Journal of Theoretical and Applied Information Technology

www.jatit.org

MINING CLASSIFICATION RULES IN THE PRESENCE OF CONCEPT DRIFT WITH AN INCREMENTAL GENETIC ALGORITHM

[1,2]I-Hui Li, [1 *]I-En Liao, [3]Wei-Zhi Pang

[1]Department of Computer Science and Engineering, National Chung Hsing University, Taichung 402, Taiwan

[2]Department of Information Management, Ling Tung University, Taichung 408, Taiwan

[3]Department of Applied Mathematics, National Chung Hsing University, Taichung 408, Taiwan

E-mail: phd9301@cs.nchu.edu.tw, ieliao@nchu.edu.tw, eijyle@ms.chgsh.chc.edu.tw

ABSTRACT

Traditional classification algorithms are ideally suited to the processing of small datasets with a stationary distribution, and therefore yield significant errors when applied to real-world datasets subject to concept drift. In the current study, this problem is resolved using an incremental genetic algorithm (IGA). An assumption is made that new training data are generated at a steady rate and pass through a fixed-size window. In the initialization process, training samples are accumulated until the window is full, and a genetic algorithm (GA) is then applied to determine the set of classification rules. As new training samples arrive in the window, old instances are forgotten. Once all the original samples have been replaced by new samples, the GA is re-executed to determine the new set of best classification rules. This procedure is repeated sequentially for as long as a learning function is required. To account for concept drift, the GA utilizes a memory-based random immigrant module, in which the initial population pool of the GA applied at each stage of the incremental learning process comprises a mix of best solutions obtained in the previous stage and an appropriate number of random immigrants. The feasibility of the proposed approach is confirmed by performing a series of classification rules mining simulations using two standard datasets, namely Mushroom and Zoo. The results demonstrate that IGA achieves a comparable classification performance to that obtained using existing incremental and non-incremental methods, but incurs a significantly lower computational overhead.

Keywords: Classification, Genetic Algorithm, Incremental Learning

1. INTRODUCTION

The term "data mining" describes the process of analyzing large volumes of data from different perspectives in order to discover knowledge which is highly accurate, comprehensible, and "interesting" (i.e. surprising or novel) [1]. The objective of classification rules mining is to search a dataset for a small number of rules to serve as classifiers for predicting the class of any new instance. Given the huge volumes of data stored in modern databases, the data mining and classification rules mining tasks are invariably performed using some form of automated approach, such as statistical-based algorithms, machine-learning schemes, neural networks, and so forth. For example, Liu [2] demonstrated the use of an enhanced ant colony optimization algorithm in performing classification rules discovery for two standard problems. Tan [3] argued that all classification algorithms should be characterized by a high degree of accuracy, simplicity and efficiency. However, traditional algorithms are generally based on the assumption that the input data are drawn randomly from a stationary distribution, i.e. they are all generated by the same concept. Therefore, such schemes perform poorly when used to infer classification rules for large real-world datasets with a time-varying characteristic [4][5]. Such datasets are liable to a concept-drifting effect [6], i.e. the properties of the target concept change over time as a result of changes in the underlying context,

* Corresponding author.

and thus the classification rules become "out of date". For example, consider the case shown in Figure 1 in which the black circles represent 'positive instances', the white circles denote 'negative instances', and the objective is to locate the position of the optimum boundary between them. The three images illustrate the time-varying distributions of the positive and negative instances and clearly show that the position of the boundary must be dynamically adjusted in accordance with changes in the dataset distribution in order to maintain its optimality.

Zhang [7] argued that since snapshots of dynamic datasets whose contents vary over time are almost certain to overlap to a greater or lesser extent, a certain degree of commonality inevitably exists in the mining results obtained at different moments in times. For example, as shown in Figure 2, the mining results obtained at moment D' differ from those obtained at the previous moment (D) via the addition of Δ^+ and the loss of Δ^-, but have the contents D' in common. Therefore, in attempting to infer suitable classification rules for such databases, it is necessary to apply some form of adaptive scheme capable of detecting and reacting to the changes which take place in the database over time.

In the present study, the problem of classification rules mining in the presence of concept drift is solved using an IGA. The GA is a particular form of evolutionary scheme designed to derive exact or approximate solutions to search or optimization problems. In the GA procedure, candidate solutions are represented as chromosomes, typically in the form of a binary data string, and the chromosome population is processed iteratively using biologically-inspired selection, crossover and mutation operations. The overall objective of these operations is to ensure the "survival of the fittest" such that the optimality of the chromosomes is progressively improved from one generation (i.e. iteration) to the next. GAs are widely applied in a diverse range of fields, including computer science, economics, manufacturing, physics, and so forth, and represent an ideal solution for the learning of classification rules [8]. In the IGA scheme proposed in this study, an assumption is made that new training instances become available at a constant rate and pass through a static window capable of storing W training samples. During the initialization phase, training instances are accumulated within this window until it become full and a GA with a randomly created chromosome population and a local heuristic search scheme is then activated to determine the best classification rules. These classification rules (i.e. chromosomes) are then mapped to memory. Since the window has a fixed size and the samples arrive at a constant rate, each time a new sample enters the window, an old sample is dropped from the window. After a certain elapsed time (the length of which depends on the arrival rate of the training instances), all of the old samples within the window are forgotten and are replaced by new samples. At this point, the GA is re-executed to determine the new best set of classification rules. To enable the GA to react to concept drift, the initial population is generated using a memory-based random immigrant scheme. That is, the population comprises a mixture of the best of the solutions obtained using the data within the previous window and an appropriate number of randomly generated new chromosomes. The number of new chromosomes is determined in accordance with the extent of the concept drift. Having created the population pool, conventional crossover, mutation and selection operations are performed iteratively until a new best set of solutions has been obtained. These solutions are then mapped to memory in place of those determined in the previous stage. This sequential process continues for as long as an incremental rules learning function is required. The performance of the IGA algorithm is benchmarked against that of the incremental decision tree algorithm presented in [9] and a conventional non-incremental GA classifier using the standard Mushroom and Zoo datasets for illustration purposes.

The remainder of this paper is organized as follows. Section 2 reviews the related literature in the data mining and incremental rules inference field, while Section 3 formulates the classification problem considered in this study, describes the basic architecture of the IGA system, and discusses the sequential operation of IGA. Section 4 introduces the major genetic operations performed in executing the IGA scheme and describes the fitness function applied to evaluate the quality of the emerging solutions. Section 5 presents the performance evaluation results. Finally, Section 6 presents some brief concluding remarks and indicates the intended direction of future research.

2. RELATED WORK

This section reviews some of the major classification schemes and techniques, namely incremental decision tree algorithms [9][10][11] [12][13][14], incremental GAs [15], and GA-based multiagent environments [16].

2.1. Incremental Decision Tree Algorithms

Decision trees are commonly used in data mining and machine learning applications to map observations about a particular item of interest to conclusions regarding its value in order to support a decision making process. Broadly speaking, decision tree algorithms can be classified as either non-incremental or incremental. In the former case, the algorithm is essentially a one-shot process in which the underlying concept of the data within the database is inferred just once in accordance with all the training instances available at that time. However, this method is clearly unsuitable for the classification of dynamic datasets whose contents and structures vary over time. Thus, various researchers have developed incremental decision tree algorithms for the classification of datasets in which new instances arrive sequentially in time [9][10][11][12][13][14]. The underlying principle of these algorithms is to map new arrivals to the existing tree structure wherever the attributes of this arrival make this possible, or to modify the relevant sub-tree within the decision tree if a direct mapping cannot be achieved. Thus, incremental decision tree algorithms avoid the requirement to reconstruct the tree from scratch each time a new instance arrives and are therefore ideally suited to the classification of dynamic datasets. As described in the following sections, incremental decision tree algorithms use either a fixed window size method or an adaptive window size method[9].

Fixed window size methods

As implied by their name, fixed window size methods infer the classifiers of the decision tree in accordance with the training data contained within a window of a fixed size, i.e. as new data instances enter the window, an equivalent number of the oldest examples within the window are automatically forgotten and excluded from the inference process. In practice, defining an appropriate window size involves obtaining an acceptable compromise between a more rapid adaptability to environmental change afforded by a small window and a more stable performance (given negligible concept drift) achieved using a larger window. Typical examples of fixed window size methods include the VFDT (Very Fast Decision Tree Learner) scheme [11][17] and the CVFDT (Concept-adapting Very Fast Decision Tree Learner) scheme [12][17]. In developing the VFDT scheme, Domingos [11] contended that the best attribute to test at any given node within the decision tree could be determined on the basis of a small number of training examples passing through that node, i.e. it was unnecessary to consider the entire training set. The problem of determining an appropriate number of instances for testing purposes at each node was evaluated statistically using a Hoeffding bound method designed to ensure that the attribute chosen for a node on the basis of a small number of samples was identical to that which would otherwise be chosen if the entire training set was considered. This was achieved by accumulating data instances from the training stream until the Hoeffding bound reduced to a value less than the difference between the observed values of the two attributes with the highest and second highest heuristic values, respectively.

In common with most traditional clustering and mining algorithms, VFDT assumes that the training samples used to infer the underlying structure of the dataset are drawn at random from a stationary distribution. In practice, however, the contents and structure of modern datasets invariably vary over time, and thus this assumption does not hold. Accordingly, in a later study [12], Domingos developed the CVFDT scheme in which the original VFDT method was extended by sliding a fixed-size window over the training dataset. Rather than reconstructing a new decision tree each time a new sample arrived, CVFDT simply updated the statistics associated with each node within the existing tree by incrementing the counts corresponding to the new sample and decrementing those relating to the oldest. In dynamic datasets, this updating procedure has the effect that some of the splits in the decision tree which previously satisfied the Hoeffding bound now fail to do so, and thus the CVFDT scheme constructs an alternative sub-tree using the new best attribute as the root. When the classification accuracy of this sub-tree is determined to be more accurate than that of the original sub-tree, the original sub-tree is pruned from the tree and its place is taken by the new sub-tree. As a result, CVFDT not only avoids the requirement to rebuild the entire decision tree each time a context change is detected, but also maintains a high level of classification accuracy.

Adaptive window size methods

The basic concept of adaptive window size strategies is to scale the size of the window dynamically in response to changes in the perceived level of concept drift within the dataset. Specifically, the window size is decreased in the event of concept drift, but is increased (or maintained) in the event of a stable concept. Two of the most well known adaptive window size methods are the WAH (Window-Adjustment-Heuristic) used in the FLORA family of algorithms developed by Wider [13] and the DNWS (Determine New

Window Size) heuristic developed by Klinkenberg [9].

WAH was first proposed as a means of combining robustness to noise with a sensitivity to concept drift in rough set analysis applications, but was later applied by Maloof [10] to create an effective rules learner for datasets characterized by changing concepts. In WAH, concept changes are detected by monitoring the system's predictive performance over time and analyzing the syntactic properties of the evolving hypotheses [13]. In the event that concept drift is discerned, WAH automatically responds by reducing the window size by 20%. However, if the concept is deemed to be extremely stable, WAH adds one new example to the window and deletes two old examples, thereby reducing the window size by 1 unit. If the concept appears to be sufficiently stable, the window size is neither increased nor decreased, but remains unchanged. Finally, in the event that none of these concept stability conditions apply, WAH assumes that more information is required and increases the window size by 1 unit by accepting a new sample into the window whilst simultaneously retaining the oldest sample.

Although WAH solves the problem of concept-drift inherent in real-world datasets, it is computationally intensive, and is therefore only really practical for the classification of small datasets. Accordingly, Klinkenberg [9] developed an adaptive window size adjustment heuristic (designated for convenience hereafter as DNWS) in which a filtering process was applied to determine the extent of the concept change and to adjust the window size accordingly. In DNWS, concept change is detected by monitoring the performance of a classifier in terms of its accuracy, recall and precision. For each of these three indicators, the average value and standard sample error are computed on the basis of *M* batches at the last time step and are then compared to a specified confidence interval. In the event that the value of one of the indicators falls below this confidence interval, an assumption is made that the concept has changed, and a further test is performed to determine the extent of this change, i.e. a small and gradual change (indicating concept drift) or a large and sudden change (indicating concept shift). If the test reveals the occurrence of concept shift, the window is immediately reduced to its smallest size. By contrast, in the event of concept drift, the window is reduced more gradually at a user-specified rate. If the indicator values reveal that the existing concept is relatively stable, the window size remains unchanged and all the current examples are stored for future reference in order to improve the performance of the classifier.

2.2. Incremental Genetic Algorithms

GAs are used in a diverse range of fields to solve a variety of search and optimization problems. Whilst GAs were originally intended for the analysis of static datasets, in more recent years they have also been applied to the problem of classifying large, time-varying datasets. However, traditional GAs often fail to converge properly when applied to dynamic datasets since they lack a suitable mechanism with which to respond to changes in their environment [9]. Thus, a new class of GA known as Incremental GAs has emerged in recent years [15]. As described in the sections below, these algorithms typically use random immigrant models [18], memory-based schemes [19] or multi-population methods [20] to enable their application to search and optimization problems characterized by changing fitness landscapes.

Random immigrant schemes

In GA schemes of this type, the problem of a dynamically changing dataset is resolved by replacing an individual (or individuals) within the population by a randomly generated individual (or individuals) in each evolutionary generation (i.e. in each iteration of the algorithm). In general, one of two different strategies may be applied in selecting the existing individual(s) for substitution, namely a purely random approach or a selective procedure in which the individual(s) with the poorest fit(s) amongst all the solutions within the population pool is(are) replaced. In introducing random immigrants into the population pool, the aim is to increase the level of diversity within the population, thereby expanding the search space and enhancing the likelihood of the GA converging to the global optimal solution rather than becoming trapped at a local sub-optimal solution.

Memory-enhanced schemes

In memory-enhanced schemes, previous optimal solutions are retained in memory and are reactivated as appropriate in response to changes in the environment. Such schemes are particularly well suited to dynamic datasets with a periodic characteristic. However, they perform less well when applied to problems with a rapidly changing fitness landscape since the probability of previous solutions being exactly applicable to the new landscape is inevitably reduced. Furthermore, significant memory resources are required to store the chromosome information, and thus appropriate strategies must be devised to partially replace the contents of the memory when it becomes full.

Memory-based immigrant schemes

Memory-based immigrant schemes combine the respective advantages of memory schemes and random immigrant schemes by storing the best chromosomes in memory for future reuse and then retrieving these chromosomes and using them to create random immigrants to replace the worst individuals in the population.

Multi-population schemes

Traditional GAs operate upon a single population of candidate solutions and have a proven ability to solve a wide variety of search and optimization problems. However, even better results can be achieved by partitioning the search space into multiple sub-populations and then solving the problem using a multi-population GA (MGA). In an MGA, each sub-population is allowed to evolve independently for a specified number of generations, and then one or more individuals are migrated between the sub-populations before allowing the solution procedure to continue. MGAs tend to be both quicker and more accurate than conventional single-population GAs. However, their performance is critically dependent upon an appropriate choice of parameter settings, namely, the migration method, the migration interval, the migration rate, the number of sub-populations, and so on.

2.3. Incremental Approach to GA-based Classification

Reviewing the literature, it is found that the problem of incremental learning in the presence of concept drift is generally addressed using statistical type algorithms or neural networks rather than GAs. To address this perceived gap in the literature, Guan [16] developed a GA-based incremental learning scheme for classification purposes. The authors argued that classification problems may involve three different types of change, namely (1) the arrival of new data, allowing the existing solutions to be further improved; (2) the detection of new attributes, allowing new classification behaviors to be identified; and (3) the detection of new classes, allowing the classification structure to be improved.

Whilst the authors conceded that these changes could be handled simply by rerunning a GA to rebuild the classification rules from scratch, they suggested that an incremental type approach was better suited to satisfying the time and resource constraints imposed in typical real-world applications. Accordingly, they proposed the multiagent environment, in which multiple classifier agents, each of which was based on a GA, collaborated with one another by monitoring incremental changes in the environment and then exchanging information regarding newly-detected training data, attributes and classes, and so on, such that the system collectively converged toward the optimal set of classification rules. In responding to changes in the environment, the GAs were designed to insert new elements into an old solution (i.e. an existing chromosome / rule) in order to form a new rule. Four specific methods were considered for integrating the old and new elements, namely (1) choosing the best of the old chromosomes and then adding randomly created new elements; (2) choosing the best of the old chromosomes and then adding elements provided by other classifier agents; (3) adding randomly-created elements to all of the old chromosomes, and (4) adding new elements provided by other classifier agents to all of the old chromosomes.

The experimental results confirmed the ability of the proposed scheme to amend the rule set by integrating the new input attributes with the existing input space rather than rebuilding the rules from scratch. Furthermore, it was shown that the scheme improved both the learning time and the quality of the classification performance compared to that achieved by retraining the GA each time a change occurred in the dataset.

3. INCREMENTAL GENETIC ALGORITHM FOR CLASSIFICATION RULES MINING

Since in real-world datasets the level of concept-drift may well vary over time, incremental decision tree methods based upon the use of a fixed window size heuristic are liable to generate significant errors when used for rules inference purposes. Furthermore, while the problem of concept drift can be resolved to a certain extent by integrating the classification system with some form of adaptive window scaling method, the performance of such schemes is critically dependent upon the choice of parameters assigned to the scaling heuristic, and in practice, these parameter values are not intuitively obvious to general data mining practitioners. The multiagent approach proposed by Guan [16] has the proven ability to accomplish classification rules mining in the presence of concept drift. However, the implementation of multiple GA-based classifiers is somewhat complicated and expensive for practical applications.

Nonetheless, the results presented in [16] confirm the ability of GAs to adapt dynamically to incremental changes of various types in the dataset (i.e. data, attribute or class changes) and to evolve a new best rule set for classification purposes without

the need to rebuild the entire rule set from scratch. Therefore, as described in Section 1, the current study proposes a scheme designated as IGA featuring a memory-based random immigrant GA and a local heuristic search method for classification rules mining in the presence of concept drift. The remainder of this section is organized as follows: Section 3.1 formulates the classification problem considered in this study, Section 3.2 describes the basic architecture of the IGA system, and Section 3.3 discusses the sequential operation of IGA.

3.1. Problem Definition

An assumption is made that all the samples are tuples of the dataset. Let S be the universal set of all the samples. Furthermore, let each sample be denoted by $X=(x_1,x_2,\ldots,x_n)$, $\forall X \in S$, where n is the total number of attributes associated with the sample X and x_i is the ith attribute in sample X. In addition, let $D=(X_1,X_2,\ldots,X_k)$, where k is a positive integer such that $D \subseteq S$ is a set of k samples. Given an assumption that the target attribute has m possible values, the set of target attribute values is given by $C=(c_1,c_2,\ldots,c_m)$, where $1 \le i \le m$, and c_i is the value of the ith target attribute.

The classification function f is defined as f: $D \rightarrow C$, thus $f(D)=(\lambda_1, \lambda_2,\ldots, \lambda_k)$.

where λ_i denotes the target attribute value of sample X_i in set D.

In other words, classifier f assigns a target attribute value λ_i for each sample X_i within set D. Having executed the classifier, the result $\lambda_i \in C$ indicates that the sample has been successfully classified; else X_i cannot be successfully classified by f.

As described above, ITGA assumes that new data instances are generated at a constant rate. The data accumulated at time reference t form a dataset designated as S_t, and are used by the GA to infer the corresponding best classification rules function $f_t(X)$. Similarly, at time point $t+1$, the data accumulated in the interval since time t are assigned to a new dataset S_{t+1} and the GA searches this dataset iteratively for a new best rule set, i.e. $f_{t+1}(X)$, taking $f_t(X)$, suitably modified by the random immigrant module, as the initial population pool.

3.2. Basic Architecture of IGA Scheme

According to Darwin's theory of evolution, biological species are geared toward the "survival of the fittest". That is, a process of natural selection ensures that advantageous genetic mutations are accumulated and passed down through the generations such that the superior members of the population survive, whilst the inferior members gradually die out. According to Darwin, individuals with an adaptive characteristic are more likely to be selected for reproduction, and thus over a long period of time, a population becomes well adapted to a stable environment or adjusts itself in accordance with changes in the environment, possibly by breeding with outsiders, such that its long term survival can be assured, albeit possibly in a different form.

In GAs, the solution procedure is analogous to that of natural evolution in the sense that the potential solutions to the problem of interest are encoded as chromosomes which are then iteratively processed using selection, crossover, mutation and evaluation operations such that best solutions gradually emerge. In the incremental GA implemented in the current study, the quality of the final solutions is improved by utilizing the memory-based immigrant scheme presented in [12]. In other words, the best chromosomes obtained in one execution of the GA (e.g. at time t) are saved to memory and are used as the basis for the starting population of the GA executed at time $t+1$. Depending on the extent of the change which takes place in the environment in the interval between time t and time $t+1$, an appropriate number of new chromosomes are randomly generated and used to replace the chromosome(s) with the poorest fit in the stored population pool. For example, consider the case where the stored population contains 10 chromosomes and the environment changes by 30% (i.e. the training samples change by 30%) Under these conditions, 3 new chromosomes are randomly generated and used in place of the 3 chromosomes with the poorest fit in the stored population. Having created the new population, the selection, crossover, mutation and evaluation operations are then performed once again attain to generate a new population of best chromosomes.

Figure 3 illustrates the basic architecture of the IGA scheme. As shown, the system comprises a total of five different modules, namely (1) the data pre-processing & population initialization module, (2) the genetic operations module, (3) the fitness function calculation and evaluation module, (4) the heuristic search module, and (5) the memory-based random immigrant module. Basically, the system commences by pre-processing the training data and encoding the new instances in the form of chromosomes. The genetic operation module, memory-based immigrant module and heuristic search module are then employed to generate new candidate solutions, which are screened using the

fitness function module to generate a new population of best solutions which are then mapped to memory. The details of the data pre-processing, immigrant migration and heuristic search modules are described in the sections below, whilst the genetic operations module and fitness function module are discussed in Section 4.

(1) Data preprocess & population initialization module

The data preprocess task commences by checking the suitability of the attributes of the new instances. Any attributes which are completely different are deleted directly, while for those attributes which are completely the same, the characteristic attributes are recorded but the attributes are deleted. Also, for each attribute, the value which appears the most frequently is recorded for further use by the later modules in the IGA scheme. The chromosome population used in the initialization process is then randomly generated.

In the present GA, each chromosome represents a classification rule and has the form of a binary string comprising a notional $n+1$ partitions corresponding to n input attributes and one target attribute to express the outcome of the classification rule. Assuming that an attribute can take k different values, then this attribute requires the use of $k+1$ binary bits for encoding purposes, i.e. k bits to represent the k different attribute values and an addition bit to indicate whether or not this attribute forms part of the classification rule. When encoding each attribute, the left-most bit is used to indicate the usage state of the attribute, and is set to "0" if the attribute is used within the classification rule, and to "1" if it is not. Clearly, each attribute can only take one value at any moment in time, and thus of the remaining k bits, just one bit has a value of "1", while the remainder have a value of "0". Note that each classification rule inevitably yields a classification outcome, and thus the target attribute has no need for a usage bit to indicate whether or not the attribute is used. In other words, every bit within the string associated with the target attribute corresponds to a potential attribute value.

Consider the case of a training sample with three input attributes A1, A2 and A3. Furthermore, assume that these three attributes can take 4, 2 and 5 different values, respectively. Finally, assume that the target attribute has 3 possible values. Table 1 presents a typical chromosome coding for a possible classification rule for this sample. This coding shows that the classification rule is based upon Attributes 1 and 3 only, and is specified in terms of their first and third possible values, respectively. Given these particular attribute conditions, the classification rule assigns the input sample to the class represented by the third value of the Target attribute. In other words, this particular chromosome expresses the following rule: IF (A1=value1) AND (A2=Empty) AND (A3=value3), THEN (Target=value3).

(2) Memory-based immigrant module

In the incremental GA implemented in this study, the best chromosomes generated by the iterative solution procedure are automatically saved to memory to provide the basis for the starting population used by the GA applied to the following set of training data. Since the aim of IGA is to react to incremental changes in the environment, the memory-based scheme used in the GA is integrated with a random immigrant module in order to increase the diversity level of the population pool, thereby expanding the search space and improving the optimality of the final solution. In the proposed approach, an assessment is made of the percentage change in the environment, and a corresponding number of new chromosomes are then randomly generated and used to replace an equivalent number of chromosomes in the stored population pool. Note that in the substitution process, the algorithm automatically replaces the stored member(s) with the lowest fitness value(s).

(3) Heuristic search module

In searching for the solution to a problem, GAs iteratively apply some form of local heuristic search technique to gradually reduce the size of the search space such that the solution procedure converges to the globally best outcome. As discussed below, the present GA applies two different heuristic search strategies.

In the first strategy, the value which appears most frequently for each attribute is recorded during the pre-processing stage and is then replaced within each chromosome generated by the crossover operation in the GA solution procedure by a randomly selected value. If the value of the fitness function is improved, the randomly selected value is accepted as the new attribute value; else the original value is restored. In general, if an IF-THEN classification rule has too many attributes, it cannot be intuitively understood by the mining practitioner. Thus, in the second search strategy, following the crossover operation, one attribute in the chromosome string is chosen at random and its first bit is inspected to determine whether or not it currently forms part of the classification rule. If the attribute does indeed form part of the classification rule, it is automatically discarded, i.e. the first bit is

changed from "0" to "1". The fitness of the resulting chromosome is then re-evaluated. If the fitness value is found to improve, then the attribute is discarded from the classification rule, else it is retained.

3.3. Sequential Implementation of IGA

In the present GA, the data stream is processed using the fixed window size method presented in [21]. The window is assumed to have a size *W*, i.e. it can hold a total of *W* training instances. When the rules classification system is initialized at time t_0, a total of *W* training samples are accumulated, and the GA is then employed to establish the best classification rules based upon this particular set of training samples. An assumption is made that the new training instances arrive at a constant rate, and therefore, each time that a new training instance arrives, an old training sample is forgotten. As a result, following a time interval governed by the value of the arrival rate, the original *W* training samples within the window are completely replaced by a new set of *W* training examples. At this time, designated as time t_1, the GA is re-executed to determine the new best classification rules. As shown in Figure 4, this process is repeated sequentially at regular time intervals for as long as an incremental learning capability is required.

Figure 5 presents a simple schematic of the *i*th stage of the IGA scheme. As shown, the initial population of the GA applied at this particular stage of the learning process comprises a subset of the best classification rules established in the previous stage of the solution procedure and an appropriate number of randomly generated classification rules. The best classification rules determined in this stage of the leaning process are saved to memory and are then used as the basis for the population used in the following stage of the learning procedure. As stated above, this procedure continues sequentially until the specified number of training stages has been completed or a self-learning capability is no longer required. The use of the memory-based random immigrant module in initializing the population of the GA at the beginning of each stage ensures that the IGA scheme is both computationally efficient and robust to concept drift. In the event that the target concept remains stable, the best solutions obtained at the previous stage are simply reused as the initial population for the following stage, i.e. there is no need to repopulate the search space each time the GA is run. However, in the event that concept drift occurs, the system adapts to this change by automatically replacing an appropriate number of the original classification rules with new random rules and then searches for an updated set of best rules.

4. GENETIC OPERATORS IN IGA SCHEME

This section describes the core operations within the GA scheme implemented in this study, namely the genetic operations and the fitness function calculation. Figure 6 presents a flowchart showing the iterative GA solution procedure. As shown, the procedure commences by retrieving the best solutions from the previous stage and replacing certain of these solutions with randomly generated new chromosomes. Having created the initial population, crossover, heuristic search, mutation and selection operations are performed to create a new population. The fitness functions of the members of this population are then determined to establish whether or not they satisfy the target criteria. If these criteria are satisfied, the GA terminates and the current population is mapped to memory. However, if the termination criteria are not satisfied, the crossover, search, mutation and selection operations are repeated to generate a new candidate set of classification rules. This procedure continues iteratively until the chromosomes within the population pool meet the target requirements, at which point the GA terminates.

4.1 Genetic Operations Module

In addition to the heuristic search mechanism (described in Section 3.2), the Genetic Operations Module executes three standard GA operations, namely crossover, mutation and selection. The details of these operations are described in the sections below.

(1) Crossover

The GA implemented in the IGA scheme uses a random two-point crossover operation. Two cutting points are randomly selected in a pair of chromosomes, and the bit strings between these two cutting points are simply exchanged between the two chromosomes.

(2) Mutation

In the mutation operation performed in the current GA, a single chromosome within the population is chosen at random and a bit within this chromosome is then selected (again at random) and its value flipped. The probability of the mutation operation being performed is governed by the mutation rate parameter. To ensure the responsiveness of the IGA scheme to changes in the environment, the mutation rate parameter is assigned a dynamic rather than static characteristic, i.e. it is assigned a value of 0.5 initially, but is then

adjusted adaptively in accordance with changes in the system environment. The value of the mutation rate parameter is specified in accordance with Eq. (1), in which AvgFit denotes the average value of the fitness functions of all the chromosomes within the population.

$$\text{Mut\_Rate} = \left(\frac{1 - \text{AvgFit}}{2}\right) \quad (1)$$

(3) Selection

Having completed the crossover and mutation operations, the chromosomes are ranked in a descending order of fitness, and the top 90% of the chromosomes are selected as members of the new population pool. The remaining members of the population pool are then selected at random from the 10% of chromosomes remaining. In other words, the selection process is performed using a Rank-based Roulette Wheel Selection (RRWS) scheme [22]. This scheme ensures that better members of the population (i.e. solutions with a higher fitness) have a greater chance of being selected for reproduction. Moreover, by randomly selecting poorer members of the population for inclusion in the reproduction process, the diversity of the population pool is maintained. Significantly, the RRWS scheme is more robust that the conventional RWS method in converging to a best solution since it prevents good solutions discovered early on from dominating the population pool.

4.2. Fitness Function Calculation & Evaluation Module

In the IGA solution procedure, the relative quality of each of the chromosomes within the population is evaluated using the fitness function presented by Tan [3] .As shown in Eq. (2), this fitness function comprises two components, namely a sensitivity term (see Eq. (3)) and a specificity term (see Eq. (4)).

$$\text{Sensitivity} = \frac{tp}{tp + fn} \quad (2)$$

$$\text{Specificity} = \frac{tn}{tn + fp} \quad (3)$$

$$fitness = \left(\frac{tp}{tp + \alpha \cdot fn}\right) + \left(\frac{tn}{tn + \beta \cdot fp}\right) \quad (4)$$

$$0.2 \le \alpha \le 1,\ 1 \le \beta \le 20$$

Note that in these equations, *tp* denotes true positive, *fp* denotes false positive, *tn* denotes true negative, and *fn* denotes false negative (see Table 2). Furthermore, α and β are user-defined parameters which determine the rate of convergence of the solution procedure and are determined experimentally in accordance with the requirements of the particular application. In general, the value of the fitness function varies in the range 0~1, with a higher value indicating a better fit, i.e. a higher solution quality.

Table 2 summarizes the possible outcomes of a generic classification rule X⇒Y. As shown, four possibilities exist, namely:

1. True positive (*tp*): the actual class is Y and the predicted class is also Y.
2. False positive (*fp*): the actual class is Y, but the predicted class is not Y.
3. True negative (*tn*): the actual class is not Y and the predicted class is also not Y.
4. False negative (*fn*): the actual class is not Y, but the predicted class is Y.

As discussed earlier in relation to Table 1 in Section 3.2, the classification rules developed using the IGA scheme are expressed in the form of simple IF-THEN statements in order to enhance their meaningfulness to the mining practitioner. In the iterative GA-based procedure performed at each stage of the incremental rule inference process, the fitness function ensures that the best classification rules (i.e. those rules which are accurate and clear) survive, whilst those which are not are discarded. Having completed the mining procedure, overlapping set theory is applied to the final population of classification rules to determine the particular subset of these rules to present to the end user.

5. SIMULATION

The performance of IGA was evaluated by performing a series of classification rules mining simulations using two standard datasets. The simulations were performed on a 2.4 GHz Intel Pentium processor with 256 MB RAM and an 80-Gbyte HDD running under the Windows XP Professional operating system. The simulation program was written using Borland C ++ Builder 6.0 and the datasets were managed using Microsoft Office Access 2003.

5.1. Simulation Procedure

The simulations were performed using two different datasets downloaded from the University of California at Irvine - Machine Learning Repository (UCI) [23], namely Mushroom and Zoo. The details of these two datasets are summarized in Table 3.

In designing the simulations, an assumption was made that the data within these datasets were supplied to IGA at a constant rate. In addition, the rule mining process was assumed to involve a total of *n* time intervals, and thus IGA performed *n+1*

mining operations. In the initial set of simulations, conducted using the Mushroom dataset, the performance of IGA was benchmarked against that of DNWS [9] in terms of the classification accuracy and the computational load. In the simulations, the data within the dataset were divided into 8 data blocks of equal size to represent the data acquired by the two schemes in 8 sequential stages, respectively. In the IGA scheme, the population size was specified as 20 chromosomes, and parameters α and β in the fitness function (see Eq. (2)) were specified as 0.5 and 8, respectively. Finally, in the DNWS heuristic, parameters α, β and γ were specified as 5.0, 0.25 and 0.50, respectively.

In the second set of experiments, performed using the Zoo dataset, the performance of IGA was compared against that of a traditional GA-based classifier in which the classification rules were inferred on a one-shot basis having acquired all the training instances within the dataset. As shown in Table 3, Zoo contains 101 training instances each of which has 18 attributes and is associated with one of 7 different classes. Of the 18 attributes, 1 attribute indicates the name of the animal, two are numeric and the remainder are Boolean variables. In the IGA simulations, the dataset was partitioned into four blocks of equal size, the population size
% was specified as 20 chromosomes and the fitness function parameters were specified as α=1 and β=1, respectively. The simulations evaluated the performance of the two schemes in terms of the number of iterations required to obtain a convergent solution, the average fitness function, the average classification accuracy, the average support, and the similarity of the classification rules. The similarity measure was determined at each stage of the IGA solution procedure by comparing the best classification rules inferred by IGA at that stage with the classification rules obtained by the one-shot GA classifier based upon the entire contents of the dataset. The similarity measure was computed in accordance with the formulation:

Overall similarity ratio= identical ratio+similar ratio (5)

The first term on the right-hand side of Eq. (5) indicates the percentage of the classification rules generated by IGA which are identical to the classification rules generated by the non-incremental GA classifier. Meanwhile, the second term indicates the percentage of the classification rules generated by IGA which are similar to (i.e. a subset of) a classification rule generated by the non-incremental GA scheme. For example, consider the following rules generated by the IGA scheme and the non-incremental GA scheme, respectively:

IF hair='1' AND milk='1' Then type='1' (Rule 1)
IF hair='1' AND milk='1' AND backbone='1' Then type='1' (Rule 2)

In this example, Rule 1 is a subset of Rule 2 and is said to be similar to Rule 2.

5.2. Simulation Results

(1) Mushroom dataset

Table 4 compares the classification performance of IGA with that of DNWS [9] when applied to the Mushroom dataset The results show that the two schemes achieve an average classification accuracy of 99% and 100%, respectively. The value of 99% obtained by IGA is judged to be sufficiently close to 100% that the discrepancy (1%) can be attributed to statistical errors**.**

Figure 7 compares the number of training instances processed by IGA and the DNWS scheme, respectively, at each stage in the simulation procedure. It can be seen that the number of instances increases linearly over the course of the DNWS simulation, but remains constant in the IGA simulation since old instances are simply forgotten when they pass out of the window. In other words, the results demonstrate that IGA incurs a low and constant processing overhead.

(2) Zoo Dataset

Figure 8 illustrates the performance of the non-incremental GA when applied to the Zoo dataset. As shown, the GA requires a total of 171 generations to classify the dataset and yields a final average classification accuracy (AvgAccurate) of 100%, a final average fitness (Fit) of 0.949, and a final average support (AvgSup) of 38.7%. It can be seen that the three performance measures have extremely low values (i.e. very close to zero) for the first 120 iterations, but increase rapidly thereafter. Thus, it can be inferred that the non-incremental GA is relatively inefficient in recognizing the better chromosomes within the population pool, but rapidly improves the quality of these solutions once it has done so.

The simulations results obtained when applying IGA to the Zoo dataset are summarized in Table 5. Inspecting the first column in the table, it is evident that the number of iterations required by IGA at each step in the simulation procedure is significantly lower than the 171 iterations required by the non-incremental GA. The reason for this improvement is two-fold: (1) IGA only processes a sub-set of the Zoo dataset at each stage, whereas the non-incremental GA processes the entire dataset; and (2) the accuracy of the initialization process in IGA is at least 56% in Steps 1~3, and thus the

classification rules converge more rapidly. Furthermore, it can be seen that the initialization accuracy of IGA increases as the simulation process proceeds and therefore yields a continuous reduction in the number of iterations required at each stage. For example, in Step 1 of the simulation procedure, the initialization accuracy is 0% (i.e. the population pool is randomly generated), and thus a total of 65 iterations are required to obtain a convergent solution. However, in Step 1, the initialization accuracy improves to 56.3% since the population pool inherits some of the best solutions from the previous step, and thus the number of iterations required reduces to 55. In the final step of the solution procedure, the initialization accuracy has a value of 70% and a convergent solution is obtained after just 25 iterations. Significantly, even in Step 0 when IGA is applied to a randomly generated chromosome pool, its performance is still far better than that of the non-incremental GA in terms of the number of iterations required, i.e. the non-incremental GA requires 171 iterations, while IGA requires just 65; representing a reduction of around 62%.

From Table 5, it can be seen that the average accuracy of the classification results obtained using IGA at Steps 0, 1, 2 and 3 are 99.7%, 92.2%, 100% and 100%, respectively. Thus, the results show that IGA consistently achieves a high level of classification accuracy when applied sequentially to non-overlapping subsets of the Zoo dataset. It can also be seen that the average support values obtained in Steps 0 ~ 3 are all higher than 36%, which indicates that the classification rules obtained at each stage of the solution procedure are highly reliable.

Table 6 summarizes the similarly ratio data obtained in Steps 0 ~ 3 of the IGA solution procedure. Given an assumption that the classification rules obtained by the IGA scheme at each stage are equal in number and the data in the dataset are uniformly distributed, one would expect the *identical ratio* at each stage (i.e. the percentage of rules generated by IGA which are identical to those generated by the non-incremental GA) to have a value of 25%. However, the results show that the *identical ratio* actually has values of 10%, 30%, 45% and 20%, respectively, in Steps 0 ~ 3 of the solution procedure. In other words, the training instances processed in the second stage of the simulation are more representative of all the samples in the dataset than those processed in the remaining stages. This is borne out by an inspection of Table 5, which shows that the average accuracy (100%) and average support (40.4%) performance indicators attain their highest values in Step 2. Table 6 also shows that the *overall similarity ratio* between the classification rules obtained using IGA and those derived using the non-incremental GA is consistently higher than 60%, which confirms the reliability of the IGA approach.

The bottom row in Table 6 presents the similarity ratio data obtained by comparing all of the classification rules generated by IGA over the four steps of the simulation procedure with the rules derived by the non-incremental GA. As shown, the *identical ratio* has a value of 45%. This apparently low value is to be expected since in separating the original dataset into four discrete data blocks, some information available to the non-incremental GA is hidden from IGA, and thus IGA inevitably infers a different set of rules. The overall *similarity ratio*, however, is seen to be 100%, which suggests that given an identical set of training instances, IGA and the non-incremental GA will generate the same classification rules.

6. CONCLUSIONS

Many real-world datasets contain huge volumes of data and are compiled and maintained over a period of months if not years. Due to the typical life spans of such systems, the concepts responsible for generating their contents may well change over time, resulting in a phenomenon known as concept drift. Traditional classification techniques such as genetic algorithms (GAs) are ideally suited to the processing of datasets with stationary distributions, but yield significant errors when applied to datasets in which the contents are generated by a series of different concepts rather than one single concept. Accordingly, this study has presented an incremental genetic algorithm (IGA) for classification rules mining in datasets characterized by concept drift. In the proposed approach, a fixed-size window is slid over the training data at a constant rate, and each time the window is completely filled with new training data, a GA integrated with a local heuristic search method is applied to infer the best classification rules. The robustness of the classification rules to concept drift is ensured by utilizing a memory-based random immigrant mechanism, in which the initial population utilized by the GA when processing the training data within a window comprises a mix of the best solutions obtained from the training data in the previous window and an appropriate number of randomly generated chromosomes. IGA is applied sequentially to the training data for as long as an incremental learning function is required.

The performance of IGA has been benchmarked against that of DNWS [9] and a traditional non-incremental GA classifier using the Mushroom and Zoo datasets, respectively. The results obtained for the Mushroom dataset have shown that IGA has a virtually identical classification performance to the DNWS scheme, but incurs a significantly lower computational overhead due to its policy of forgetting old examples when they drop out of the window. Meanwhile, the results obtained for the Zoo dataset have shown that the number of iterations required by IGA to obtain a convergent solution reduces as the cumulative number of processed windows increases. For example, when processing the training instances in the first window, IGA requires 65 iterations to obtain convergence, whereas when processing the data within the fourth window, it requires just 25. By contrast, the non-incremental GA requires a total of 171 iterations to infer the best classification rules, i.e. the rules which yield a classification accuracy of 100%. The minimum classification accuracy of IGA is 92%, while the mean classification performance is 98%. Therefore, IGA is more computationally efficient than the non-incremental GA, but achieves a virtually identical classification performance. Finally, the average support values obtained at each stage of the IGA incremental mining process exceed 36% in every case, and thus the reliability of the classification rules is confirmed.

Overall, the experimental results confirm the feasibility and efficiency of IGA for classification rules mining in the presence of concept drift. In future studies, the current authors will investigate a number of potential enhancements to the IGA scheme. Firstly, in the current version of IGA, the initial chromosome population (i.e. that used during the very first stage of the solution procedure at $t=0$) is generated randomly. Therefore, future studies will aim to develop heuristic techniques for generating a fitter starting population based on the particular characteristics of the target dataset, the unique features of the data within the dataset, and so forth. In this way, it is anticipated that the number of iterations required to derive the best solutions will be reduced, and their quality improved. Secondly, the simulations in this study have considered only the Mushroom and Zoo datasets. Accordingly, in future studies, the feasibility and versatility of IGA will be more rigorously examined through its application to a wide range of datasets containing different types of data. Thirdly, in its present form, IGA features a single GA, which works alone to solve the classification rules mining problem. However, in a future study, an attempt will be made to implement IGA in a parallel environment, i.e. by using multiple collaborative agents (classifiers), which exchange information with one another, migrate best solutions amongst one another, and so forth, such that IGA can be applied to the solution of more complex problems in a rapid and efficient manner. Finally, in the present study, an assumption has been made that new training instances arrive at a constant rate, i.e. the concept drift has a stable characteristic. Thus, a future study will investigate the feasibility of adapting IGA to the classification rules mining of datasets characterized by unpredictable changes in concept.

7. REFRENCES

[1] E. Noda, A.A. Feritas, and H.S. Lopes, "Discovering interesting rules with a genetic algorithm. Evolutionary Computation", *Proceedings of the 1999 Congress on (CEC99)*, Vol. 2, 1999, pp. 1322-1329.

[2] B. Liu, H.A. Abbas, and B. McKay, "Classification Rule Discovery with Ant Colony Optimization", *IEEE/WIC International Conference*, 2003, pp. 83-88.

[3] K.C. Tan, A. Tay, T.H. Lee, and C.M. Heng, "Mining multiple comprehensible classification rules using genetic programming", *Evolutionary Computation, Proceedings of the 2002 Congress on*, Vol. 2, 2002, pp. 1302-1307.

[4] M. Aissiou and M. Guerti, "Genetic Algorithms Application for the Automatic Recognition of the Arabic Stop Sounds", *Journal of Applied Sciences Research*, Vol. 3, No.5, 2007, pp. 358-366.

[5] L. Araujo and J.J. Merelo, "A Genetic Algorithm for Dynamic Modelling and Prediction of Activity in Document Streams", *Genetic and Evolutionary Computation Conference, Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, London, England, United Kingdom, 2007, pp. 1896 - 1903.

[6] H. Wang, W. Fan, P.S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers", *Proceedings of 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Washington DC, 2003, pp. 226-235.

[7] M. Zhang, B. Kao, D. Cheung, and C.L. Yip, "Efficient algorithms for incremental updates of frequent sequences", *Proceedings of Pacific-Asia Conf. On Knowledge Discovery and Data Mining (PAKDD'02)*, 2002, pp. 186-197.

[8] L. Yang, D.H. Widyantoro, T. Ioerger, and J. Yen, "An entropy-based adaptive genetic algorithm for learning classification rules", *Evolutionary Computation, Proceedings of the 2001 Congress on*, vol. 2, 2001, pp. 790-796.

[9] R. Klinkenberg and I. Renz, "Adaptive Information Filtering: Learning in The Presence of Concept Drifts", *In Sahami, M., Craven, M., Joachims, T., McCallum, A. editors, Workshop Notes of the ICML-98 Workshop on Learning for Text Categorization*, Menlo Park, CA., AAAI Press, 1998, pp. 33–40.

[10] M. Maloof, "Incremental Rule Learning with Partial Instance Memory for Changing Concepts", *Proceedings of the International Joint Conference on Neural Networks*, Los Alamitos, CA: IEEE Press, 2003.

[11] P. Domingos and G. Hulten, "Mining high-speed data streams", *Proceedings of 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'00*, Boston, MA, 2000, pp. 71-80.

[12] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams", *Proceedings of 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'01*, San Francisco, CA, 2001, pp. 97-106.

[13] G. Widmer and M. Kubat, "Learning in The Presence of Concept Drift and Hidden Contexts", *Machine Learning*, vol. 23, no. 1, 1996, pp. 69-101.

[14] G.H. Xie, "An efficient approach for mining concept-drifting data stream", *Master Thesis of Information Education in National University of Tainan*, 2004.

[15] S. Yang, "Memory-Based Immigrants for Genetic Algorithms in Dynamic Environments", *Proceedings of the 2005 conference on Genetic and evolutionary computation, GECCO'05*, Washington, DC, USA, 2005, pp. 1115-1122.

[16] S.U. Guan and F. ZhuCollard, "An incremental approach to genetic-algorithms-based classification. Systems", *Man and Cybernetics, Part B, IEEE Transactions*, vol. 35, no. 2, 2005, pp. 227 – 239.

[17] J. Han and M. Kamber, "Data Mining Concepts and Techniques", *Morgan Kaufmann Publishers* (2007).

[18] F. Vavak and T.C. Fogarty, "A comparative study of steady state and generational genetic algorithms for use in nonstationary environments" *In T. C. Fogarty, editor, AISB Workshop on Evolutionary Computing, LNCS*, vol. 1143, Springer, 1996, pp. 297–304.

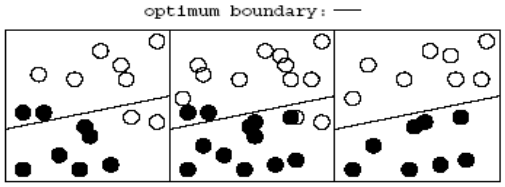
[19] J. Branke, "Memory enhanced evolutionary algorithms for changing optimization problems", *Proceedings of the 1999 Congress on Evolutionary Computation*, vol. 3, 1999, pp. 1875–1882.

[20] J. Branke, T. Kausler, C. Schmidth, and H. Schmeck, "A multi-population approach to dynamic optimization problems", *Proceedings of the Adaptive Computing in Design and Manufacturing*, 2000, pp. 299–308.

[21] G. Chen, X. Wu, and X. Zhu, "Mining sequential patterns across data streams", *Computer Science Technical Report CS-05-04*, University of Vermont, 2005.

[22] A.J. Omar, R. Lakishmi, and C.R. Rao, "Improved Selection Operator for GA", *Journal of Theoretical and Applied Information Technology*, 2008, pp.269-277.

[23] http://www.ics.uci.edu/~mlearn/MLRepository.html



S_0 arrived during $[t_0, t_1]$ | S_1 arrived during $[t_1, t_2]$ | S_2 arrived during $[t_2, t_3]$

Figure. 1: Concept drifting phenomenon [6].

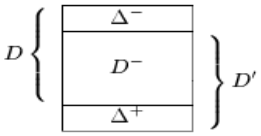


Figure 2: Definitions of D, D', Δ^-, D^- and Δ^+ [7].

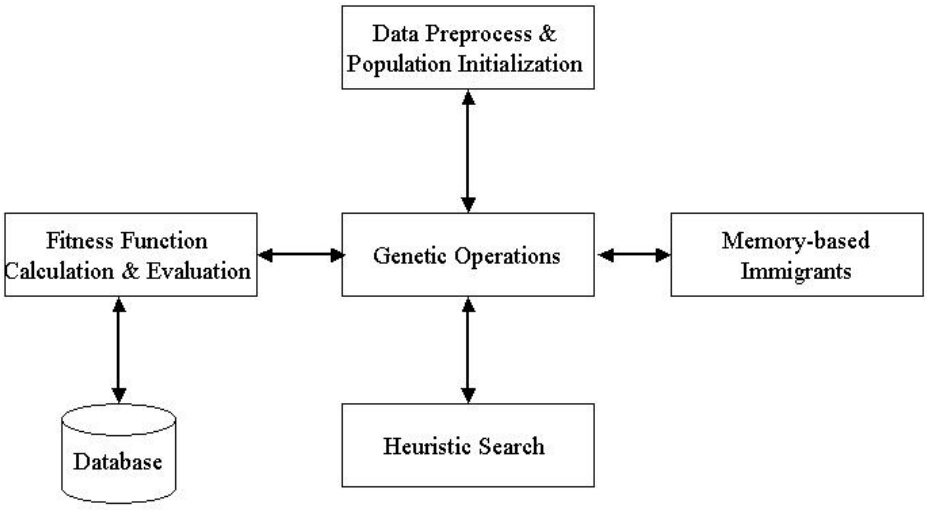


Figure 3. System architecture.

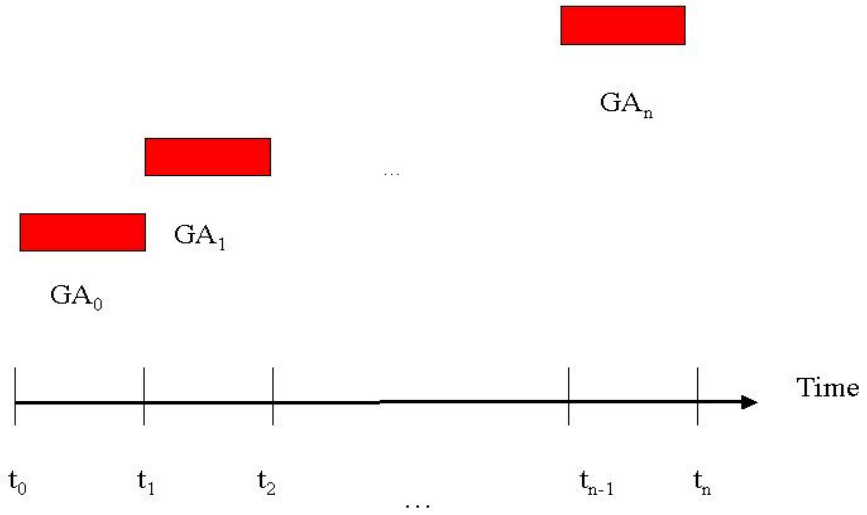


Figure 4: Time-series based GA learning process.

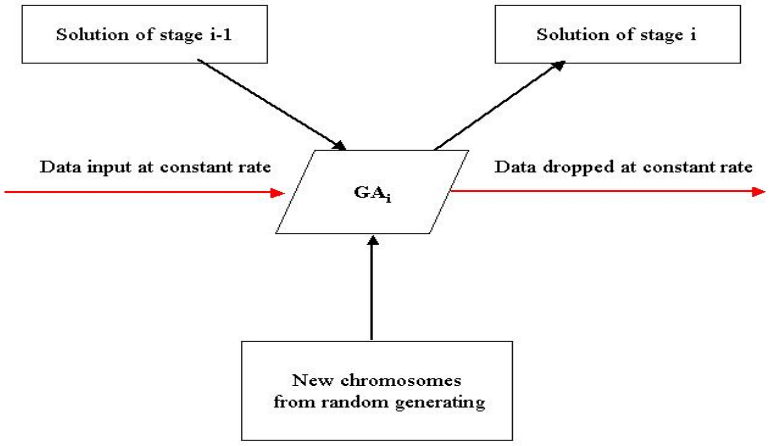


Figure 5: i-th stage in IGA scheme.

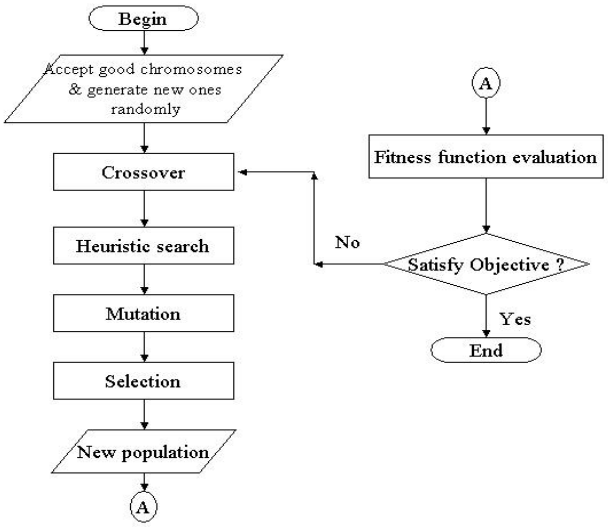


Figure 6: Iterative GA solution procedure.

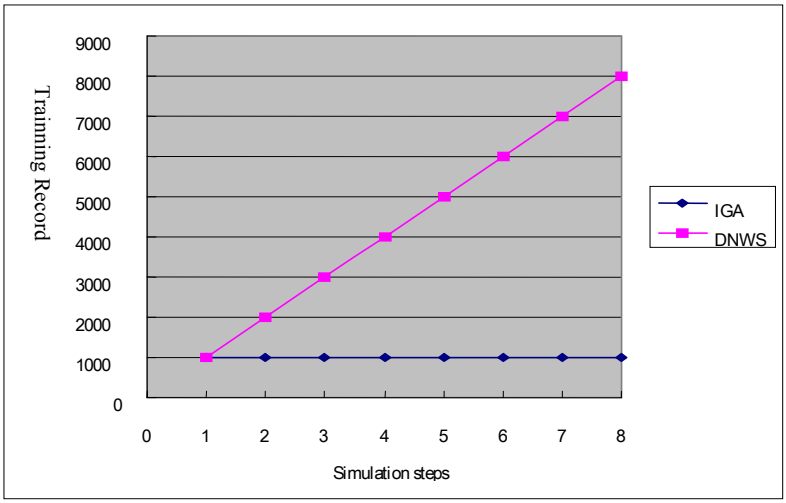


Figure 7: Number of training instances processed by IGA and DNWS at each stage o f simulation procedure.

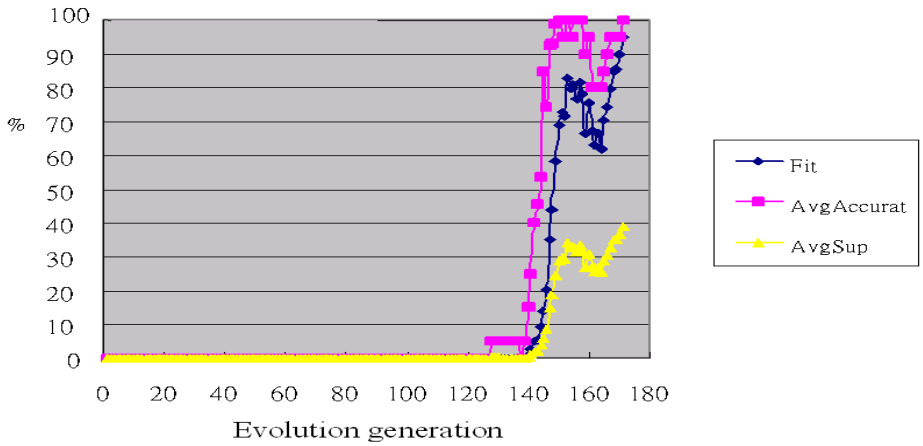


Figure 8: Simulation results obtained when classifying Zoo using non-incremental GA

Table 1: Chromosome Coding.

A1	A2	A3	Target
01000	100	000100	001

Table 2: Prediction cases of classification rules.

		Predicted	
		Yes	N
Actual Class	Yes	*tp*	*fp*
	No	*fn*	*tn*

Table 3: Summary of test data set.

Data Set	Records	No. Of Attributes	Values Of Target Attribute
Mushroom	8124	22	2
Zoo	101	18	7

Table 4: Average classification performance of IGA and DNWS for Mushroom dataset.

Simulation	IGA	DNWS
1	100%	100%
2	95%	100%
3	100%	100%
4	100%	100%
5	95%	100%
6	100%	100%
7	100%	100%
8	100%	100%
Average	99%	100%

Table 5: Simulation results obtained when classifying Zoo using IGA.

Simulation Step	Evolution Generation	Fitness Function	Accuracy of Initialization	Accuracy	Support
0	65	0.958	0%	99.7%	37.6 %
1	55	0.881	56.3%	92.2%	37.7 %
2	34	1	65 %	100 %	40.4 %
3	25	0.902	70 %	100 %	36.4 %

Table 6: Similarity ratio data obtained when classifying Zoo dataset with IGA

Simulation	Identical Ratio	Similar	Overall Similarity
0	10%	65%	75%
1	30%	50%	80%
2	45%	25%	65%
3	20%	40%	60%
Total	45%	55%	100%