much lesser number of points for entire pattern matching.

In the experiment it was noticed that the success rate of the genetic techniques for pattern matching improves with the increase of the degree of overlapping objects in the pattern. The best result is usually achieved by any of these techniques for the centrally located P1 pattern.

We have performed our pattern matching experiment on some synthetic point patterns except the image of Lena. We can extend our test for some real point patterns in future. There is also a scope for future work to increase the dimension of the mathematical functions beyond 100 to examine the limits of the slow partitioning process.

Acknowledgments

The authors wish to sincerely thank the anonymous referees for their precious comments and constructive criticism on the previous version of this article. We would also like to thank R. Banerjee of C&MB Division, SINP for his help in preparation of this manuscript.

References

[1] T.S. Caetano, T. Caelli, D. Schuurmans, D.A.C. Barone, Graphical models and point pattern matching, IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (10) (2006) 1646–1663.
[2] M. Carcassoni, E.R. Hancock, Spectral correspondence for point pattern matching, Pattern Recognition 36 (2003) 193–204.
[3] J. Demsar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research 7 (2006) 1–30.
[4] J.A. Denton, J.R. Beveridge, An algorithm for projective point matching in the presence of spurious points, Pattern Recognition 40 (2007) 586–595.
[5] K. Doksum, Robust procedures for some linear models with one observation per cell, Annals of Mathematical Statistics 38 (1967) 878–883.
[6] M. Friedman, The use of ranks to avoid the assumption of normality implicit in the analysis of variance, Journal of the American Statistical Association 32 (1937) 674–701.
[7] G. Garai, B.B. Chaudhuri, A cascaded genetic algorithm for efficient optimization and pattern matching, Image and Vision Computing 20 (2002) 265–277.
[8] S. Garcia, A. Fernandez, J. Luengo, F. Herrera, Advanced non-parametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power, Information Sciences 180 (2010) 2044–2064.
[9] F. Glover, Tabu search – Part I, ORSA Journal of Computing 1 (3) (1989) 190–206.
[10] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, New York, 1989.
[11] D. Goldberg, S. Voessner, Optimizing global–local search hybrids, in: W. Banzhaf, J. Daida, A. Eiben, M. Garzon, V. Honavar, M. Jakaiela, R. Smith (Eds.), GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference, Morgan Kaufmann, 1999. pp. 220–228.
[12] C.S. Hong, Z. Quan, Integral Global Optimization: Theory, Implementation and Application, Springer-Verlag, Berlin, Germany, 1988.
[13] R.L. Iman, J.M. Davenport, Approximations of the critical region of the friedman statistics, Communications in Statistics 9 (1980) 571–595.
[14] H. Ishibuchi, T. Yamamoto, T. Nakashima, Hybridization of fuzzy GBML approaches for pattern classification problems, IEEE Transactions on Systems, Man and Cybernetics – Part B 35 (2) (2005) 359–365.
[15] A.K. Jain, Fundamentals of Digital Image Processing, Prentice-Hall, Englewood Cliffs, 1989.
[16] T. Jiang, F. Yang, An evolutionary Tabu search for cell image segmentation, IEEE Transactions on Systems, Man and Cybernetics – Part B: Cybernetics 32 (5) (2002) 675–678.
[17] C.Y. Lee, X. Yao, Evolutionary programming using mutations based on the levy probability distribution, IEEE Transactions on Evolutionary Computation 8 (1) (2004) 1–13.
[18] Y.W. Leung, Y.P. Wang, An orthogonal genetic algorithm with quantization for global numerical optimization, IEEE Transactions on Evolutionary Computation 5 (1) (2001) 41–53.
[19] J. Li, G. Shi, Q. Fenng, H. Wan, A new point pattern matching method for palm point, in: Proc. of Intl. Conf. on Artificial Intelligence and Pattern Matching (AIPR07), Orlando, Florida, 9–12 July, 2007, pp. 302–306.
[20] C.A. Perez, C.A. Salinas, P.A. Estevez, P.A. Valenzuela, Genetic design of biologically inspired receptive fields for neural pattern recognition, IEEE Transactions on Systems, Man and Cybernetics – Part B 33 (2) (2003) 258–270.
[21] A.L. Rhyne, R.G.D. Steel, Tables for a treatments versus control multiple comparisons Sign test, Technometrics 7 (1965) 293–306.
[22] J.-T. Tsai, T.-K. Liu, J.-H. Chou, Hybrid Taguchi-genetic algorithm for global numerical optimization, IEEE Transactions on Evolutionary Computation 8 (2004) 365–377.
[23] Y. Wang, C. Dang, An evolutionary algorithm for global optimization based on level-set evolution and latin squares, IEEE Transactions on Evolutionary Computation 11 (5) (2007) 579–595.
[24] X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, IEEE Transactions on Evolutionary Computation 3 (2) (1999) 82–102.
[25] P.-Y. Yin, Particle swarm optimization for point pattern matching, Journal of Visual Communication & Image Representation 17 (2006) 143–162.
[26] L. Zhang, W. Xu, C. Chang, Genetic algorithm for point pattern matching, Pattern Recognition Letters 24 (2003) 9–19.