Soft Comput
DOI 10.1007/s00500-014-1339-0

METHODOLOGIES AND APPLICATION

Instance selection by genetic-based biological algorithm

Zong-Yao Chen · Chih-Fong Tsai · William Eberle · Wei-Chao Lin · Shih-Wen Ke

Abstract Instance selection is an important research problem of data pre-processing in the data mining field. The aim of instance selection is to reduce the data size by filtering out noisy data, which may degrade the mining performance, from a given dataset. Genetic algorithms have presented an effective instance selection approach to improve the performance of data mining algorithms. However, current approaches only pursue the simplest evolutionary process based on the most reasonable and simplest rules. In this paper, we introduce a novel instance selection algorithm, namely a genetic-based biological algorithm (GBA). GBA fits a "biological evolution" into the evolutionary process, where the most streamlined process also complies with the reasonable rules. In other words, after long-term evolution, organisms find the most efficient way to allocate resources and evolve. Consequently, we can closely simulate the natural evolution of an algorithm, such that the algorithm will be both efficient and effective. Our experiments are based on comparing GBA with five state-of-the-art algorithms over 50 different domain datasets from the UCI Machine Learning Repository. The experimental results demonstrate that GBA outperforms these baselines, providing the lowest classification error rate and the least storage requirement. Moreover, GBA is very computational efficient, which only requires slightly larger computational cost than GA.

Keywords Instance selection · Data reduction · Data mining · Machine learning · Genetic algorithms · Biological evolution

1 Introduction

In data mining or knowledge discovery in databases (KDD), data pre-processing has became one of the most important steps for good quality data mining. That is, if the chosen dataset contains too many instances (i.e., data samples), it can result in large memory requirements, slow execution speed, and over-sensitivity to noise. In addition, one problem with using the original data points is that there may not be any data points located at the precise points that would make for the most accurate and concise concept description (Pyle 1999).

Since the size of today's data collections often exceeds the size of data which current data mining algorithms handle properly, instance selection or data reduction should be considered before performing data mining tasks such as data classification and clustering. Instance selection is an important data pre-processing step in the data mining or KDD process. The aim of instance selection is to reduce the data size by filtering out noisy data from a given dataset, which are likely to degrade the mining performance (Wilson and Martinez 2000; Li and Jacob 2008). In particular, instance selection is used to shrink the data, and then data mining algorithms can be applied to the reduced dataset, which still

Communicated by V. Loia.

Z.-Y. Chen · C.-F. Tsai (✉)
Department of Information Management,
National Central University, Jhongli, Taiwan
e-mail: cftsai@mgt.ncu.edu.tw

W. Eberle
Department of Computer Science, Tennessee Technological University, Cookeville, USA

W.-C. Lin
Department of Computer Science and Information Engineering,
Hwa Hsia Institute of Technology, New Taipei, Taiwan

S.-W. Ke
Department of Information and Computer Engineering,
Chung Yuan Christian University, Jhongli, Taiwan

Published online: 21 June 2014

achieve sufficient results if the selection strategy is appropriate (Reinartz 2002).

Similarly, outlier detection is the task to discover observations that lie an abnormal distance from other values in a population. That is, outliers are the unusual observations (or bad data points) that are far removed from the mass of data (Aggarwal and Yu 2001; Barnett and Lewis 1994). Consequently, filtering out the detected outliers are very useful for obtaining good mining results (Knorr et al. 2000). Therefore, in the data mining perspective, the aim of instance selection is similar to the one of outlier detection (Liu and Motoda 2001).

In this paper, a novel instance selection method, namely genetic-based biological algorithm (GBA), is proposed. GBA simulates the biological evolution process and rules from nature where, after long-term evolution, organisms find the most efficient way to allocate resources and evolve (Ball 2002). Inspired by nature, GBA is constructed as efficient and effective problem solving for instance selection.

The major contribution of this paper is to introduce a novel evolutionary-based instance selection algorithm, i.e., GBA. It is the first approach to extend the genetic algorithm based on biological evolution. Moreover, GBA is assessed over a large-scale experiment based on 50 different domain problems and is compared with five well-known and representative instance selection algorithms. Experimental results demonstrate that GBA outperforms the other methods, which can provide the lowest classification error, the least storage requirement, and moderate time complexity.

The rest of this paper is organized as follows: Sect. 2 provides a brief review of related literatures, including the concept of instance selection and several well-known instance selection algorithms, such as ENN, IB3, DROP3, ICF, GA, and CCIS. The proposed GBA method for instance selection is introduced in Sect. 3. In Sect. 4, the experimental results based on 50 UCI datasets containing various domain problems are present. Finally, conclusions are given in Sect. 5.

2 Instance selection

Instance selection can be defined as follows: Given a dataset D composed of training set T and testing set U, let X_i be the i-th instance in D, where $X_i = (X_1, X_2, \ldots, X_m)$ which contains m different features. Let $S \subset T$ be the subset of selected instances that resulted from the execution of an instance selection algorithm. Then, U is used to test a classification technique trained by S (Cano et al. 2003; Derrac et al. 2010).

In the literature, there are a number of related studies proposing instance selection methods for obtaining better mining quality. Specifically, Pradhan and Wu (1999) and Jankowski and Grochowski (2004) surveyed several relevant selection techniques, which can be divided into three application-type groups: noise filters, condensation algorithms, and prototype searching algorithms. In addition, extensive comparative experiments were conducted by Wilson and Martinez (2000) and Brighton and Mellish (2002). They indicate that iterative case filtering (ICF) and decremental reduction optimization procedure 3 (DROP3) are the cutting-edge instance selection algorithms, which make the k-NN classifier provide better performances over other instance selection methods. Four well-known instance selection algorithms are overviewed as follows: ENN, IB3, ICF, and DROP3. Moreover, a description of genetic based algorithms is also included.

2.1 ENN

Edited nearest neighbor (ENN) (Wilson 1972) is a representative noise-filtering algorithm, in which S starts out the same as T, and then each instance in S is removed if it does not agree with the majority of its k nearest neighbor (with $k = 3$, typically). This edits out noisy instances, as well as close border cases, leaving smoother decision boundaries.

Repeated edited nearest neighbor (RENN) (Wilson 1972), an extension of ENN, applies the ENN algorithm repeatedly until all instances remaining have a majority of their neighbors with the same class, which continues to widen the gap between classes and smoothes the decision boundary as long as any changes are made in the selected set.

2.2 IB3

IB3 was introduced by Aha et al. (1991), which is based on an acceptable instance concept to carry out the selection. In Aha et al. (1991), IB1, IB2, and IB3 were compared with the C4 decision tree algorithm (Quinlan 1986) over six datasets. In particular, the sizes of the chosen datasets range from 303 to 800 data samples and the numbers of attributes of each data sample range from 7 to 21. The results show that IB3 can significantly reduce IB1's storage requirements and does not display IB2's sensitivity to noise. On average, IB3 allows k-NN to provide a higher range of classification accuracy than C4.5. However, they found that IB3's learning performance is highly sensitive to the number of irrelevant attributes used to describe instances.

2.3 DROP3

The decremental reduction optimization procedure 3 (DROP3) was proposed by Wilson and Martinez (2000). In Wilson and Martinez (2000), a number of different algorithms were compared, including ENN, IB3, and DROP series. In addition, 30 datasets collected from UCI Machine

Learning Repository[1] were used for their study. In particular, the sizes of the chosen datasets range from 101 to 8124 data samples and the numbers of attributes of each data sample range from 4 to 75, which are slightly larger than Aha et al. (1991). Their results show that DROP3 had higher average accuracy than IB3 and had the best mix of storage reduction and generalization accuracy.

2.4 ICF

Iterative case filtering (ICF), proposed by Brighton and Mellish (2002), uses the ENN algorithm to remove the noise from the training set. Then, in the second step, ICF removes each instance x for which the reachability(x) is bigger than the coverage(x). This procedure is repeated for each instance in T. After that, ICF recalculates the reachability and coverage properties and restarts the second step (as long as any progress is observed).

In Brighton and Mellish (2002), ICF was examined and compared with various instance selection algorithms, including ENN and IB3, in terms of classification accuracy. The same 30 datasets used in Wilson and Martinez (2000) were chosen in their study. They found that ICF can achieve the highest degree of instance set reduction as well as the retention of classification accuracy. More specifically, an average of 80 % of cases were removed and classification accuracy did not drop significantly.

2.5 Genetic algorithms

Evolutionary or genetic algorithms (GA) have become an effective instance selection approach to improve the performance of data mining algorithms (Derrac et al. 2010). In GA, a population of strings (called chromosomes), which encode candidate solutions (called individuals) to an optimization problem, evolves for better solutions. In general, the genetic information (i.e., chromosome) is represented by a bit string (such as binary strings of 0s and 1s) and sets of bits encode the solution. Then, genetic operators are applied to the individuals of the population for the next generation (i.e., a new population of individuals). There are two main genetic operators, which are crossover and mutation. Crossover creates two offspring strings from two parent strings copying selected bits from each parent. On the other hand, mutation randomly changes the value of a single bit (with small probability) to the bit strings. Furthermore, a fitness function is used to measure the quality of an individual to increase the probability that the single bit can survive throughout the evolutionary process.

Instance selection is a type of NP-hard problems (Guyonm 2003). However, related works have demonstrated that heuristic algorithms can provide reasonably well results over such problem (Cano et al. 2003; García-Pedrajas et al. 2010; Jing 2013). In addition, the fitness function makes the heuristic algorithms more flexibility because the fitness function can be specialized for different requirements. For instance, the chosen reduction rate of heuristic algorithms is dependent on the fitness function used. On the other hand, since a classifier-based evaluator is used in the heuristic algorithms, they can provide higher accuracy than other algorithms based on non-supervised learning-based evaluator.

In Cano et al. (2003), GA has shown that it can obtain better results than many traditional and non-evolutionary instance selection methods in terms of better instance selection rates and higher classification accuracy. On the other hand, Li and Jacob (2008) propose an adaptive sampling procedure using GA and its performance is assessed over three real-word and very large datasets. The results show that their proposed approach outperforms existing methods in classification, association rules, and summary statistics.

García-Pedrajas et al. (2010) introduce a cooperative evolutionary approach for instance selection, namely cooperative co-evolutionary instance selection (CCIS). It is based on two separate populations that evolve cooperatively in a divide and conquer manner. In particular, the training set is divided into several subsets that are searched independently. A population of global solutions relates the search in different subsets and keeps track of the best combinations obtained.

This approach is compared with three well-known algorithms, which are IB3, DROP3, and ICF, and a genetic algorithm over 50 datasets from the UCI Machine Learning Repository. The results show that CCIS is very effective both in terms of improving performance and in reducing computational cost.

Recently, Jing (2013) proposed a hybrid genetic algorithm for the feature selection problem. The results indicate that the heuristic algorithms are suitable for solving the data reduction problem including instance and feature selection. However, although instance selection can be approached by the heuristic algorithms, the hybrid genetic algorithm (2013) is not considered in this paper. This is because the feature and instance selection problems are somewhat different due to the fitness function, specific limitations, and conditions.

3 Genetic-based biological algorithm

3.1 The basic concept

In the previous section, the concept of genetic algorithms, or GAs, originally proposed by Holland (1992), was introduced. While GAs have demonstrated success for a diverse set of problem, genetic algorithms are only able to handle simple concepts. Basically, the idea is that if resources are limited,

[1] http://archive.ics.uci.edu/ml/.

the "organisms" will follow the most reasonable and simplest rules—allowing for a more effective use of resources, or the "reproduction of species" (Odum 1994; Ball 2002). Thus, by using simple rules, organisms that maximize the "savings cost" will be more efficient. While reasonable rules help with this approach, if we can fit a "biological evolution" into the evolutionary process, where the most streamlined process also complies with the reasonable rules, we will closely simulate the natural evolution of an algorithm, and the algorithm will be both efficient and effective.

In other words, the algorithm only pursues the simplest evolutionary process. While in general this is reasonable, as it is able to solve problems fairly efficiently, it does discard a number of other elements to pursue the efficiency in the evolutionary process. This could result in a performance degradation and could also cause it to fall into the local optimal solution. So, to counteract this effect, we will factor in four other key factors that can be found in evolution: "reasonable convergence", "inter-generational mating", "nonlinear adaptability" and "mass migration". These are detailed in the following sections.

3.2 Specific features of GBA

3.2.1 Reasonable convergence

In the process of evolution, creatures are not only mating within their population, but they are also mating with other populations. This is a hybrid concept. Inevitably, the two parent populations have lived in different environments, and the hybrid population will live in the environment of one or both of the parent populations. Thus, the hybrid population will not have the collection of alleles that are the most advantageous for either of those environments; a substantial loss of fitness, i.e., their likelihood of successfully reproducing is lessened. Therefore, hybrids of different populations have better performance than the purebred of their parents in the growth rate, fecundity, and adaptability. In order to maintain the racial differences to produce hybrid populations, organisms must have some differences. Existing studies have pointed out that reducing the gene pools of various wild and indigenous breeds result in the loss of genetic diversity (Pollan 2001). Since the indigenous breeds are often better adapted to local extremes in climate and have immunity to local pathogens, this represents a significant genetic erosion of the gene pool for future breeding (Pollan 2001; Ellstrand 2003). Therefore, to achieve continuous evolution of the organism, we must maintain the diversity of the gene pool and use natural evolution to find out the most viable species, rather than the local best species.

Due to the limitations of the algorithm (small population numbers), we were unable to provide exotic species to do interbreeding. While increasing the mutation rate will allow us to achieve this concept, in a traditional GA, if the mutation rate is too high, it often leads to a low efficiency for solving the process (similar to random search), and if mutation rate is too low (loss of genetic diversity), it will cause the algorithm to fall into the local optimal solution, and thus cannot effectively escape local optimal. It is difficult to effectively escape local optimal solution. In a biosphere, for example, it is unreasonable to take a long time to escape the local optimum. Therefore, our plan is to have algorithms with a high mutation rate that can also converge effectively. While in a traditional GA this would probably be unreasonable, we will use both "inter-generational mating" and "mass migration" to achieve this goal.

3.2.2 King of the genetics: inter-generational mating

We observed that certain organisms usually have a group of animals, the king of the spouse, in this group; the spouse's age is not the same as the king. As a result of the evolutionary mating process, the distribution of power does not solely rest with the younger individuals and should depend on the strength of the individuals. For example, in the lion species, even if the lion is older, if he is the strongest, he will be the king. However, age does affect the ability of organisms. In other words, only a few of the very powerful kings are able to do inter-generational mating. For example, old lions that are strong will have a longer mating capability. Most existing genetic algorithms use this concept of new-generational mating, but our GBA retains a small part of the previous generation plus the most powerful individuals from the new generation, allowing these kings to continue to compete with the new generation.

3.2.3 Nonlinear adaptability

The phenomenon of protecting vulnerable groups does not appear in the natural world. The "law of the jungle" and the "survival of the fittest" have always been the most authentic expression of nature. For instance, if a newborn deer cannot stand up in a short time, it will face elimination, as it will easily be eaten by wolves, tigers, or other animals. In this case, it will accelerate the elimination of the organism which has a lower adaptability, leaving surviving organisms with more resources (e.g., more food, fewer competitors, etc.) to breed their next generations. Consequently, we can see that the natural world does not waste time on these organisms which have a lower adaptability, because it not only cannot help the overall evolution (the next generations may not be better), but also reduces the resources of other organisms, which may even lead to an overall evolution with low efficiency. Under the conditions of limited resources, this situation is inevitable.

When a problem has a very large number of combinations as feasible solutions, the fitness value of all combinations

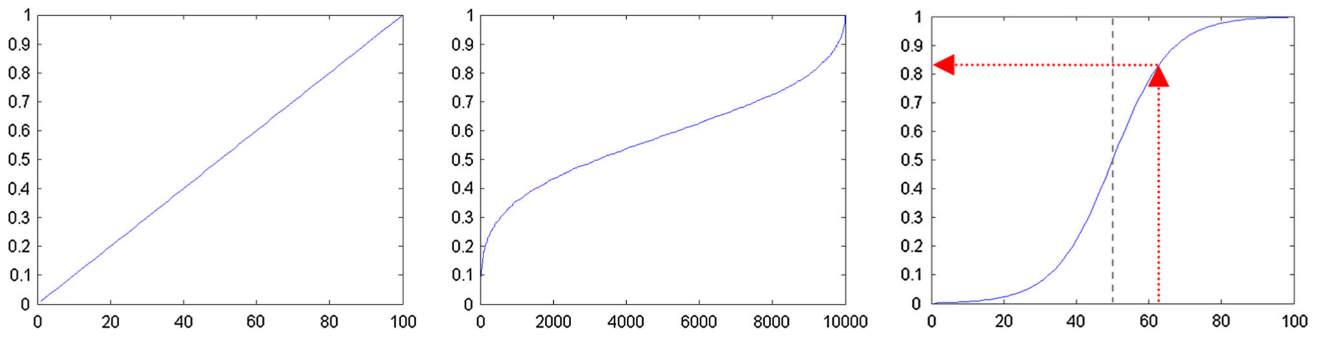


Fig. 1 Nonlinear and linear adaptability

will be close to linear, as shown in Fig. 1a. In contrast, Fig. 1b shows the worst case which, unfortunately, is the fitness curves of many practical problems. This is inconsistent with the natural law, because when the adaptability (fitness value) of organisms is below a certain threshold (the dashed line in the middle of Fig. 1c), these organisms will have little access to subsistence. In the natural world, the curve of the fitness should be as shown in Fig. 1c. Particularly, each curve can be regarded as a membership function of the fitness, in which the input of the membership function is the original fitness value, whereas the output is the weighted fitness value. In Fig. 1, the *x* axis represents the original fitness value (between 0 $\simeq$ 100), and the *y* axis is the weighted fitness value (between 0 $\simeq$ 1). For the example of Fig. 1c, if the original fitness is 62, then the weighted fitness will be 0.832.

While organisms with low resilience may also provide a few good genes (if the individual has a lot of good genes, it will have a higher adaptability), when the resources available to the organisms have a higher adaptability to do mutation or fine-tuning, it will have a greater benefit than the lower ones. This is where existing algorithms should be improved. We use the following steps to adjust the fitness f_i to fit the natural law of evolution and thus improve the efficiency of our algorithm. The process of the nonlinear adaptability can be defined as follows:

f is the fitness matrix,
N is the population size,
Or_i is the organism in the population.

Step 1 Use the fitness function (1) to calculate the fitness value f_i

$$f_i = \text{Fitness function } (Or_i), \quad i = 1, 2, \ldots N \tag{1}$$

Step 2 Use Eq. (2) to normalize the fitness f_i to fall in [0, 1].

$$No_i = \frac{F_i - Min(F) \times (1-0)}{Max(F) - Min(F)}, \quad i = 1, 2, \ldots, p \tag{2}$$

Step 3 Put the normalized fitness No_i into Eq. (3) to obtain the corresponding nonlinear adaptability Non_i.

$$Non_i = \frac{Hyperbolic\_tangent(\frac{No_i - 0.5}{\sigma^2}) + 1}{2}, \quad i = 1, 2, \ldots, N \tag{3}$$

We use the hyperbolic tangent function as the mapping model for the nonlinear adaptability. In our approach, the parameter σ is set as 0.4 (reasonable distribution and the best results), resulting in the mapping model shown in Fig. 1c.

Unlike the non-conversion approach, after conversion, organisms with low adaptability will have a lower mating rate, and organisms with a high adaptability will have a higher mating rate.

3.2.4 Great migration

Prior research has shown that, when a population has the following conditions: (1) large population size, (2) random mating, (3) no natural selection or mutation, (4) no great migration and (5) alleles (genome length) are the same, it will make the allele frequency remain a constant and the genotypic frequency is maintained at a certain level (Stern 1962; Emigh 1980). This is known in population genetics as the "Hardy–Weinberg law" or "Hardy–Weinberg equilibrium", and can be stated as follows:

$$(p+q)^2 = 1 \quad \text{or} \quad p^2 + 2pq + q^2 = 1, \tag{4}$$

where p is the dominant gene and q is the recessive gene. When one considers the status of the three genes, then the formula becomes $(p+q+r)^2 = 1$.

Accordingly, even the properties "most rare form" or "may disappear" of the gene can continue to survive. This phenomenon appears to ensure that the rate of evolution of genes is equal, but because the genes have not changed, this means that the evolutionary rate is 0 (non evolution) (Stern 1962). But, in the natural world, the evolution rate is not 0. Because of resource constraints (population size, food, territory size, etc.), it will retain the organisms which have a higher adaptability. As a result, organisms cannot randomly perform mating, but they can mutate and migrate, also known as the great migration.

However, while the traditional GA algorithm has most of its processes in line with natural laws, the great migration is not taken into account. Consequently, we will implement a great migration to improve the effectiveness and efficiency of GA algorithms.

Prior research has demonstrated that some great migration can produce more species, while several species have a much better adaptability than without migration, and some of them are even extinct (Flynn and Wyss 1998; Koepfli et al. 2007; Morgan 2002). This demonstrates that the great migration is necessary, as it helps to improve biological diversity (give a stable gene pool the opportunity to do hybridization between two populations) and survival rates (hybrid offspring will be better). Accordingly, we will modify the processes and conditions of great migration to improve algorithm performance:

1. Great migration is not frequent (when the whole gene pool tends to be stable).
2. Great migration will divide the population into two parts:
   - Foreign population: the higher adaptability can make the organism overcome the great migration. (Retain "K" organism as the foreign population, and do not have any change.)
   - Local population: through a high mutation rate to change "M" individuals where these individuals will be treated as other ethnic groups (achieve the gene diversity).
3. Use the foreign and local population to continue the evolution process.

3.2.5 GBA algorithm

The pseudo code of GBA is described below (Fig. 2), using the following definitions:

Pop is the population
N is the population size
Or_i is the organism
f_i is the fitness value of each organism
Non_i is the non-linear fitness value
King is the kings of a generation
K is the size of *KING* ($K < N$)
MP is the mating pool
$MPO_m \in MP, m = 1, 2, \ldots, M$
M is the size of *MP* ($M = N - K$)
$f_i \in F, i = 1, 2, \ldots, N$
$Or_i \in POP, i = 1, 2, \ldots, N$
$KO_k \in King, k = 1, 2, \ldots, K$
GF is the best fitness of each generation
G is the threshold of great migration
NG is the new generation.

3.2.6 The differences between GA and GBA

Table 1 summarizes the primary differences between the GBA and a traditional GA.

In Table 1, the fitness value of the GBA is mapped to a nonlinear model which enables us to achieve nonlinearly adaptability. It is different from the traditional linear adaptability that can accelerate the evolution process and remove the poor individuals, as it provides more mating opportunities for the high-end individuals. Since it incorporates the concept of intergenerational mating, GBA just needs to calculate the fitness for each newborn individual, rather than the entire population (because there are always a small number of individuals from the previous generation). In the new generation process of GBA, it joins the n-kings and the newborn individuals to form the next generation. Therefore, in the mating process of GBA, n-kings have the opportunity to mate with the new generation; even when the high mutation rate means that there is no individual similar to the kings, their genes have the opportunity to be passed down. Consequently, the n-kings will have more opportunities for fine-tuning and evolution.

4 Experiments

4.1 Experimental setup

In this paper, 50 different domain datasets from the UCI Machine Learning Repository were selected for the experiments. In addition, five state-of-the-art algorithms including IB3, ICF, DROP3, GA (Cano et al. 2003), and CCIS with ten subpopulations (García-Pedrajas et al. 2010) are compared with our proposed GBA approach.

Moreover, tenfold cross validation was used to divide each dataset into ten non-duplicated and approximately equal subsets. Nine subsets are used during classifier training and the remaining subset is used for classifier testing. As a result, the classifier is learned ten times and the classification performance is the average error over the ten subsets. More specifically, the k-NN ($k = 1$) classifier was considered since 1-NN can provide reasonable classification performances in most applications (Jain et al. 2000) and all related studies used it as the baseline classifier.

Finally, to assess the performance of these instance selection methods, the classification error, storage reduction, and the time spent by performing instance selection are examined.[2]

[2] The experimental environments are as follows: CPU: Intel(R) Core(TM) i7-3770 @ 3.40 GHz, RAN: 32 GB, OS: Windows 7–64bit, Code: Matlab R2012a.

Fig. 2 The pseudo code of GBA

```
1. Initialize population
   Randomly generate $N$-organisms $Or_i$ to compose the population $POP$
   Create a empty set of king $King$
   For $i$=1,2,…$N$
       $POP \leftarrow$ Randomly generate $Or_i$
   End
   Ø= $King$
2. Evaluate each organism of a new generation
   Calculate the fitness $f_i$ for each organism, and then convert it to the non-linear fitness $Non_i$
   For $i$=1,2,…$N$
       $f_i \leftarrow$ Fitness function ($Or_i$)
       $No_i = \frac{F_i - Min(F) \times (1-0)}{Max(F) - Min(F)}, i = 1,2,...p$
       $Non_i = \frac{Hyperbolic\_\tan gent(\frac{NO_i - 0.5}{\sigma^2}) + 1}{2}, i = 1,2,...N$
   End
3. Inter-generational mating
   Randomly select $K$ Kings from $POP$ based on nonlinear fitness $Non_i$ to compose the kings $King$
   For $k$=1,2,…$K$
       $King \leftarrow$ Select the king $KO_k$ from $POP$ based on nonlinear fitness $Non_i$
   End
4. Selection
   Randomly select $M$-organism $Or_m$ from population $POP$ in to the mating pool $MP$ based on nonlinear fitness $Non_i$
   For $m$=1,2,…$M$
       $MP \leftarrow$ Select the $Or_m$ from $POP$ based on nonlinear fitness $Non_i$
   End
5. Mating
   Crossover for each pair of the mating pool $MP$ to generate the New Generation $NG$
   For $m$=1,2,…($M/2$)
       $NG \leftarrow$ Crossover the $MPO_m$ and $MPO_{(m+M/2)}$
   End
6. Mutation
   Randomly mutation for each organism $MPO_m$ of the mating pool $MP$
   For $m$=1,2,…$M$
       $MPO_m$ = Random Mutation the $MPO_m$
   End
7. Great migration
   If the best fitness value is stable then apply the great migration
   If there is no change in the best fitness $GF$ in $G$ continuous generations then
       Do a strong mutation for the new generation $NG$ (Local population)
   End
8. New generation
   If great migration is not applied in this generation then
       Combine the Kings $King$ and the new generation $NG$ to replace the existing population
       $Pop \leftarrow King \cup NG$
   else
       Combine the foreign population ($King$) and the local population (strong mutation $NG$)
   end
9. If Stop conditions are met then stop else return to step 2.
```

Table 1 The differences between GA and GBA

Process	Traditional GA	GBA
Evaluation	Reckoning of a fitness value for each gene in this generation	Reckoning of a nonlinear fitness value for each gene in new generation
Select kings	–	Find out the kings of this generation
Mating	Select, crossover and mutation (just only new generation)	(Inter-generational mating) select, crossover and mutation (new generation and kings of the last generation)
Mass migration	–	Mass migration
New generation	Population: a whole new generation	Combine the new generation and the kings of the last generation

4.2 Classification error

Table 2 shows the classification error of 1-NN with instance selection by IB3, DROP3, ICF, GA, CCIS, and GBA and 1-NN without instance selection, respectively. Note that the underlined error rates mean the best instance selection results and '*' followed by the underlined error rates represent that 1-NN with instance selection performs better than the baseline 1-NN without instance selection.

As shown, on average the proposed GBA approach performs best, which allows 1-NN to provide lower error rates than 1-NN with the state-of-the-art algorithms individually. Particularly, 1-NN by GBA outperforms 1-NN by the other instance selection methods over 27 datasets. In addition, 1-NN with GBA performs better than the baseline over four datasets.

Moreover, GBA outperforms the other methods when the number of data samples is from medium to large. These datasets are 'Abalone' (4,177), 'Hypothyroid' (3,772), 'Kr vs. kp' (3,196), 'Optdigits' (5,620), 'Page_blocks' (5,473), 'Pendigits' (10,993), 'Phoneme' (5,404), 'Satimage' (6,435), 'Segment' (2,310), 'Sick' (3,772), 'Texture' (5,500), and 'Waveform' (5,000).

However, for those datasets composed of high-dimensional data (i.e., over 50 dimensions) including 'Audiology' (69), 'Gene' (60), 'Lrs' (101), 'Optdigits' (64), 'Promoters' (57), and 'Sonar' (60), all of these instance selection algorithms cannot perform very well over all of these kinds of datasets. When we compare the evolutionary-based algorithms (i.e., GA, CCIS, and GBA), only GBA performs well over the 'Optdigits' dataset. This reveals one major limitation of current instance selection methods for high-dimensional datasets, which can be considered as the future research direction.

On the other hand, there are ten datasets belonging to 10-class classification problems or more, GBA outperforms the other methods. To compare these three evolutionary-based algorithms (i.e., GA, CCIS, and GBA), GBA can make 1-NN provide the lowest error rates over five datasets out of the ten including 'Abalone' (29), 'Opdigits' (10), 'Pendigits' (10), 'Texture' (11), and 'Vowel' (11). CCIS and GA are only good with the 'Soybean' (19) and 'Yeast' (10) datasets, respectively.

4.3 Storage requirement

Table 3 shows the storage reductions achieved by IB3, DORP3, ICF, GA, CCIS, and GBA. Note that 1 means there was no data reduction, and 100 % of the storage space is needed. These results show that on average GBA can filter out the largest number of instances, which requires the least storage space. Particularly, CCIS and GBA perform best over 24 and 21 datasets, respectively.

For the 12 datasets containing medium to large numbers of data samples, GBA performs best over eight datasets, including 'Abalone' (4,177), 'Optdigits' (5,620), 'Page_blocks' (5,473), 'Pendigits' (10,993), 'Phoneme' (5,404), 'Sick' (3,772), 'Texture' (5,500), and 'Waveform' (5,000).

For the six datasets composed of high-dimensional data (i.e., over 50 dimensions), GBA performs best over the 'Gene' (60) and 'Optdigits' (64) datasets and CCIS for 'Audiology' (69), 'Gene' (60), and 'Lrs' (101) and GA for 'Sonar' (60). This again indicates that it is a hard problem for instance selection over high-dimensional datasets.

On the other hand, for the ten datasets belonging to 10-class classification problems or more, only GBA and CCIS can provide the largest dataset reduction, in which GBA performs well over seven datasets including 'Abalone' (29), 'Opdigits' (10), 'Pendigits' (10), 'Primary_tumor' (22), 'Soybean' (19), 'Texture' (11), and 'Vowel' (11) and CCIS for 'Audiology' (24), 'Lrs' (10), 'Vowel' (11), and 'Yeast' (10).

If we examine the lowest error rate versus the least storage requirement by GBA and CCIS, the finding is interesting that using GBA can not only reduce the dataset sizes the most but also allow 1-NN to provide the lowest error rates over 12 datasets including 'Abalone', 'Card', 'Dermatology', 'Optdigits', 'Page_blocks', 'Pendigits', 'Phoneme', Pima', 'Sick', 'texture', 'Vowel', and 'Waveform'. On the other hand, CCIS only exhibits this characteristic over three datasets including 'GERMAN', 'Heart', and 'Post_operative'. Overall, these results demonstrate the superiority of GBA over the state-of-the-art algorithms.

Table 2 Classification error of IB3, DORP3, ICF, GA, CCIS, and GBA

Dataset	IB3	DROP3	ICF	GA (CHC)	CCIS	GBA	1-NN
Abalone	0.835	0.7995	0.8072	0.8554	0.8321	0.7738	0.8034
Anneal	0.3326	0.1551	0.1831	0.2056	0.2023	0.2748	0.0157
Audiology	0.5046	0.6046	0.6091	0.6046	0.5182	0.7503	0.3273
Autos	0.475	0.515	0.56	0.47	0.455	0.5229	0.33
Balance	0.4129	0.292	0.3145	0.2903	0.2371	0.2112	0.2226
Breast-cancer	0.5107	0.3429	0.4214	0.3964	0.3	0.3939	0.3714
Cancer	0.1957	0.1348	0.158	0.1565	0.113	0.0590	0.0479
Card	0.3594	0.3174	0.3116	0.3551	0.2652	0.1648*	0.2174
Dermatology	0.2612	0.1167	0.2889	0.1972	0.1389	0.1097	0.0472
Ecoli	0.4515	0.2727	0.3091	0.3424	0.2757	0.2274	0.206
Gene	0.4025	0.3552	0.4104	0.4025	0.4117	0.3910	0.2647
German	0.439	0.359	0.408	0.383	0.335	0.3770	0.312
Glass	0.4571	0.4048	0.481	0.4286	0.381	0.3015	0.2952
Glass-g2	0.2938	0.2938	0.3563	0.3563	0.3063	0.4198	0.2
Heart	0.3296	0.2889	0.3259	0.3185	0.2296*	0.2296*	0.2333
Heart-c	0.3533	0.29	0.29	0.27	0.2467	0.2920	0.24
Hepatitis	0.3533	0.2067	0.2534	0.24	0.24	0.4899	0.1933
Horse	0.525	0.4194	0.4167	0.4556	0.4472	0.5761	0.3667
Hypothyroid	0.543	0.1833	0.2268	0.2316	0.2456	0.0747	0.0692
Ionosphere	0.32	0.2686	0.2829	0.2829	0.2286	0.2105	0.1314
Iris	0.2667	0.12	0.1533	0.2133	0.1533	0.1067	0.0467
Kr vs. Kp	0.2762	0.2022	0.2204	0.2423	0.2677	0.1774	0.0828
Labor	0.26	0.24	0.32	0.4	0.34	0.3217	0.06
Led24	0.63	0.555	0.615	0.65	0.635	0.6786	0.535
Liver	0.4823	0.4235	0.4324	0.4274	0.4177	0.3708*	0.3794
Lrs	0.4359	0.2491	0.2717	0.3302	0.2302	0.2782	0.1887
Lymphography	0.4643	0.3214	0.3	0.3357	0.3357	0.3974	0.1929
New-thyroid	0.219	0.0952	0.1238	0.1619	0.0429	0.1192	0.0333
Optdigits	0.3338	0.1505	0.208	0.2153	0.2139	0.0538	0.0256
Page-blocks	0.4731	0.1468	0.1658	0.2108	0.223	0.0732	0.0362
Pendigits	0.3155	0.1211	0.1468	0.1865	0.1877	0.0514	0.0066
Phoneme	0.2965	0.2267	0.2622	0.2561	0.2624	0.1717	0.0952
Pima	0.3882	0.3263	0.3487	0.3592	0.3039	0.3073	0.3013
Post-operative	0.6778	0.5111	0.4889	0.3778	0.3444	0.4658	0.4889
Primary-tumor	0.7515	0.6667	0.6819	0.7212	0.7394	0.7015	0.6515
Promoters	0.29	0.27	0.26	0.26	0.41	0.3675	0.25
Satimage	0.3417	0.204	0.2417	0.2603	0.2628	0.1402	0.0927
Segment	0.2857	0.1563	0.1861	0.2125	0.2204	0.1281	0.0355
Sick	0.3769	0.1722	0.1899	0.218	0.2008	0.0863	0.043
Snoar	0.31	0.32	0.44	0.415	0.335	0.3321	0.155
Soybean	0.3941	0.2485	0.2794	0.3191	0.2338	0.2819	0.0779
Texture	0.3131	0.1302	0.1604	0.1977	0.2029	0.0682	0.0105
Tic-tac-toe	0.2242	0.1221	0.1547	0.1979	0.2063	0.3102	0.0779
Vehicle	0.4417	0.3714	0.4203	0.4274	0.4072	0.4374	0.2929
Vote	0.2721	0.2139	0.2442	0.2302	0.186	0.1150	0.0675
Vowel	0.4889	0.405	0.4374	0.5041	0.504	0.2828*	0.2919
Waveform	0.4168	0.3488	0.3826	0.3902	0.393	0.2616	0.286

Table 2 continued

Dataset	IB3	DROP3	ICF	GA (CHC)	CCIS	GBA	1-NN
Wine	0.2588	0.147	0.1529	0.2	0.1412	0.0746	0.0353
Yeast	0.6128	0.5209	0.5473	0.33	0.5061	0.5044	0.4689
Zoo	0.22	0.2	0.2	0.33	0.15	0.1280	0.06
Average	0.3975 (6)	0.2961 (2)	0.329 (4)	0.3365 (5)	0.3093 (3)	0.2929 (1)	0.2053

Table 3 Storage reduction by IB3, DORP3, ICF, GA, CCIS, and GBA

Dataset	IB3	DROP3	ICF	GA (CHC)	CCIS	GBA
Abalone	0.8222	0.2529	0.1839	0.4904	0.4472	0.1326
Anneal	0.2438	0.1271	0.1314	0.0831	0.126	0.0832
Audiology	0.5931	0.2917	0.1397	0.2333	0.101	0.1238
Autos	0.486	0.3935	0.2184	0.2189	0.1043	0.1029
Balance	0.3222	0.2236	0.1732	0.0704	0.0245	0.1274
Breast-cancer	0.202	0.262	0.1818	0.1163	0.0209	0.0200
Cancer	0.0862	0.0643	0.0435	0.0964	0.0095	0.0097
Card	0.1926	0.2811	0.1736	0.1142	0.0205	0.0204
Dermatology	0.2976	0.1467	0.073	0.0891	0.0403	0.0401
Ecoli	0.4184	0.165	0.0904	0.1165	0.0416	0.0426
Gene	0.3518	0.3842	0.2223	0.471	0.1738	0.1738
German	0.2079	0.31	0.1361	0.0826	0.0183	0.1444
Glass	0.4182	0.3145	0.1446	0.1254	0.0829	0.1276
Glass-g2	0.2238	0.334	0.1531	0.1211	0.0571	0.0541
Heart	0.1786	0.2407	0.1304	0.1054	0.028	0.0288
Heart-c	0.2048	0.2397	0.1552	0.0662	0.0268	0.0261
Hepatitis	0.1393	0.175	0.0936	0.1079	0.0322	0.0411
Horse	0.2875	0.2226	0.1326	0.1083	0.0277	0.0264
Hypothyroid	0.1882	0.0393	0.0271	0.2764	0.0776	0.0272
Ionosphere	0.174	0.1592	0.037	0.0877	0.0376	0.1388
Iris	0.243	0.1659	0.1222	0.0993	0.0392	0.0963
Kr vs. Kp	0.2219	0.2514	0.2306	0.2306	0.1371	0.1372
Labor	0.2019	0.3481	0.1269	0.1423	0.0885	0.0909
Led24	0.7084	0.4167	0.2917	0.2128	0.0895	0.0870
Liver	0.2061	0.4357	0.2164	0.1031	0.0428	0.1029
Lrs	0.4226	0.1555	0.0826	0.1086	0.037	0.0373
Lymphography	0.3433	0.309	0.1537	0.1515	0.0574	0.0597
New-thyroid	0.1995	0.1278	0.0706	0.1665	0.0356	0.0361
Optdigits	0.3222	0.0982	0.0558	0.4818	0.3237	0.0557
Page-blocks	0.2087	0.0456	0.0268	0.32	0.0869	0.0268
Pendigits	0.3052	0.054	0.033	0.4985	0.4067	0.0326
Phoneme	0.1533	0.1983	0.1229	0.3142	0.3293	0.1229
Pima	0.1789	0.2552	0.1406	0.103	0.0173	0.0173
Post-operative	0.2198	0.2605	0.2296	0.1667	0.0197	0.0506
Primary-tumor	0.7448	0.2794	0.2213	0.2206	0.3565	0.0772
Promoters	0.2146	0.4458	0.2917	0.225	0.2198	0.2188
Satimage	0.3258	0.1306	0.0613	0.4962	0.118	0.0851
Segment	0.2913	0.1348	0.0957	0.085	0.1362	0.1347
Sick	0.1163	0.0801	0.0541	0.3363	0.0815	0.0540
Snoar	0.1984	0.3431	0.1362	0.084	0.0516	0.1383

Table 3 continued

Dataset	IB3	DROP3	ICF	GA (CHC)	CCIS	GBA
Soybean	0.3665	0.1917	0.1903	0.1062	0.0636	0.0190
Texture	0.3193	0.0973	0.0649	0.4231	0.3217	0.0648
Tic-tac-toe	0.1751	0.2564	0.2672	0.082	0.0348	0.0348
Vehicle	0.3788	0.3215	0.2034	0.1031	0.0493	0.0497
Vote	0.1263	0.1281	0.0947	0.1332	0.0189	0.0190
Vowel	0.3596	0.4112	0.2844	0.1945	0.1223	0.1223
Waveform	0.3563	0.2736	0.1155	0.4487	0.3598	0.1155
Wine	0.2311	0.1721	0.0932	0.1006	0.0367	0.0988
Yeast	0.5559	0.2942	0.1611	0.1407	0.0252	0.0747
Zoo	0.3385	0.2341	0.3033	0.1407	0.0879	0.1398
Average	0.3014 (6)	0.2309 (5)	0.1437 (3)	0.192 (4)	0.1059 (2)	0.0778 (1)

4.4 Time complexity

4.4.1 Computational complexity

Here, we provide a comparison of three algorithms (i.e., GA, CCIS, and GBA) in Big O notion. For the fitness function of each algorithm, the k-NN classifier is used ($k = 1$). Therefore, the complexity of each method is as follows:

$$\begin{aligned}
&\text{GA}: O(N_{Pop} \times Iter \times CFC(D))\\
&\text{CCIS}: O((N_{Pop} + M_{Pop}) \times Iter \times CFC(D))\\
&\text{GBA}: O((N_{Pop} - E) \times Iter \times CEC(D)),
\end{aligned}$$

where N_{pop} is the population size, *Iter* represents the total iteration times, and $CFC(D)$ is the classifier complexity with D instances. The CCIS method has an additional pool for elite individuals, and the size of the elite pool is M_{pop}. However, there is no additional pool of the elite mechanism in GBA since the repeated assessment for the E elite individuals is not necessary in GBA.

4.4.2 Comparison to related works

Besides the Big O analysis, the following experiments provide some quantitative results of comparing GA, CCIS, and GBA in terms of time efficiency. However, it should be noted that the results can only be regarded as an indirect comparison since the computing equipment used in these works are different.

Figure 3 summarizes the average time (seconds) spent by GA (Cano et al. 2003) and GBA. Clearly, GBA takes slightly longer time than GA over most datasets, i.e., 200 vs. 57 s. Furthermore, according to García-Pedrajas et al. (2010) CCIS needs about 1,000 s more than GA over the same 50 datasets. Therefore, we can say that on average GA, CCIS, and GBA require about 1, 16.7, and 3.33 min to perform instance selection, respectively.

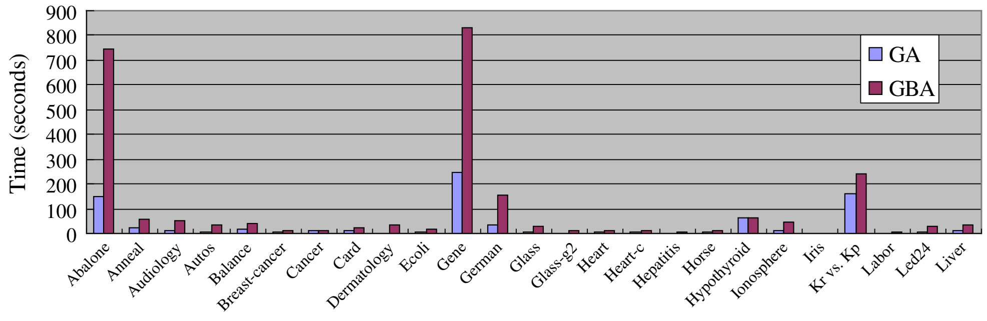
It should be noted that García-Pedrajas et al. (2010) does not provide the information of the experimental environment. Therefore, it may be difficult to directly compare these algorithms. However, CCIS is a distributed-based algorithm whereas GBA and GA are not. Therefore, the distributed characteristic of CCIS should allow it to perform the instance selection task more efficiently than GBA and GA. In other words, the results indicate that the time spent by GBA is reasonable and is a more computationally efficient instance selection method than CCIS.

Next, we further compare the computational cost by GBA and CCIS over larger scale datasets (Fig. 4), high-dimensional datasets (Fig. 5), and the datasets having large numbers of classes (Fig. 6), respectively. Note that the comparison is based on the GA baseline to show whether GBA and CCIS require longer or less computational time of performing instance selection. In other words, the positive and negative bars shown in Figs. 4, 5 and 6 mean that GBA/CCIS require longer and less time for instance selection than GA, respectively. As we can see that GBA significantly outperform CCIS where on average GBA is a more efficient algorithm than CCIS.

5 Conclusion

Instance selection has been recognized as an important data pre-processing step in the data mining and knowledge discovery in databases (KDD) process. In this paper, a novel evolutionary algorithm is proposed by combining biological evolution and the traditional genetic algorithms, namely a GBA, to closely simulate the natural evolution of an algorithm.

To assess the performance of GBA, large-scale experiments were conducted. They were based on comparing five well-known state-of-the-art algorithms, which are IB3, ICF, DROP3, GA, and CCIS, over 50 different domain datasets. First of all, classification error rates were examined. The


(a) Comparisons between GA and GBA over the 25 datasets

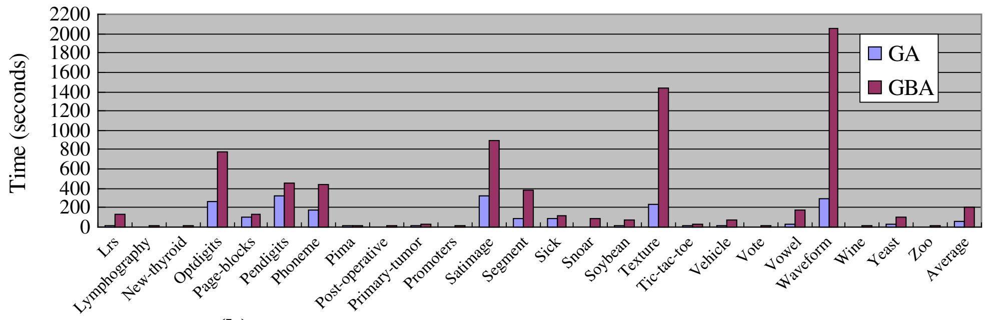

(b) Comparisons between GA and GBA over the other 25 datasets

Fig. 3 Average time spent by GA and GBA

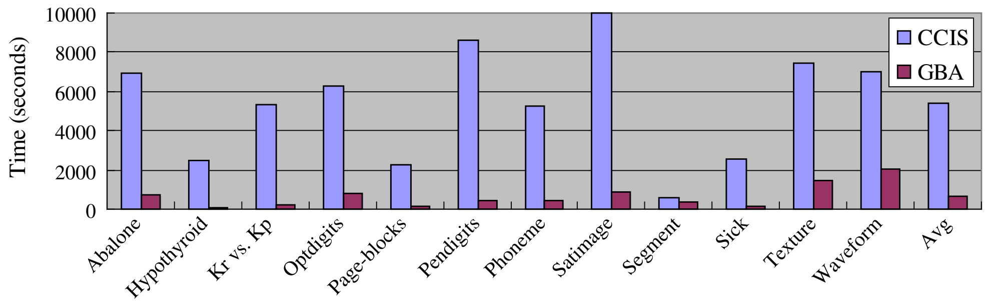


Fig. 4 Average time spent by CCIS and GBA over large-scale datasets

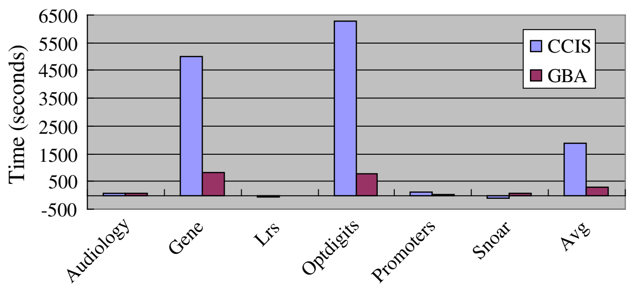


Fig. 5 Average time spent by CCIS and GBA over high-dimensional datasets

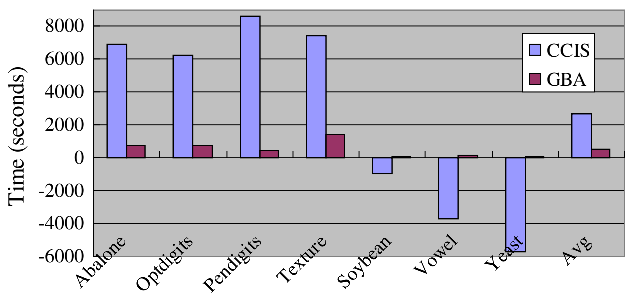


Fig. 6 Average time spent by CCIS and GBA over the datasets having large numbers of classes

experimental results show that on average GBA performs best, which allows the 1-NN classifier to provide the lowest error rates. In particular, it outperforms the other algorithms over 27 datasets. Specifically, 1-NN with GBA performs better than the baseline without instance selection over four datasets.

Moreover, GBA performs better than the others over the datasets containing large numbers of data samples and large numbers of classes. However, none of these algorithms can perform very well over the datasets containing high-dimensional data.

Second, the storage requirements after performing these instance selection algorithms were examined. On average, GBA can filter out the largest number of instances, which requires the smallest storage space. This demonstrates the robustness of GBA that the classification accuracy is not sacrificed in return for the larger reduction rate. For most of the datasets containing large numbers of data samples and large numbers of classes, GBA performs better than the others. However, this examination shows that these algorithms are limited to when handling high-dimensional datasets.

Finally, the three evolutionary algorithms, i.e., GA, CCIS, and GBA, were compared in terms of time complexity. GA is the most efficient algorithm since they are based on simple rules. However, GBA only takes a slightly longer time than GA over most datasets. Moreover, on average, GBA requires significantly shorter time than CCIS, especially when the datasets contain large numbers of data samples, high-dimensional data, and large numbers of classes.

Several issues can be considered as the future works. First, the computational cost of the current evaluation function is certainly high. Therefore, other efficient evaluation functions can be taken into account for GBA. Second, since the dataset size is increasing rapidly, some related techniques and platforms can be employed to solve the big data problem, such as parallel computing and cloud computing, etc. For example, Nojima et al. (2009) have shown that genetic-based algorithms can be ported to the cloud computing platform or parallel system. For GBA, the great migration mechanism can take advantage of the cloud computing platform. This is because the great migration can be regarded as the social activities between two or more populations, which can improve the performance of evolutionary algorithms (Uludağ et al. 2013; Xie et al. 2014).

Appendix: The schema theorems corresponding to GA and GBA

The original model of GA is

$$m(H,t+1) = m(H,t) \times \frac{f(H)}{\overline{f}} \times \left[1 - r_c \frac{\delta(H)}{l-1} - o(H) r_m\right],$$

where H represents the schema, t is the generation, $m(H,t)$ is the number of strings belonging to schema H at generation t, $f(H)$ is the observed fitness, r_c is the crossover rate, $\delta(H)$ is the defining length, l is the length of the code, r_m is the mutation rate, and $o(H)$ is the order of a schema.

The modified model of GBA is

$$m(H,t+1) = m(H,t) \times \frac{Nf(H)}{\overline{Nf}} \times \left[1 - r_c \frac{\delta(H)}{l-1} - o(H) r_m - o(H) r_{mg} \times MGT(m(H,t),t)\right] + GK(H),$$

where H represents the schema, t is the generation, $m(H,t)$ is the number of strings belonging to schema H at generation t, $Nf(H)$ is the nonlinear fitness functions, r_c is the crossover rate, $\delta(H)$ is the defining length, l is the length of the code, r_m is the mutation rate, $o(H)$ is the order of a schema, r_{mg} is the great migration rate, $MGT(m(H,t),t)$ is the trigger of the Great Migration, and $GK(K)$ is the genetic king protection mechanisms, which can retain the good schema H.

The definition of nonlinear fitness functions *Nf*(H) is

$$Nf = \frac{Hyperbolictangent\left(\frac{f(H)-0.5}{\sigma^2}\right)+1}{2}$$

It will increase/reduce the fitness strength depends on the threshold.

In addition, the *GK*(H) represents the Genetic King Protection Mechanisms, and the definition of *GK* is

If $Nf(H) \geq Threshold$ then $GK(H) = 1$ else $GK(H) = 0$.

If the schema H is good enough, it will be retained by genetic king protection mechanisms.

The definition of great migration $MGT(m(H,t),t)$ is

If the best fitness value is stable then $MGT(m(H,t),t) = 1$, else $MGT(m(H,t),t) = 0$.

If the best fitness value is stable, then apply the great migration (A strong mutation) (c.f. Fig. 6 for the pseudo code of GBA).

References

Aggarwal CC, Yu PC (2001) Outlier detection for high dimensional data. In: Proceedings of the ACM SIGMOD conference, pp 37–46

Aha DW, Kibler D, Albert MK (1991) Instance-based learning algorithms. Mach Learn 6(1):37–66

Ball P (2002) Natural strategies for the molecular engineer. Nanotechnology 13:R15–R28

Barnett V, Lewis T (1994) Outliers in statistical data. Wiley, Hoboken

Brighton H, Mellish C (2002) Advances in instance selection for instance-based learning algorithms. Data Min Knowl Discov 6:153–172

Cano JR, Herrera F, Lozano M (2003) Using evolutionary algorithms as instance selection for data reduction: an experimental study. IEEE Trans Evolut Comput 7(6):561–575

Derrac J, García S, Herrera F (2010) A survey on evolutionary instance selection and generation. Int J Appl Metaheur Comput 1(1):60–92

Ellstrand NC (2003) Dangerous liaisons: when cultivated plants mate with their wild relatives. Johns Hopkins University Press, Baltimore

Emigh TH (1980) Comparison of tests for Hardy–Weinberg equilibrium. Biometrics 36(4):627–642

Flynn JJ, Wyss AR (1998) Recent advances in South American mammalian paleontology. Trends Eco Evol 13(11):449–454

García-Pedrajas N, del Castillo JAR, Ortiz-Boyer D (2010) A cooperative coevolutionary algorithm for instance selection for instance-based learning. Mach Learn 78:381–420

Guyonm I (2003) An Introduction to variable and feature selection. J Mach Learn Res 3:1157–1182

Holland JH (1992) Adaptation in natural and artificial system: an introductory analysis with applications to biology, control, and artificial intelligence. A Bradford Book, Chester

Jain AK, Duin RPW, Mao J (2000) Statistical pattern recognition: a review. IEEE Trans Pattern Anal Mach Intell 22(1):4–37

Jankowski N, Grochowski M (2004) Comparison of instances selection algorithms I: algorithms survey. International conference on artificial intelligence and soft computing, pp 598–603

Jing SY (2013) A hybrid genetic algorithm for feature subset selection in rough set theory. Soft Comput 18:1373–1382

Knorr EM, Ng R, Tucakov V (2000) Distance-based outliers: algorithms and applications. VLDB J 8:237–253

Koepfli KP, Gompper ME, Eizirik E, Ho CC, Linden L, Maldonado JE, Wayne RK (2007) Phylogeny of the Procyonidae (Mammalia: Carvnivora): molecules, morphology and the Great American interchange. Mol Phylogenet Evol 43(3):1076–1095

Li X-B, Jacob VS (2008) Adaptive data reduction for large-scale transaction data. Eur J Oper Res 188(3):910–924

Liu H, Motoda H (2001) Instance selection and construction for data mining. Kluwer, Boston

Morgan GS (2002) Late Rancholabrean mammals from southernmost Florida and neotropical influence in Florida pleistocene faunas. Smithson Contrib Paleobiol 93:15–38

Nojima Y, Ishibuchi H, Kuwajima I (2009) Parallel distributed genetic fuzzy rule selection. Soft Comput 13:511–519

Odum HT (1994) Ecological and general systems: an introduction to systems ecology. University Press of Colorado, Niwot

Pollan M (2001) The year in ideas. A-Z. Genetic pollution, The New York Times

Pyle D (1999) Data preparation for data mining. Morgan Kaufmann, Burlington

Pradhan S, Wu X (1999) Instance selection in data mining. Technical report. Department of Computer Science, University of Colorado at Boulder

Quinlan JR (1986) Induction of decision trees. Mach Learn 1:81–106

Reinartz T (2002) A unifying view on instance selection. Data Min Knowl Discov 6:191–210

Stern C (1962) Wilhelm Weinberg. Genetics 47:1–5

Uludağ G, Kiraz B, Etaner-Uyar AŞ, Özcan E (2013) A hybrid multi-population framework for dynamic environments combining online and offline learning. Soft Comput 17:2327–2348

Wilson DL (1972) Asymptotic properties of nearest neighbor rules using edited data. IEEE Trans Syst Man Cybern 2(3):408–421

Wilson DR, Martinez TR (2000) Reduction techniques for instance-based learning algorithms. Mach Learn 38:257–286

Xie XF, Liu J, Wang ZJ (2014) A cooperative group optimization system. Soft Comput 18:469–495