Available online at www.sciencedirect.com

ScienceDirect

Information Sciences 177 (2007) 4295–4313

INFORMATION SCIENCES
AN INTERNATIONAL JOURNAL

www.elsevier.com/locate/ins

A self-adaptive migration model genetic algorithm for data mining applications ☆

K.G. Srinivasa [a,*], K.R. Venugopal [a], L.M. Patnaik [b]

[a] *Department of Computer Science and Engineering, University Visvesvaraya College of Engineering, Bangalore 560001, India*
[b] *Microprocessor Applications Laboratory, Indian Institute of Science, Bangalore 560012, India*

Received 5 January 2006; received in revised form 1 May 2007; accepted 2 May 2007

Abstract

Data mining involves nontrivial process of extracting knowledge or patterns from large databases. Genetic Algorithms are efficient and robust searching and optimization methods that are used in data mining. In this paper we propose a Self-Adaptive Migration Model GA (SAMGA), where parameters of population size, the number of points of crossover and mutation rate for each population are adaptively fixed. Further, the migration of individuals between populations is decided dynamically. This paper gives a mathematical schema analysis of the method stating and showing that the algorithm exploits previously discovered knowledge for a more focused and concentrated search of heuristically high yielding regions while simultaneously performing a highly explorative search on the other regions of the search space. The effective performance of the algorithm is then shown using standard testbed functions and a set of actual classification datamining problems. Michigan style of classifier was used to build the classifier and the system was tested with machine learning databases of Pima Indian Diabetes database, Wisconsin Breast Cancer database and few others. The performance of our algorithm is better than others.

Keywords: Adaptive genetic algorithms; Optimization; Data mining; Machine learning

1. Introduction

Data mining is a process of extracting nontrivial, valid, novel and useful information from large databases. Hence Data mining can be viewed as a kind of search for meaningful patterns or rules from a large search space, that is the database. In this light, Genetic Algorithms are a powerful tool in data mining, as they are robust search techniques. Genetic Algorithms (GA) are a set of random, yet directed search techniques.

☆ This work is partially supported by AICTE, New Delhi, under AICTE Career Award for Young Teachers to K.G. Srinivasa, Assistant Professor, Dept. of CSE, M S Ramaiah Institute of Technology, Bangalore.
* Corresponding author.
E-mail addresses: kgsrinivas@msrit.edu (K.G. Srinivasa), venugopalkr@gmail.com (K.R. Venugopal), lalit@micro.iisc.ernet.in (L.M. Patnaik).

0020-0255/$ - see front matter
doi:10.1016/j.ins.2007.05.008

They process a set of solutions simultaneously and hence are parallel in nature. They are inspired by the natural phenomenon of evolution. They are superior to 'gradient descent' techniques as they are not biased towards local optima.

Genetic Algorithms have found a wide gamut of application in data mining, where knowledge is mined from large databases. Genetic algorithms can be used to build effective classifier systems [16,4], mining association rules [17,7] and other such datamining problems. Their robust search technique has given them a central place in the field of data mining and machine learning.

GA can be viewed as an evolutionary process where at each generation, from a set of feasible solutions, individuals or solutions are selected such that individuals with higher fitness have greater probability of getting chosen. At each generation, these chosen individuals undergo crossover and mutation to produce a population of the next generation. This concept of survival of the fittest proposed by Darwin is the main cause for the robust performance of GAs. Crossover helps in the exchange of discovered knowledge in the form of genes between individuals and mutation helps in restoring lost or unexplored regions in search space. The steps in genetic algorithm are:

1.1. Genetic algorithm( )

```
{
Initialize population randomly;
Evaluate fitness of each individual in the population;
While stopping condition not achieved
   {
      Perform selection;
      Perform crossover and mutation;
      Evaluate fitness of each individual in the population;
   }
}
```

In this algorithm, the three basic operators of GA namely, selection, crossover and mutation operators are fixed apriori. The optimum parameters for these operators depend on problem on which the GA is applied and also on the fitness of the current population. A new breed of GA called adaptive GAs [31,3,15], fix the parameters for the GA operators dynamically to adapt to the current problem. Generally the operators are adapted based on the fitness of individuals in the population. Apart from the operators themselves, even the control parameters can be adapted dynamically. In these adaptive genetic algorithms, the GA parameters are adapted to fit the given problem. The algorithm is as follows.

1.2. Adaptive genetic algorithm( )

```
{
Initialize population randomly;
Evaluate fitness of each individual in the population;
While stopping condition not achieved
   {
      Perform selection;
      Perform crossover and mutation;
      Evaluate fitness of each individual;
      Change selection, crossover and mutation operators.
   }
}
```

The use of parallel systems in the execution of genetic algorithms has led to parallel genetic algorithms. The Island or Migration model of genetic algorithm [23] is one such genetic algorithm where, instead of one

population, a set of populations is evolved. In this method, at each generation all the populations are independently evolved and at some regular interval fixed by migration rate, few of the best individuals are exchanged among populations.

2. Related work

Lobo et al. have proposed an adaptive GA which works even when optimal number of individuals is not known [22]. In this method parallel searches are conducted with different number of individuals, expecting one of the populations to have the appropriate number of individuals that yields good results. However, this method is not truly adaptive in the sense that the appropriate number of individuals is not learnt but is obtained by trial and error. This method is not feasible as it is not realistic to perform large number of blind searches in parallel. On similar way, some of the applications of the parameter-less genetic algorithms [11] and multi-objective rule mining using genetic algorithms are discussed in [1].

An adaptive GA which runs three GAs in parallel is proposed in [14]. Here at each *epoch* fitness values of elite individuals is compared and the number of individuals are changed according to the results. For example, if GA with the largest number of individuals provides best results, then in the next epoch all individuals have large number of individuals. However, the optimum number of individuals required by a population depends on which region of the search space the individuals are in and is not same for all subpopulations.

An adaptive GA where mutation rate for an individual is encoded in the gene of an individual is proposed in [3]. The system was proposed with the hope that finally individuals with good mutation rate survive. However, only individuals with low mutation rate survive in the later phases of the search. An adaptive GA that determines mutation and crossover rate of an individual by its location in a two dimensional lattice plane is proposed in [21]. The algorithm keeps diversity of these parameters by limiting the number of individuals in each lattice.

A meta-GA is a method of using GA to fix the right parameters for the GA. However, in this process the number of evaluations needed is high and the process is a costly one. One such meta-GA is proposed in [20]. A GA that adapts mutation and crossover rates in Island model of GA is proposed in [33]. Here adaptation is based on average fitness of the population. Parameters of a population here are updated to those of a neighboring population with high average fitness.

The breeder genetic algorithm BGA depends on a set of control parameters and genetic operators. It is shown that strategy adaptation by competing subpopulations makes the BGA more robust and more efficient in [28]. Each subpopulation uses a different strategy which competes with other subpopulations. Experiments on multi-parent reproduction in an adaptive genetic algorithm framework is performed in [9]. An adaptive mechanism based on competing subpopulations is incorporated into the algorithm in order to detect the best crossovers. A parallel genetic algorithm with dynamic mutation probability is presented in [18]. This algorithm is based on the farming model of parallel computation. The basic idea of the dynamic establishing mutation rate is presented. Similarly,an adaptive parallel genetic algorithm for VLSI Layout Optimization is discussed in [29]. A major problem in the use of genetic algorithms is premature convergence. An approach for dealing with this problem is the distributed genetic algorithm model is addressed in [10]. Its basic idea is to keep, in parallel, several subpopulations that are processed by genetic algorithms, with each one being independent of the others. But all these algorithms either consider mutation rate or crossover rate as a dynamic parameter, but not the both at the same time. The application of a breeder genetic algorithm to the problem of parameter identification for an adaptive finite impulse filter is addressed in [25]. A population-based optimization method, called Free Search (FS) is also discussed in [19].

A Self-adaptive Genetic Algorithms with Simulated Binary Crossover is discussed in [6]. It talks about a single population GA for starters and it does not have a multipoint crossover let alone adaptive multipoint crossover. The only similarity between [6] and proposed model is that both the methods chooses adaptively the proportion of the population for crossover. Similarly an adaptive dynamic crossover operators based on fuzzy connectives and adaptive real coded GAs based on fuzzy logic controller is discussed in [13]. But it fails to prove the model analytically and the convergence rate of our proposed model(SAMGA) is better.

The Nix and Vose Punctuated Equilibria is given in [34]. Markov Modelling for Genetic algorithms and their convergence rate are discussed in [24,2,8,12,27]. The average hamming distance of individuals in the

population to derive convergence rate is used in [32]. Genetic Algorithms based on binomially distributed populations is proposed in [30]. In this paper the definition of convergence is restated and convergence rate is derived based on expectation value of individuals.

3. Overview

Any heuristic search can be characterized by two concepts, exploitation and exploration. These concepts are often conflicting in the sense that if exploitation of a search is increased, then exploration decreases and vice versa. Fortunately, the parallel nature of GA helps in alleviating this problem slightly by simultaneously processing a set of solutions thus providing for exploration, while at the same time survival of the fittest enforces exploitation. However, if a single population processes a set of individuals, then the schemas in the population that give better results only survive as generations proceed and hence search is not very explorative. Further, if the explorative power of search is increased too much using mutation, convergence does not occur. Thus, we need a method wherein, the explorative power of search is increased for individuals in bad region of the state space and exploitation power of search is increased for individuals in the high fitness region of the state space.

In this paper, the proposed adaptive migration(island) model of GA is built such that, given a search space, the number of individuals in the population that resides in a relatively high fitness region of the search space increases thus improving exploitation. For these high fitness population, the mutation rate and number of points of crossover are decreased thus making the search more focused. On the other hand, for populations in a relatively low fitness zone of search space, the number of individuals is decreased but the mutation rates and number of points of crossover are increased to make the search of these regions more explorative. Thus, the search can be termed as one which exploits the heuristically promising region of search space while exploring other regions. The search is also competitive as an increase in fitness of one population makes the search in other populations with lower fitness more explorative and rigorous.

4. Algorithm

The genetic algorithm described here is an adaptive GA where between evaluation of individuals and applying the three operators of GA (selection, crossover and mutation), the parameters used for these operators are updated based on the evaluation of individuals.

4.1. Problem definition

Let S be made up of all the 2^k, k bit binary numbers representing the entire search space. Given the objective function f which is a mapping $f: S \to R$ where R is a set of real numbers, the problem is to find an $x^* \in S$ such that $f(x^*) \geqslant f(x)\ \forall x \in S$.

4.2. Pseudocode

Let $E = \{P_1, P_2 \ldots, P_{np}\}$ be an ecosystem with np populations in it. Populations $P_1, P_2 \ldots, P_{np} \subset S$. $P_i[j]$ stands for the jth individual of population P_i, clearly $P_i[j] \in S$. Let n_i be the size of population P_i and nc_i be the number of points of crossover used for reproduction in population P_i. Let $\bar{f}_i$ be the average fitness of a population P_i. Let $f(P_i)$, the fitness of population P_i be the fitness of the best individual of that population. Let $\bar{f}$ be the average fitness of the ecosystem. Let $\bar{n}$ be the average number of individuals per population. Let pm_i be the rate of mutation used for population P_i. The rate of crossover being one. Then, the pseudocode for the adaptive GA can be given as below

1. **begin**
2. *var* prev = 0;
3. **for** i = 1: *np*, **do**
   (a) Set n_i, the number of individuals in the population P_i to some arbitrary value n_0
   (b) Initialize all individuals of population P_i to some random bit strings

(c) Set number of crossover points used for population P_i, nc_i to one
(d) Set mutation rate used for population P_i, pm_i to 0.01
4. **next**
5. **for** gen = 1: maximum generation limit, **do**
(a) *var* nsum = 0;
(b) *var* fsum = 0;
(c) **for** i = 1: *np*, **do**
i. Evaluate fitness of all the individuals of the population P_i and find $f(P_i)$ the best fitness of the population
ii. $nsum = nsum + n_i$
iii. $fsum = fsum + f(P_i)$
(d) prev = $\bar{f}$
(e) $\bar{f} = \frac{fsum}{np}$
(f) $\bar{n} = \frac{nsum}{np}$
(g) **for** i = 1: *np*, **do**
i. $nc_i = nc_i + \frac{\bar{n}}{n} - 1$
ii. $pm_i = pm_i + \left(\frac{\bar{n}}{n} - 1\right) * 0.0001$
iii. $n_i = n_i + \frac{f(P_i)}{\bar{f}} - 1$
iv. **if** $(n_i == 0)$
Delete population P_i (extinction)
v. **endif**
(h) **next**
(i) **for** i = 1: *np*, **do**
i Perform elitist selection for population P_i with the modified population size n_i
ii Perform *nc* point non-uniform crossover on the selected individuals of population P_i
iii Perform mutation on the individuals of population P_i with mutation probability pm_i
(j) **next**
(k) **if** prev==$\bar{f}$
Exchange or migrate best individuals between populations.
(l) **endif**
6. **next**
7. **end**

The algorithm shown above is adaptive in four respects. The first parameter adaptively changed is the population size of each population. The population size is dynamically varied based on the fitness of the best individual of that population compared to the mean fitness of the population. The number of individuals in population P_i is updated as,

$$n_{i,t+1} = n_{i,t} + \frac{f(P_i)}{\bar{f}} - 1 \tag{1}$$

where t is used to represent time in generations. Using this update, the size of population with fitness greater than the mean population fitness grows while size of population whose fitness is below the mean population fitness shrinks. Thus, it can be visualized that more number of individuals are concentrated in the heuristically better promising region of search space (exploitation).

The second parameter that is dynamically adapted is the number of points of crossover. The update used for number of crossover points is given by

$$nc_{i,t+1} = nc_{i,t} + \frac{\bar{n}}{n_i} - 1 \tag{2}$$

Using this update we can see that, if the number of individuals in a population is less compared to the mean number of individuals in a population, then the number of points of crossover is increased. In an indirect way

update on the number of points of crossover is linked to fitness of population. From update on population size, it is clear that population size increases with fitness of population but from update of number of points of crossover, we see that for relatively low population sizes the number of points of crossover is increased and hence search is more explorative.

The third parameter considered is the mutation rate whose update is similar to the update on the number of crossover points, given by,

$$pm_{i,t+1} = pm_{i,t} + \left(\frac{\bar{n}}{n_i} - 1\right) * 0.0001 \tag{3}$$

The significance of this update is similar to the update on the number of points of crossover. Obviously, higher the mutation rate, more explorative is the search. The factor of 0.0001 is chosen arbitrarily as it is a considerably small factor to update probability of mutation.

The final parameter that is adaptive in the algorithm is the rate of migration. Migration refers to copying individuals from one population to another. Migration helps in discovering new schemas got by crossover of schemas of two populations. In the algorithm, there is no exact parameter for migration rate. Migration occurs when the average fitness of the populations remains unchanged between two generations. Thus, when populations have attained a steady state, migration occurs to try and discover a new schema.

The selection scheme used in the genetic algorithm is the elitist scheme. The best individuals of each population are copied unaltered to the next generation population. The remaining individuals are selected based on their fitness. The use of elitist selection guarantees that the best individual of any generation is at least as good as the best individual of the previous generation. It helps in achieving global convergence. Here, 100 chosen from the k most fit but k is large to prevent premature convergence.

One interesting phenomenon that can be noticed when this algorithm is used on a complex, search space, is that just like in nature, if a population consistently performs badly, its number finally decreases to zero and the population becomes extinct. This suggests that the total number of individuals in the ecosystem is affected by the problem size and complexity.

5. Mathematical analysis

The Holland's schema theorem for a general case can be given as,

$$M(H,t+1) \geqslant \left((1-p_c)M(H,t)\frac{f(H)}{f_{avg}} + p_c\left[M(H,t)\frac{f(H)}{f_{avg}}(1-losses) + gains\right]\right)(1-p_m)^{O(H)}$$

where $M(H,t)$ is the number of individuals in a population with schema H are present in the current generation, f_{avg} is average fitness of population, $O(H)$ is order of schema H and p_c and p_m are rates of crossover and mutation respectively. In our algorithm, we consider p_c to be one. Hence the Schema theorem becomes,

$$M(H,t+1) \geqslant \left[M(H,t)\frac{f(H)}{f_{avg}}(1-losses) + gains\right](1-p_m)^{O(H)}$$

In the proposed algorithm, as the population size varies each generation, the schema lower bound for the varying population size becomes,

$$M(H,t+1) \geqslant \left[\frac{M(H,t)}{n_t}n_{t+1}\frac{f(H)}{f_{avg}}(1-losses) + gains\right](1-p_m)^{O(H)} \tag{4}$$

where n_t is population size at generation t.

If we consider loss to be any crossover that disrupts the schema, then our calculation of gain must account for the preservance of the schema when both parents are of the schema H. Now for an n point crossover to be non-disruptive, even number of crossover points, only can occur between fixed bits of schema [5] as shown in

Fig. 1. The remaining crossover points must be outside defining length. Hence the probability of n point crossover generating only even number of crossovers between fixed bits for a schema of k order hyperplane is $P_{k,\text{even}}$. Probability of disruption P_d of n point crossover is bounded by,

$$P_d(n, H_k) \leqslant 1 - P_{k,\text{even}}(n, H_k)$$

Fig. 1 shows an n point crossover which is non-disruptive. d_1, d_2 and d_3 are fixed bits, L_1 and L_2 are distances of d_3 and d_2 from d_1 respectively and L is the string length. As a special case, probability of even number of crossover points falling between fixed bits for a second order hyperplane is given by [35]

$$P_{2,\text{even}}(n, L, L_1) = \sum_{i=0}^{\frac{n}{2}} {}^{n}C_{2i} \frac{L_1}{L}^{2i} \frac{L - L_1}{L}^{n-2i} \tag{5}$$

That is, the probability is given by the product of number of ways of choosing an even number of points from an n point crossover, the probability of placing an even number of points between the two defining points and the probability of placing the other points outside the defining points. L here is the string length and L_1 is the defining length.

We can extend the probability of disruption for a kth order hyperplane as

$$P_{k,\text{even}}(n, L, L_1, \ldots, L_{k-1}) = \sum_{i=0}^{\frac{n}{2}} {}^{n}C_{2i} \frac{L_1}{L}^{2i} \frac{L - L_1}{L}^{n-2i} P_{k-1,\text{even}}(n, L_1, L_2, \ldots, L_{k-1}) \tag{6}$$

That is, probability that an even number of crossover points fall between k defining bits $P_{k,\text{even}}$ is given by the probability that even number of crossover points fall between the first two defining bits and the rest of the points fall outside the defining bits into $P_{k-1,\text{even}}$. Hence, taking bound on the probability of disruption

$$P_d(H_k) \leqslant 1 - \sum_{i=0}^{\frac{n}{2}} {}^{n}C_{2i} \frac{L_1}{L}^{2i} \frac{L - L_1}{L}^{n-2i} P_{k-1,\text{even}}(n, L_1, \ldots, L_{k-1}) \tag{7}$$

Now, as mentioned earlier, the lower bound on the gain is given by the preservance of schema when disruptive crossover occurs between two parents both following the same schema. Hence, gain is given by $gains \geqslant n * P_{d^*}$ Probability that P_1 and P_2 are in schema H.

After selection, the number of individuals in schema H is given by $M(H,t)\frac{f(H)}{f_{\text{avg}}}$. Total number of individuals in a population is n. Hence probability that given a parent, it is in schema H is given by $\frac{M(H,t)}{n}\frac{f(H)}{f_{\text{avg}}}$. Hence the gain is got as,

$$gains \geqslant n_{t+1} P_d \frac{M(H,t)}{n_t} \frac{f(H)}{f_{\text{avg}}} \frac{M(H,t)}{n_t} \frac{f(H)}{f_{\text{avg}}} \tag{8}$$

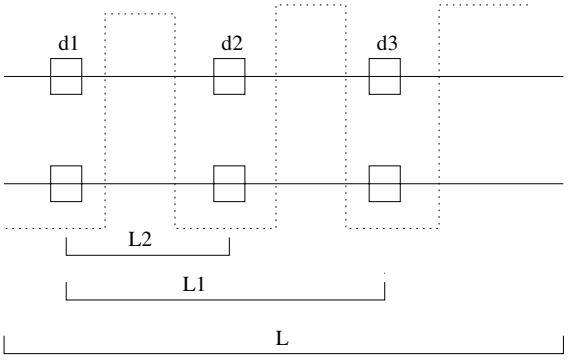

Fig. 1. Non-disruptive n point crossover.

Using this lower bound on *gains* in Eq. (4) and replacing loss with disruption

$$M(H,t+1) \geqslant \left[\frac{M(H,t)}{n_t} n_{t+1} \frac{f(H)}{f_{\text{avg}}}(1-P_d) + n_{t+1} P_d \left(\frac{M(H,t)}{n_t}\frac{f(H)}{f_{\text{avg}}}\right)^2\right](1-p_{\text{m}})^{\text{O}(H)}$$

Simplifying,

$$M(H,t+1) \geqslant \frac{M(H,t)}{n_t} n_{t+1} \frac{f(H)}{f_{\text{avg}}}\left[1 - P_d + P_d\left(\frac{M(H,t)}{n_t}\frac{f(H)}{f_{\text{avg}}}\right)\right](1-p_{\text{m}})^{\text{O}(H)}$$

But $n_t f_{\text{avg}} = \sum f$ for generation t. Therefore we get the schema theorem as

$$M(H,t+1) \geqslant \frac{M(H,t)}{n_t} n_{t+1} \frac{f(H)}{f_{\text{avg}}}\left[1 - P_d\left(1 - \frac{M(H,t) f(H)}{\sum f}\right)\right](1-p_{\text{m}})^{\text{O}(H)} \tag{9}$$

This is the schema theorem that deals with a single population. An ecosystem of population where populations with better fitness have more number of individuals than those with low fitness population is considered in our algorithm. Consider a low fitness population in the low yielding region of the search space. Consider the case when this population comes across a high fitness point in the search space. Let $\overline{H}$ be the schema with high fitness that the low fitness population came across. All other individuals in population have a low fitness compared to this high fitness point and hence

$$f(\overline{H}) = \sum f$$

Applying this to Eq. (9) we get

$$M(\overline{H},t+1) \geqslant \frac{M(\overline{H},t)}{n_t} n_{t+1} \frac{f(\overline{H})}{f_{\text{avg}}}[1 - P_d(1 - M(\overline{H},t))](1-p_{\text{m}})^{\text{O}(\overline{H})}$$

Since we have only found a single point with high fitness, $M(\overline{H},t) = 1$. Therefore

$$M(\overline{H},t+1) \geqslant \frac{M(\overline{H},t)}{n_t} n_{t+1} \frac{f(\overline{H})}{f_{\text{avg}}}(1-p_{\text{m}})^{\text{O}(\overline{H})}$$

Thus there is no disruption. $f_{\text{avg}} = \sum f / n_t$ and $f(\overline{H}) = \sum f$. Therefore

$$M(\overline{H},t+1) \geqslant \frac{M(\overline{H},t)}{n_t} n_t n_{t+1}(1-p_{\text{m}})^{\text{O}(\overline{H})}$$

But $M(\overline{H},t) = 1$. Therefore

$$M(\overline{H},t+1) \geqslant n_{t+1}(1-p_{\text{m}})^{\text{O}(\overline{H})}$$

Thus if a population P_i in the low fitness region comes across even one high fitness solution, in the very next generation approximately $n_{i,t+1}(1-p_{\text{m}})^{\text{O}(\overline{H})}$ of the $n_{i,t+1}$ individuals of the population in the next generation are in schema $\overline{H}$. This immediate drift towards high fitness region of a low fitness population, suggests the robust performance of the method. The assumption that the fitness of the best solution is approximately equal to sum of fitness of all individuals in the population is used to display the power of the algorithm when the entire population is in a very low fitness area of search space. This analysis is a proof for the performance of the algorithm in the worst case. In general, the drift towards high fitness region is faster when the population lies in a low fitness region. The high fitness populations of the ecosystem drive the low fitness populations to perform a more explorative search while the high fitness populations perform a concentrated exploitational search in the high yielding regions. As soon as the low fitness populations find a better region, the individuals in the population crowd this region of the search space. This competitive nature of populations ensures robust performance. The search conducted by the genetic algorithm can thus be explained as exploitational with respect to high yielding regions of the search space and explorative with respect to other regions. The adaptive updates to the parameters of number of points of crossover and rate of mutation for each population are responsible for the explorative nature of the low fitness population. The adaptive parameter of population size helps in concentrated search in high fitness region of search space. The term explorative indicates that larger

area of search space is covered by the search. There is no doubt that as mutation rate increases, the search is more explorative.

Lemma 1. *The total number of individuals in the ecosystem is a constant*; *that is*,

$$\sum_{i=1}^{np} n_{i,t} = \sum_{i=1}^{np} n_{i,t+1} = \text{constant}$$

Let the initial number of individuals in the ecosystem be n_{sum}. Therefore,

$$\sum_{i=1}^{np} n_{i,0} = n_{\text{sum}} = \text{constant}$$

Now at generation t, let total number of individuals in the ecosystem be n'_{sum}. Therefore,

$$\sum_{i=1}^{np} n_{i,t} = n'_{\text{sum}}$$

In the next generation $t+1$, apply the update given by Eq. (1) to all the populations, then the total number of individuals in the ecosystem in generation $t+1$ is given by,

$$\sum_{i=1}^{np} n_{i,t+1} = \sum_{i=1}^{np} \left(n_{i,t} + \frac{f(P_i)}{\bar{f}} - 1 \right)$$

$$\sum_{i=1}^{np} n_{i,t+1} = \sum_{i=1}^{np} n_{i,t} + \sum_{i=1}^{np} \frac{f(P_i)}{\bar{f}} - \sum_{i=1}^{np} 1$$

Therefore

$$\sum_{i=1}^{np} n_{i,t+1} = n'_{\text{sum}} + \frac{1}{\bar{f}} \sum_{i=1}^{np} f(P_i) - np$$

But

$$\bar{f} = \frac{\sum_{i=1}^{np} f(P_i)}{np}$$

Therefore

$$\sum_{i=1}^{np} n_{i,t+1} = n'_{\text{sum}} + np - np = n'_{\text{sum}} = \sum_{i=1}^{np} n_{i,t}$$

Substituting $t = 0$ we get

$$\sum_{i=1}^{np} n_{i,0} = \sum_{i=1}^{np} n_{i,t} = n'_{\text{sum}} = n_{\text{sum}} = \text{constant}$$

Thus, the number of individuals in the ecosystem is constant, i.e., the algorithm just groups these individuals into different populations such that, individuals in the same population reproduce and show co-operation while individuals in different populations compete to perform better. Thus the algorithm displays interpopulation competition and intra population co-operation.

5.1. Convergence analysis

The convergence rate of a genetic algorithm is defined in this paper as the rate at which the hamming distance between the individuals of ecosystem becomes zero. In other words, convergence occurs when all individuals of population have the same genotype. Therefore, the convergence rate of the genetic algorithm is the rate at which the number of copies of this individuals increases till all the individuals of the ecosystem have

same genotype (zero hamming distance). Hence we can derive a bound on the rate of convergence by fixing rate of increase of expectation value of individual shown in Eq. (9).

Before deriving rate of change of expectation value consider the update to number of individuals in a population. From Eq. (1) we have,

$$n_{i,t} = n_{i,t-1} + \frac{f(P_{i,t-1})}{\bar{f}_{t-1}} - 1$$

Expanding the recursive equation for t generations we get,

$$n_{i,t} = n_{i,0} + \sum_{j=0}^{t-1} \frac{f(P_{i,j})}{\bar{f}_j} - t \tag{10}$$

Now, the use of Elitist selection guarantees the convergence of the genetic algorithm. During the process of convergence under steady state, a population is either increasing or decreasing in population size. This suggests a linear relation of the summation term in Eq. (10) and t. Let the coefficient of linear term be β. Therefore,

$$n_{i,t} = n_{i,0} + \beta t - t$$

Therefore, the population size is linearly dependent on number of generations. Thus,

$$n_{i,t} = n_{i,0} + (\beta - 1)t \tag{11}$$

where β is some constant.

Applying this in Eq. (9) we get,

$$M^i(H,t) \geqslant M^i(H,t)(n_{i,0} + (\beta-1)t)\frac{f(H)}{\sum \bar{f} n_{i,t-1}}\left[1 - P_d\left(1 - \frac{M^i(H,t)f(H)}{\sum f_{i,t}}\right)\right](1-p_{\mathrm{m}})^{\mathrm{O}(H)} \tag{12}$$

where $M^i(H,t)$ is the expectation value of schema H in population P_i. Let

$$f(H) = \bar{f} + c\bar{f}$$

where c is some constant. That is the average fitness of schema H is above the average fitness of the population by a factor c. Therefore,

$$\frac{f(H)}{\bar{f}} = \text{constant}$$

Now,

$$n_{i,t-1} = n_{i,0} + (\beta - 1)(t-1)$$

Now to calculate a bound on convergence rate, we use the Θ notation. As $n_{i,t}$ and $n_{i,t-1}$ are linearly dependent on t, in the calculation of order of convergence they cancel each other out. Mutation just induces small randomness into the search and it does not alter the order of convergence but only the actual convergence rate.

From Eq. (12) the order of convergence $\Theta(M^i(H,t))$ as,

$$\Theta(M^i(H,t)) = \Theta(\alpha M^i(H,t-1)^2)$$

Expanding the recursion of $M^i(H,t-1)$ we get,

$$\Theta(M^i(H,t)) = \Theta(\alpha^3 M^i(H,t-2)^4)$$

Further expanding the recursive relation we get,

$$\Theta(M^i(H,t)) = \Theta(\alpha^7 M^i(H,t-3)^8)$$

Finally expanding the entire recursive relation we get,

$$\Theta(M^i(H,t)) = \Theta(\alpha^{1+2+4+\cdots+2^{t-1}})$$

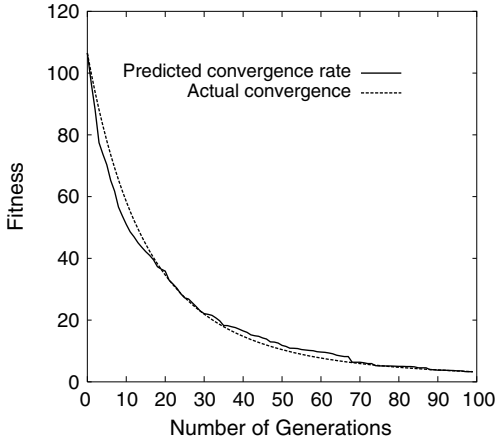

Fig. 2. Plot of actual versus predicted convergence.

As the term in power is the summation of geometric series with factor two, we get,

$$\Theta(M^i(H,t)) = \Theta(\alpha^{2^t-1})$$

Thus the rate of convergence of the algorithm is of the order,

$$\Theta(M^i(H,t)) = \Theta(\alpha^{2^t})$$

Clearly this is of exponential of exponential order. This shows that the algorithm has a much better convergence rate as compared to simple GA.

The graph in Fig. 2 shows a plot of convergence of the proposed Genetic Algorithm for the first De Jongs test bed function (F1) and the convergence rate derived. From the graph it is clear that the convergence rate of the algorithm is close to predicted convergence rate. The actual convergence rate was compared with the function,

$$C(x) = e^{e^x}$$

Such that,

$$1.54078 \geqslant x \leqslant 0.163734$$

This clearly proves that the actual convergence rate of the algorithm is indeed exponential of exponential as predicted from mathematical analysis.

6. Experiments

Each population of the ecosystem can be run in parallel on different processors as the algorithm is a parallel genetic algorithm. However, the algorithm was implemented on a single processor system. Fig. 3 shows a view of the ecosystem implemented. The ecosystem consists of a linked list of populations and variables to hold average fitness of population and average number of individuals in a population. The linked list structure helps in an easy implementation of the phenomenon of extinction where, when the population becomes extinct that node is simply removed.

Each population which is a node of the linked list is a structure consisting of the population parameters like, number of points of crossover, population size and rate of mutation. Further, the population also contains an array of bit strings which represent the individuals of the population. The genotype to phenotype conversion and fitness evaluation is performed by a fitness function.

To evaluate the performance of the algorithm, it was used to find the minimal points of complex high dimensional landscapes obtained using the testbed functions shown in Table 1. Function F1, which is the De Jong's function 1, is a unimodal function with one distinct peak. Function F2 (Rosenberg function), is

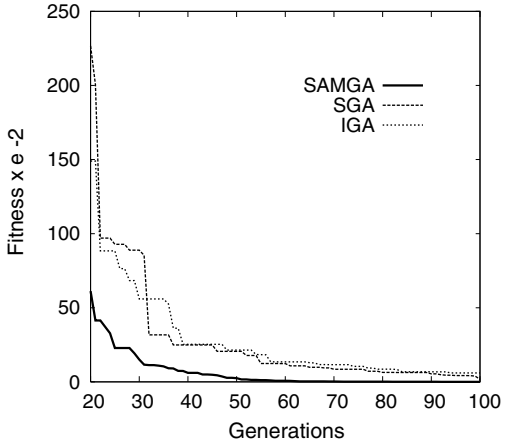

Fig. 3. Convergence for function F1.

Table 1
Testbed functions used

F1	$f(x) = \sum_1^{10} x_i^2 - 5.12 \leqslant x_i \leqslant 5.12$
F2	$f(x) = 100(x_1^2 - x_2^2) + (1 - x_1)^2 - 2.048 \leqslant x_i \leqslant 2.048$
F3	$f(x) = 200 + \sum_{i=1}^{10} x_i^2 - 10\cos(2\pi x_i) - 5.12 \leqslant x_i \leqslant 5.12$
F4	$f(x) = \sum_{i=1}^{10} x * \sin(x) - 5.12 \leqslant x_i \leqslant 5.12$

a function which is basically an almost flat valley and hence finding the minimal point becomes a difficult problem. Function F3 (Rastragin function) is a multimodal function with many peaks. The function is a good testbed function as any algorithm has a good chance of getting stuck at one of the peaks which is a local minima. Function F4 is again a multimodal function. In the experiments conducted, the IGA and SAMGA both had 10 populations of 60 individuals each. The SGA had 600 individuals in its population. For SGA and IGA the crossover rate chosen was 1 and mutation rate 0.01.

The plot in Fig. 3 shows the convergence of our algorithm(SAMGA), island model (IGA) and simple genetic algorithm (SGA) for the function F1. It is observed that the Self-Adaptive Migration GA (SAMGA) converges much faster than the other algorithms. The plot in Fig. 4 shows the convergence of the three algorithms for Rosenberg function. As our function is low dimensional, the fitness was multiplied by 1000 before plotting. Similarly, the plot in Fig. 5 shows the convergence for Rastragin function. This is a multimodal function with many peaks of almost equal heights. In both the cases, SAMGA outperforms IGA and SGA.

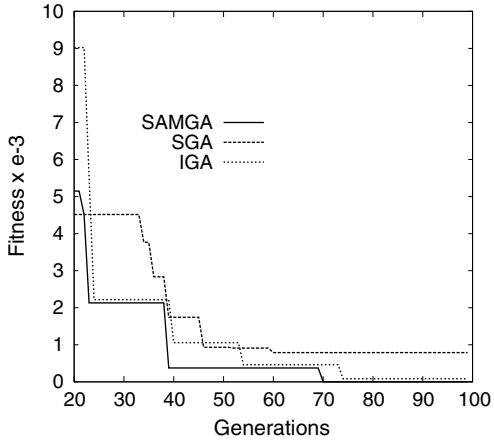

Fig. 4. Convergence for function F2.

The plot in Fig. 6 shows the convergence for function F4. Here SAMGA and SGA have similar performance and both outperform IGA. The graphs in Fig. 7–10, shows the convergence rate of all the three

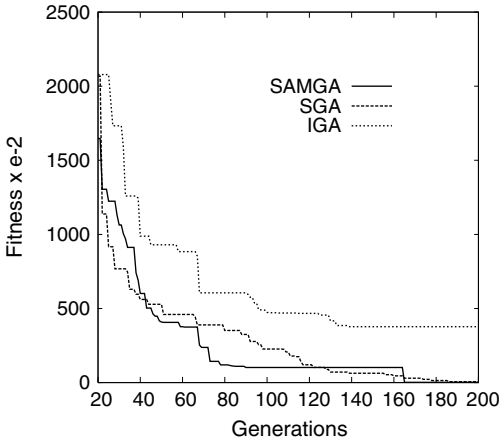

Fig. 5. Convergence for function F3.

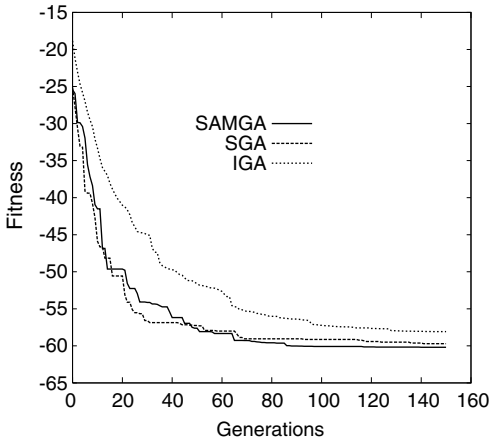

Fig. 6. Convergence for function F4.

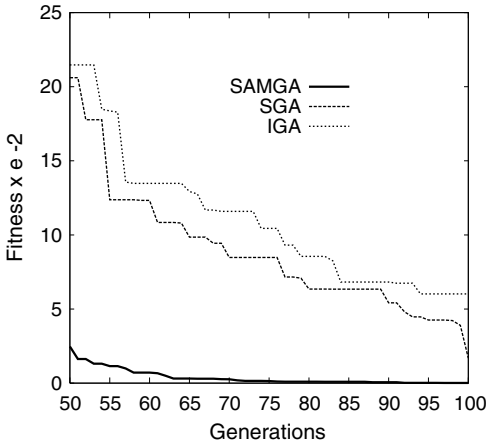

Fig. 7. Convergence for function F1.

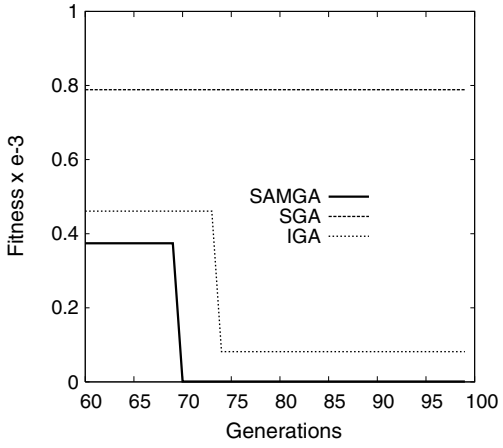

Fig. 8. Convergence for function F2.

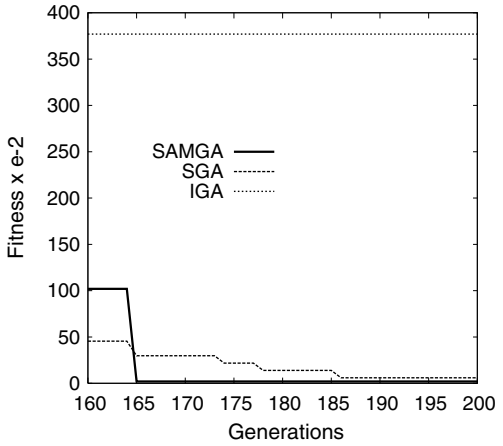

Fig. 9. Convergence for function F3.

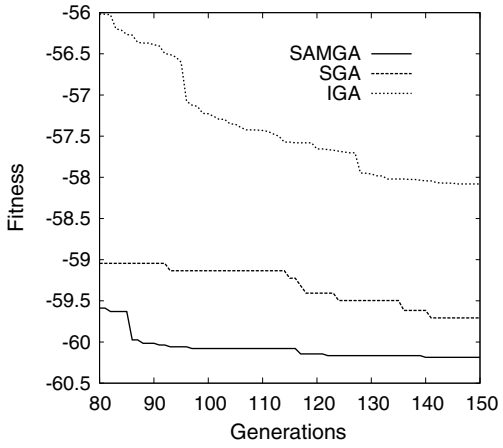

Fig. 10. Convergence for function F4.

algorithms for varying generations and functions. In all these cases, the performance of SAMGA is most significant in the later generations nearer to convergence.

To test how the explorative power of the proposed algorithm helps in overcoming GA-deception the algorithm along with simple GA and Island GA was used to solve ugly 3 bit deceptive function. With 250 individuals in four populations, SAMGA was able to overcome deception in 150 generations, whereas IGA overcame deception only after 228 generations. With 1000 individuals SGA was able to overcome deception in 557 generations. This experiment proves that the explorative nature on the lower fitness populations in SAMGA helps in overcoming GA deception better.

6.1. Example search

Consider the landscape given by the equation

$$y = (x - 1) * \cos(x - 1)$$

The proposed GA was used on this landscape with an initialization of three populations of four individuals each. Fig. 11 shows the function plot along with the initial set of individuals in the ecosystem. Fig. 12 shows the search result after 10 generations and Fig. 13 shows the search result after 20 generations. The number of

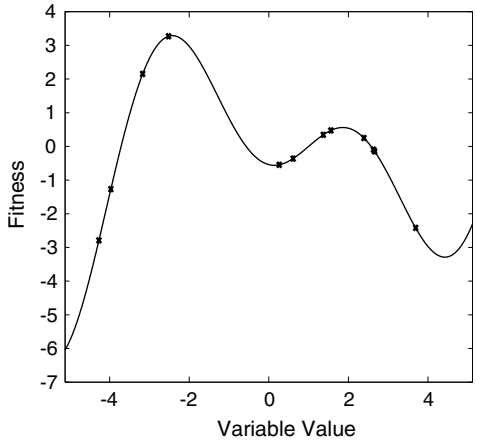

Fig. 11. Example: Initial set of individuals.

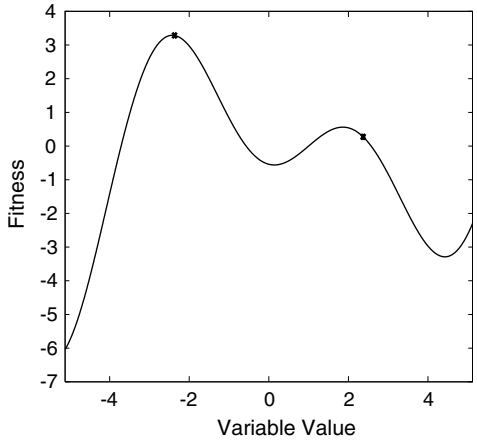

Fig. 12. Example: Individuals after 10 generations.

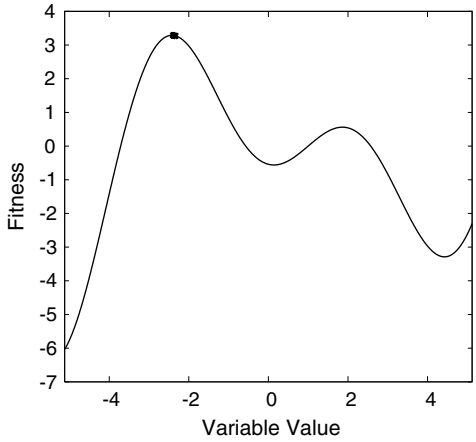

Fig. 13. Example: Individuals after 20 generations.

individuals present is not clear from the graph as in the 10th and 20th generations, many individuals have the same bit string. However, from Lemma 1, the total number of individuals is the same in ecosystem. From the result it can be seen that from the initial random distribution, in the 10th generation individuals get settled at the first and second peaks (one individual at the lower peak and eleven at higher peak). Finally, in the 20th generation, all the individuals get settled at the higher peak which is the maxima.

7. Performance analysis

To evaluate the performance of the algorithm on real data mining applications, a Michigan Style classifier was built (see Appendix for details on classifier system) and the classifier was tested on Datamining applications from the UCI Machine learning repository. The training set used was the first 40% of the data. The next 60% was used for testing. The various data mining applications chosen for the test were

- Pima Indian Diabetes (PID) Database with 768 cases, 2 classes and 8 attributes.
- Wisconsin Breast Cancer Database (Wisc) with 699 entries, 2 classes and 9 attributes.
- Hepatitis Database (Hep) with 155 entries, 2 classes and 19 attributes.
- Ionosphere Database (Ion) with 351 entries, 2 classes and 34 attributes.

The different methods that were compared are given below. For a survey of these methods on machine learning problems refer [26]:

- Self-Adaptive Migration GA (SAMGA) – The proposed algorithm.
- Learning Vector Quantization (LVQ) – Establishes number of codebooks to approximate various domains of input vector by quantized values.
- Backpropagation Neural Network (Bpnn) – A neural network that uses feedback of errors for learning.
- Radial Basis Function neural network (RBF) – It is a Neural network
- Incremental Decision Tree Induction (ITI) – Builds decision tree incrementally as new training instances are provided.
- Linear Machine Decision Tree (LMDT) – Uses multi-class decision trees with multivariate tests at internal decision nodes.

The table in Fig. 14 summarizes the performance of SAMGA and other machine learning techniques. It can be seen that SAMGA has consistently good performance and relatively high classification rates as compared to other methods. It can be seen that the performance of the methods vary from application to application but SAMGA gives consistent good performance for all the applications.

Methods / Data Set	SAMGA	ITI	LVQ	LMDT	Bpnn	RBF
Pima	74.6	73.16	71.28	73.51	73.4	71.6
Wisc	96.28	91.14	94.82	95.74	95.8	94.9
Ion	89.4	93.65	88.58	86.89	86.6	87.6
Hep	83.4	75.93	78.3	84.59	75.4	77

Fig. 14. Accuracy of the classification results.

Methods / Data Set	SAMGA	ITI	LVQ	LMDT	Bpnn	RBF
Pima	73	71.6	70	72.51	72.22	70.16
Wisc	94.1	90.40	93.24	93.44	93.80	92.49
Ion	86.2	91.45	85.12	85.90	84.77	84.76
Hep	84.7	72.32	74.31	83.45	74	75.70

Fig. 15. Accuracy of the classification results after 10-fold cross validation.

Similarly, the performance of SAMGA after 10-fold cross validation of the datasets used is given in Fig. 15. It can be seen that SAMGA shows better performance when compared with the other techniques even after the 10-fold cross validation of the datasets.

8. Conclusions

Genetic Algorithms have proved to be a robust general purpose search technique. They have a central place in data mining applications due to their ability to search large search spaces efficiently. In this paper we have proposed a Self-Adaptive Migration GA search techniques, have two central but competing concepts of exploitation and exploration. The proposed algorithm can be characterized by focused and deeper exploitation of the heuristically promising regions of the search space and wider exploration of other regions of the search space. The algorithm achieves this by varying the number of individuals in a population which helps in better exploitation of high fitness regions of the search space by increasing the number of individuals in the region. The algorithm also helps in better exploration by increasing the number of crossover points and the mutation rates for the low fitness population.

The paper provides a mathematical analysis of the method using schema theorem which accounts for lower bound on the gain. It has been shown that the method proposed assures that, when a low fitness population of the ecosystem stumbles across a high fitness solution due to its explorative nature there is an immediate drift of that population towards the high fitness region of search space.

The proposed method has been tested on a set of testbed functions and on real data mining problems of classification. The results show that the method achieves faster convergence and provides better accuracies in classification. The convergence of the method has been compared with those using single population genetic algorithm and Island model GA and results prove that the proposed algorithm outperforms both SGA and IGA. The classifier system built using SAMGA was evaluated against other classification systems.

Appendix

Genetic Algorithm classifiers can be divided into two classes, the Michigan Approach classifiers and the Pittsburg approach classifier based on how rules are encoded in the population of individuals. In the Michigan

Age		Weight		Class
18	21	50	70	0

Fig. 16. Rule representation.

Age		Weight		Class
00010010	00010101	00110010	01000110	0

Fig. 17. Bit string representation of the rule.

approach, each individual of the population is a single rule of the entire rule base. In the Pittsburg approach, each individual encodes a set of prediction rules. The classifier used in this paper is a Michigan approach classifier.

In the classifier system, each rule is represented by a bit string. Consider the rule

$$if\ (18 \leqslant age \leqslant 21)\ and\ (50 \leqslant weight \leqslant 70)$$
$$then\ class = normal$$

Now if there are only two classes, normal(0) and abnormal(1), then this rule can be represented in the format given by Fig. 16. Fig. 17 gives the bit string representation of the same rule. Generally for the fitness evaluation of the rule credit assignments are provided. For the Michigan style classifier, care should be taken to eliminate identical individuals from the population so that the rule base is not made up of the same highly fit rule.

References

[1] Ashish Ghosh, Bhabesh Nath, Multi-objective rule mining using genetic algorithms, Information Sciences 163 (1–3) (2000) 123–133.
[2] K.B. Athreya, H. Doss, J. Sethuraman, On the convergence of the Markov chain simulation method, Annals of Statistics 24 (1996) 69–100.
[3] T. Back, Self adaptation in genetic algorithms, in: F.J. Varela, P.B. (Eds.), Proceedings of First European Conference on Artificial Life, 1992, pp. 263–271.
[4] K.A. De Jong, W.M. Spears, D.F. Gordon, Using genetic algorithms for concept learning, Machine Learning 13 (1993) 161–188.
[5] K.A. De Jong, An analysis of the behaviour of a class of genetic adaptive systems. Ph.D. thesis, Department of Computer and Communication Sciences, University of Michigan, Ann Arbor, 1975.
[6] K. Deb, H.G. Beyer, Self adaptive genetic algorithms with simulated binary crossover, Evolutionary Computation 9 (2) (2001) 197–221.
[7] P. Deepa Shenoy, K.G. Srinivasa, K.R. Venugopal, Lalit M. Patnaik, Evolutionary approach for mining association rules on dynamic databases, in: Proc. of PAKDD, LNAI, 2637, Springer-Verlag, 2003, pp. 325–336.
[8] A.E. Eiben, E.H.L. Aarts, K.M. Van Hee, Global convergence of genetic algorithm: a Markov chain analysis, in: H.P. Schefel, R. Manner (Eds.), Parallel Problem Solving from Nature, Springer, Berlin, 1991, pp. 4–12.
[9] A.E. Eiben, I.G. Sprinkhuizen-Kuyper, B.A. Thijseen, Competing crossovers in an adaptive GA framework, in: Proceedings of the Fifth IEEE Conference on Evolutionary Computation, IEEE Press, 1998, pp. 787–792.
[10] F. Herrera, M. Lozano, Gradual distributed real-coded genetic algorithms, IEEE Transactions on Evolutionary Computation 4 (1) (2000) 43–62.
[11] Fernando G. Lobo, David E. Goldberg, The parameter-less genetic algorithm in practice, Information Sciences 167 (1–4) (2000) 217–232.
[12] J. He, L. Kang, Y. Chen, Convergence of genetic evolution algorithms for optimization, Parallel Algorithms and Applications 5 (1995) 37–56.
[13] F. Herrera, M. Lozano, Adaptive genetic algorithms based on fuzzy techniques, in: Proc. of Sixth Intl. Conf. on Information Processing and Management of Uncertainty in Knowledge Based Systems (IPMU' 96), Granada, July 1996, pp. 775–780.
[14] R. Hinterding, Z. Michalewicz, T.C. Peachey, Self adaptive genetic algorithm for numeric functions, in: Proceedings of the 4th Conference on Parallel Problem Solving from Nature, 1996, pp. 420–429.
[15] J.H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Arbor, 1975.
[16] J.H. Holland, Escaping brittleness: the possibilities of general-purpose learning algorithms applied to parallel rule-based systems, in: T. Mitchell et al. (Eds.), Machine Learning, vol. 2, Morgan Kaufmann, 1986, pp. 593–623.
[17] Jacinto Mata Vzquez, Jos Luis lvarez Macas, Jos Cristbal Riquelme Santos, Discovering Numeric association rules via evolutionary algorithm, PAKDD, 2002, pp. 40–51.

[18] Joanna Lis, Parallel genetic algorithm with the dynamic control parameter, in: International Conference on Evolutionary Computation, 1996, pp. 324–329.
[19] Kalin Penev, Guy Littlefair, Free search comparative analysis, Information Sciences 172 (1–2) (2005) 173–193.
[20] E. Kee, S. Aiery, W. Cye, An adaptive genetic algorithm, in: Proceedings of The Genetic and Evolutionary Computation Conference, 2001, pp. 391–397.
[21] T. Krink, R.K. Ursem, Parameter control using the agent based patchwork model, in: Proceedings of The Congress on Evolutionary Computation, 2000, pp. 77–83.
[22] J.H. Lobo, The parameter-less genetic algorithm: rational and automated parameter selection for simple genetic algorithm operation, Ph.D. thesis, University of Lisbon, Portugal, 2000.
[23] J.H. Martin, J. Lienig, J.H. Cohoon, Population structures: island (migration) models: evolutionary algorithms based on punctuated equilibria, in: T. Back, D.B. Fogel, Z. Michalewicz (Eds.), Handbook of Evolutionary Computation, Inst. Phys. Publ. Oxford University Press, New York, 1997, pp. C6.3:1–C6.3:16.
[24] A. Nix, M.D. Vose, Modeling genetic algorithms with Markov chains, Annals of Mathematics and Artificial Intelligence 5 (1992) 79–88.
[25] Oscar Montiel, Oscar Castillo, Roberto Seplveda, Patricia Melin, Application of a breeder genetic algorithmnext term for finite impulse filter optimization, Information Sciences 161 (3–4) (2004) 139–158.
[26] Peter W. Eklund, A performance survey of public domain supervised machine learning algorithms, Technical Report, The University of Queensland Australia, 2002.
[27] G. Rudolph, Convergence analysis of canonical genetic algorithms, IEEE Transactions on Neural Networks 5 (1995) 96–101.
[28] D. Schierkamp Voosen, H. Muhlenbein, Strategy adaptation by competing subpopulations, Parallel Problem Solving from Nature III, Springer-Verlag, Berlin, Germany, 1994, pp. 199–208.
[29] V. Schnecke, O. Vornberger, An adaptive parallel genetic algorithm for VLSI layout optimization, in: Proc. of fourth Intl. Conf. on Parallel Problem Solving from Nature, 1996, pp. 859–868.
[30] M. Srinivas, L.M. Patnaik, Binomially distributed populations for modelling GAs, in: Proceedings of Fifth International Conference in Genetic Algorithms, Morgan Kauffmann Publishers, 1993, pp. 138–143.
[31] M. Srinivas, L.M. Patnaik, Adaptive probabilities of crossover and mutation in genetic algorithms, IEEE Transactions on Systems, Man and Cybernetics 24 (4) (1994) 17–26.
[32] Sushil J. Louis, Gregory J.E. Rawlins, Syntactic analysis of convergence in genetic algorithms, Foundations of Genetic Algorithms (2002) 141–152.
[33] S. Tongchim, P. Chongstitvatan, Parallel genetic algorithm with parameter adaptation, Information Processing Letters 82 (1) (2002) 47–54.
[34] M.D. Vose, G.E. Liepins, Punctuated equilibria in genetic search, Complex Systems 5 (1) (1991) 31–44.
[35] William M. Spear, Kenneth De Jong, An analysis of multipoint crossover, Foundations of Genetic Algorithms Workshop, Bloomington, 1991, pp. 301–315.