Journal of Electrical and Computer Engineering Innovations (JECEI)

Journal homepage: http://www.jecei.sru.ac.ir

SRTTU JECEI

Research paper

Presenting New Methods for Improving the Ant-Miner Algorithm

H. Nosrati Nahook, *S. Tabatabaei*[*]

Department of Computer Engineering, Faculty of Engineering, Higher Education Complex of Saravan, Saravan, Iran.

Article Info

Article History:
Received 01 January 2021
Reviewed 28 February 2021
Revised 19 March 2021
Accepted 29 April 2021

Keywords:
Ant-Miner
Qualitative contrast intensifier
Logical negation operator
Data analysis
Categorizing of laws

[*]Corresponding Author's Email Address:
hsn.nosrati@gmail.com

Abstract

Background and Objectives: The Ant-Miner algorithm works based on Ant Colony Optimization as a tool for data analysis, and is used to explore classified laws from a set of data. In the current study, two new methods have been proposed for the purpose of optimizing this algorithm. The first method adopted logical negation operation on the records of the produced laws, while the second employed a new pheromone update strategy called "Generalized exacerbation of quality conflict". The two proposed methods were executed in Visual studio C#.Net, and 8 public datasets were applied in the test. Each one of these datasets was executed 10 times both in an independent way and combined with others, and the average results were recorded.
Methods: In this study, we have proposed two approaches for the earlier method. Using the first method in the construction of rule records, idioms that include the rules can be made in the form of <Not attribute = value>. Compared to the idioms of the early algorithm, these idioms are more compatible while constructing rules with high coverage. The advantage of this generalization is the reduction of the produced rules, which results in greater understandability of the output. During the process of pheromone update in the ordinary ACO algorithms, the amount of the sprayed pheromone is a function of the quality of rules. The objective of the second method is to strengthen the conflict between not-found, weak, good, and superior solutions. This method is a new strategy of pheromone update where ants with high-quality solutions are motivated through increasing the amount of pheromone sprayed on the trail that they have found; conversely, the ants that find weaker solutions are punished through eliminating pheromone from their trails.
Results: The optimization of the initial algorithm using the two proposed methods produces a smaller number of rules, but increases the number of construction diagrams and prevents the production of low-quality rules.
Conclusion: The results of tests performed on the dataset indicated the enhancement of algorithm efficiency in idioms of fewer tests, increased prediction accuracy of laws, and improved comprehensibility of the produced laws using the proposed methods.

Introduction

Swarm intelligence is a branch of soft calculation in which intelligent biological collective behaviors are applied [1]. Swarm intelligence can be seen among many insects, including honeybee and ants, and is an indication of a principled structure and an integrated behavior without the existence of any central monitoring on the components of the set. The Ant Colony Optimization (ACO) is among the most famous swarm intelligence algorithms. Dorigo et al. proposed the ACO as a metaheuristic method for solving optimization problems. This algorithm works based on the behavior of ant colonies that, despite having simple members, produce a collective behavior enabling them to perform complex tasks that is impossible to perform individually.

One of the major applications of swarm intelligence algorithms is data analysis, which is used to extract useful patterns from datasets. One of the techniques used in data analysis is classification which assigns a sample to a class or group according to its predicted features. The output of categorizing is a model that makes it possible for the future samples to be classified. There are several techniques for categorizing such as decision trees according to C4.5, logical and linear regression, non – linear optimization (Cuckoo optimization algorithm) [2], [3] Artificial Neural Networks and Ant-Miner.

Regarding the optimization of the Ant Colony algorithm, it can be said that sets of social insects, such as ant colonies, are distributed systems that, despite their simplicity, produce a collective behavior that enables them to perform anything that cannot be performed alone [4] .The highly coordinated and self-regulated structure of ant colonies can be applied in constructing a factor-based artificial system to solve complex calculation problems. Ants coordinate their actions through spraying pheromones in an environment.

Pheromone is a chemical that ants leave behind while moving around, and other ants can smell it. According to the amount of pheromone left by other ants, an ant can select its path. The pheromone that has been left in an environment establishes a pheromone trail, enabling ants to find rich sources of food discovered by other ants [5].

If some obstacles are placed in the path of ants, they will face two options. Since no pheromone exists on this path, the first ant to reach the obstacle randomly selects a path, either the top or the bottom, to overcome it. Then, the next ant selects its path randomly according to the pheromone left by the previous ant, and the remaining ants follow the same procedure.

Since the path is shorter, the time taken to travel on it is shorter too; consequently, more ants will select this path compared to the other one. Because of the existence of more pheromone in this path, most ants, though not all, will converge to the top path.

Importantly, selection of the upper path is more probable but not certain. In other words, if the upper path has more pheromone compared to the lower path, it cannot be concluded that all ants will select the upper path; it can only be stated that the ants will move from the shorter path with higher probability. If this was not the case, assume that the first ant would randomly select the longer path and leave a pheromone trail behind. Then, the remaining ants would follow it, and the shorter path would never be discovered. Therefore, randomness and probability play major roles in the ACO.

Another consideration is the evaporation of the pheromone left behind. If the obstacle is removed and the pheromone is not evaporated, the ants will follow the previous trail. However, this is not the case for actual conditions, and the evaporation of pheromone and probability are two factors that make it possible for the ants to discover the shorter path [5], [6].

In the case of artificial ants, we can say that In the case of artificial ants, we can say that through spraying pheromone and applying probability rules based on local information, ants can indirectly find the shortest path that exists between two points. The idea behind the system of artificial ants is that the solution space for each optimization problem is shown as the nodes of a graph that indicate the various modes of the solution. The artificial ants simulate pheromone spray through changing the pheromone related to the modes of a solution that is discovered. According to the indirect communications model, they only have local access to the pheromone variables [7], [8].

ACO is a system of artificial ants that has been formed according to the ideas explained through real ants. Colonies of both real and artificial ants are formed from simple members which attempt to attain certain goals using collective behavior. Considering real ants, this goal is to reach a food source through the shortest path possible. On the other hand, finding a suitable solution for an optimization problem is the goal of artificial ants. A single ant (either artificial or real) can find a solution for its own problems, but suitable solutions are discovered only through cooperation [9], [10].

Artificial ants live inside a virtual world, such as graphs made up of nodes that indicate the search space of a particular problem. The equivalent of pheromone in an artificial system is a numerical variable related to each mode of the search graph. This pheromone is sprayed by the ants during their quest to establish a solution in the graph of solution modes and is influenced by that graph. The pheromone trail is equal to a sequel of pheromone values that are related to problem modes. The artificial ants of the ACO are used to simulate the behavior of real ants [11], [12].

In the ACO algorithm, the behavior of natural ants is simulated on the artificial ones. The ant-based algorithms are successful examples of collective intelligence systems and include instances ranging from the traditional Traveling Salesman Problem (TSP) to routing in telecommunication systems. The ACO algorithm is a metaheuristic algorithm and a class of ant-based algorithms. The first algorithm that was introduced according to the behavior of ants was the Ant System [13]. This algorithm was the result of research studies conducted by Dorigo et al. in Milan Polytechnic University on intelligent calculation methods for the purpose of solving NP-hard problems.

The Ant System algorithm was first applied in solving the TSP problem. This algorithm, despite being very successful in solving TSP problems on small samples (below 30 cities), was not quite useful for solving larger problems due to time constraints. For this purpose, some changes were made on this algorithm to increase its efficiency, and the MAX-MIN and Ant Colony System are but a few examples [14].

Dorigo et al. formulated the above innovative ACO model that uses pheromones in solving Combinational Optimization Problems and has been used since then in solving many optimization problems. The COP model can be defined in this manner: P = (S, $\boldsymbol{\Omega}$, f)

Where **S** is the search space defined on a finite set of discrete decision variables, $\boldsymbol{\Omega}$ is the set of limitations among variables, and f is the objective function that is defined as $f: \boldsymbol{S} \longrightarrow \mathbb{R}_0^+$ and has to be optimized (maximized or minimized).

The search algorithm has to pursue a suitable solution in the search space *S* so as to optimize the objective function *f* according to the set of limitations Ω.

The search space S is a set of discrete variables X_i and values $V_i^j = \{V_i^1, V_i^2, \dots, V_i^{|D_i|}\}$, where i=1,2,3, . . .,n. Values are assigned to a variable in such a way that the value of V_i^j is related to X_i and is indicated through $X_i = V_i^j$. Each variable whose value has been assigned is called a solution component, and is shown as C_{ij}. The set of all probable solution components is indicated with C. $s \in S$ is a possible solution where all variables have taken a value from their domain in a way that all limitations in the Ω set are fulfilled. The $s^* \in S$ solution is called a global optimal solution if and only if $f(s^*) \leq f(s)\ \forall\, s \in \boldsymbol{S}$.

The set of all globally optimal solutions is indicated with $s^* \subseteq S$. To solve a COP problem, at least one $s^* \in S$ is essential. A solution component C_{ij} is assigned for each parameter in the pheromone trail T_{ij}. The set of all parameters in the pheromone trail is indicated with T. the value assigned for the pheromone trail related to the solution component C_{ij} at time *t* is represented as $\tau_{ij}(t)$, the value of pheromone is used through the ACO algorithm during the search process and is updated [15].

In ACO, the explained model is indicated through a graph called the "Construction graph" and is surveyed by artificial ants to obtain a solution for a particular problem. The construction graph $G_c = (V, E)$ is a graph whose vertices are connected to each other and is comprised of the set of vertices V and the set of edges E. The set of components C can be linked with both V and E sets in a graph G_C. An artificial ant creates a solution in an incremental way while moving from one vertex to another through the edges of a graph. In addition, it sprays a particular amount of pheromone on the components of the graph – either vertices or edges. The amount of sprayed pheromone $\Delta\tau$ depends on the quality of the obtained solution. The following ants are influenced through the pheromone trail and use it as a guide in making proper decisions in the search graph.

Each ant creates a solution using the components of the selected solution incrementally through the application of pheromone. Local search can be applied to improve the quality of solutions. Then, the pheromone trail of the ant is updated while moving forward. The amount of sprayed pheromone depends on the evaluation function that has been used to determine the quality of the created solution. These stages are reiterated until the final conditions are met. In the following section, the major components of the algorithm are going to be introduced [16], [17], [18].

Each ant $\mathrm{ant}_l \in \{\mathrm{ant}_1, \mathrm{ant}_2, \dots, \mathrm{ant}_m\}$, where $l = 1, 2, 3, \dots, m$, creates a solution from the members of the set of solution components $\boldsymbol{C} = \{c_{ij}\}$ in the construction graph $\boldsymbol{G_c}$ i indicates the index of the solution components, while j is the value index of the range of this variable. Each ant starts with a null solution $s^l = \emptyset$. In each stage of creating the solution, one c_{ij} – obtained from the set of possible neighbors) $\boldsymbol{N}(s^l) \subseteq \boldsymbol{C}$ (– is added to the solution being created. The process of creating a solution can be considered a path in the construction graph $\boldsymbol{G_c}$ in a way that the set of limitations $\boldsymbol{\Omega}$ defined possible neighbors in each stage – according to the present state of the solution. Decision component c_{ij} in each stage is selected randomly according to (1) [18]:

$$p(c_{ij}) = \frac{\tau_{ij}^{\alpha}(t).\, \eta_{ij}^{\beta}}{\sum \tau_{ij}^{\alpha}(t).\, \eta_{ij}^{\beta}}, \qquad \forall\, c_{ij} \in \mathbf{N}(s^l) \tag{1}$$

where $\tau_{ij}(t)$ is the amount of pheromone dependent on the c_{ij} component in time t, η_{ij} is the problem-dependent heuristic value that is related to the c_{ij} component, and α and β are positive parameters. The values which determine the relative importance of pheromones are heuristic depending on the information $(\cdot \leq \alpha, \beta \leq 1)$.

Local search is a selective solution that can be applied after the creation of a solution with the aim of optimizing it. It can be used in the form of a problem-specific procedure before the pheromone is updated. It can also enhance the quality of the solution and increase the overall output of the algorithm. The search process, depending on the regions where the search is conducted, can be costly. Local search can be used after the creation of a solution through an ant or – in an iteration-based manner – on the best solution created through a set of ants in each iteration [19]. After a solution is created, the pheromone is updated on the construction graph to act as a guide for the following ants in making proper decisions for generating their own solutions. Pheromone update is a two-stage process.

- Pheromone strengthening: This is performed through increasing the amount of pheromone related to decision components $c_{ij} \in s^l$ according to the quality of the created solution. The pheromone is strengthened according to (2):

$$\tau_{ij}(t+1) = \tau_{ij}(t) + f(s^l), \forall\, c_{ij} \in s^l \tag{2}$$

where $f: \boldsymbol{S} \to \mathbb{R}^+$ is a fitness function evaluating the quality of the solution s^l.

- Pheromone evaporation: It is performed through reducing the amount of pheromone related to all instances of $c_{ij} \in \boldsymbol{C}$ in the construction graph. Thus, the amount of pheromone in the unfavorable components (those that are not selected regularly) is reduced and replaced with the other components of undiscovered regions. This prevents premature convergence.

Pheromone evaporation is performed according to (3) [19]:

$$\tau_{ij}(t+1) = (1-\rho).\,\tau_{ij}(t) + \rho, \forall\, c_{ij} \in \mathbf{C} \tag{3}$$

where ρ is the parameter of the evaporation factor, ranging in the domain of $\rho \in [0,1]$.

Related Studies

The Ant-Miner algorithm was first introduced in 2002 by Parpinelli et al [20]. It is an ACO-based algorithm and extracts an ordered list of IF-THEN classificatory laws in the following manner: IF <Term-1> AND <Term-2> AND . . . <Term-n> THEN <Class> where each term is in the form of <attribute = value>, and the output is a predicted class. As mentioned before, the Ant-Miner algorithm has been introduced as an ACO-based algorithm for the purpose of exploring the classificatory laws from labeled samples. It has proven as a powerful rival against other famous taxonomy algorithms. The experimental results of this algorithm showed that it has acceptable performance, and provides simpler laws. Many changes have been made to this algorithm to enhance its efficiency. In the following, a summary of such changes has been presented.

In 2002, Liu et al. [21] proposed the Ant-Miner2 algorithm that used a swarm-based exploratory function to explore laws. To explain, the ACO algorithm does not require exact exploratory information since pheromone has to counteract possible errors in the exploratory value. In other words, a simple exploratory value can act as a more complex value. Although the AntMiner2 uses a simple exploratory function, according to the concentration of the majority class with lower calculation costs, the obtained results are similar to the results of the early Ant-Miner (using entry for the measurement of the exploratory value) [21].

In 2003, Liu, Abbas, and MacKay proposed the AntMiner3 algorithm in which the method of updating pheromone and the process of more transitions were changed. In the new method of updating pheromone, the strengthening and evaporation of pheromone is conducted using an equation, while in the early Ant-Miner these processes were performed using two separate equations. AntMiner3 needs more ants for convergence and obtaining a solution, but explores more laws compared to the early Ant-Miner. In addition, the average accuracy of the set of explored laws in this method is higher than that in the early Ant-Miner [22].

In 2005, Chan and Freitas proposed a new process of law trimming called the “hybrid trimming of laws” in [24]. This process is a combined form of the early Ant-Miner trimming which functions according to information usefulness. This hybrid feature enables the method to benefit from the efficiency of the early Ant-Miner (in idioms of maximization and prediction accuracy) and the speed of trimming according to information usefulness.

The latter technique is quite fast since it does not require any survey on the training set. If the number of idioms present in the record of law is more than *r*, the number of idioms will be reduced to *r*. The reduction operation is conducted in this manner: for each term that is present in the record of each law, the law trimmer estimates the possibility of selecting that term. This possibility is according to the predetermined value of usefulness of that information relative to the attribute of that class. Then, the information trimmer selects *r* where each term is selected according to the usefulness of information that it provides. After the selection of *r* idioms, the reduced law is transferred to the early Ant-Miner trimmer. Experimental results show that the time taken to calculate this method is significantly lower than that of the early Ant-Miner, while no reduction has been observed with regards to the accuracy of the laws. In addition, the produced laws are shorter, a quality that increases understandability of the laws.

In 2006, Chan and Freitas proposed a newer version of Ant-Miner called Multi-Label Ant-Miner (MuLAM) [24] where, like the early Ant-Miner, each ant does not just produce one piece of law, but a set of laws. Each law produces at last one piece of law and at most equal to the number of attributes in a class. These laws must be different for each predicted class. An ant produces a piece of law only if it can accurately predict all attributes of that class . After an ant has finished making the records of its laws, predicting the value of the class begins. However, before the ant starts the prediction algorithm for the current law, it produces the law with an empty set. This law has all the class attributes (with empty values) that have been predicted. For each attribute of the class, the algorithm decides (according to several pre-trimming criteria) whether or not to add a particular class attribute to the law result as a prediction.

The Ant-Miner algorithm

In general, Ant-Miner is an inductive rule-based algorithm that uses the collective behavior of ACO. In ANT-Miner, the problem is making a stratification model, and a set of rules for stratification is proposed as the solution. Thus, each ant within the set attempts to make a rule to be applied in the set of stratification rules. The rules produced by Ant-Miner are as follows: IF <Conditions> THEN <Class> where <Conditions> or the record of rules includes a logical combination of predictive attributes in the following manner: term1 AND term2 AND . . .

Each triad term is in the form of <attribute, operator, value> where the value is related to the domain of attribute, and the operator is always "=" [25]. The early version of Ant-Miner only works on discrete attributes, and the continuous attributes (actual values) are converted into discrete ones in a preprocessing stage. However, the continuous attributes can be used through making some modifications in the algorithm. The <Class> section (or the consequences of rules) includes the predicted class of cases whose predictive attributes fulfill the <Conditions> section of the rules [26].

Ant-Miner explores a list of classified rules. In each attempt, the ant tries to explore a rule through the probable selection of idioms according to the heuristic function and the amount of pheromone in that term. Then, the ant updates the amount of pheromone in that selected path to guide the following ants in selecting their own paths. The best rule made by the ants is added to the set of explored rules. This algorithm is reiterated until the explored rules include an acceptable part of the dataset [26]. Table 1 illustrates the overall representation of the early Ant-Miner pseudo code.

In 2007, Martens et al. [27] proposed the Ant-Miner+ algorithm. This is one of the most important versions of Ant-Miner since it has excelled significantly relative to early Ant-Miner algorithm due to the changes made on it. In this version, the MAX-MIN (MMAS) system was used in the Ant-Miner algorithm [28]. Also a minimum τ_{min} and a maximum value τ_{max} were determined for the pheromone. Here, the value of pheromone must be higher than τ_{max} and must not fall below τ_{min}. In addition, the selection of classes in the Ant Miner+ is performed before the laws are produced by the ants. Therefore, an extra set of vertices are added to the beginning of the construction graph. This is similar to a situation where classes are considered the same as other variables, and their exploratory value and pheromones are calculated like others. If the classes selected by several ants differ, ants can produce laws simultaneously in one iteration. However, when each ant is affected by the pheromones of other ants – though related to other classes, the quality of the produced rules will be influenced negatively.

Table 1: The Ant-Miner algorithm

```
Begin Ant – Miner
$training\_set \leftarrow all\_training\_cases$;
$discovered\_rule\_set \leftarrow \emptyset$;
While $|training\_set| > \max\_uncovered\_cases$ do
    $InitializePheromoneAmounts()$;
        $CalculateHeuristicValues()$;
        $R_{best} \leftarrow \emptyset$;
    $i \leftarrow \emptyset$;
    Repeat
            $ConstructRuleAntecedent(ant_i)$;
            $ComputeRuleClass(ant_i)$;
            $R_{current} \leftarrow PruneRule(ant_i)$;
        $Q_{current} \leftarrow CalculateRuleQuality(R_{current})$;
            $UpdatePheromone(R_{current})$;
      If $Q_{current} > Q_{best}$ Then
                $R_{best} \leftarrow R_{current}$;
      End if
      $i \leftarrow i + 1$;
    Until $i = \max\_trials$ OR $Convergence()$
    $discovered\_rule\_set \leftarrow discovered\_rule\_set + R_{best}$;
    $training\_set \leftarrow training\_set - Cases(R_{best})$;
End While
End Ant – Miner
```

By increasing the vertices of the construction graph, it reduces the quality of production rules and increases the total number of production rules by discovering rules with high coverage, which leads to a decrease in the classification accuracy of production rules.

In addition, the Ant Miner+ can be executed on continuous attributes, while they have to be converted to discrete ones during a pre-processing stage to use such attributes in the early version of Ant-Miner.

In addition, in the ordinary ACO formula, the exploratory value (η) and pheromone (τ) components rise to the power of α and β, respectively. These powers are used to make varying emphases on each component. In the early Ant-Miner, the values of α and β equal 1. On the other hand, in the Ant Miner+, ants can select those values themselves. The values of weighted parameters range from 1 to 3 [28].

A. The Construction Graph

In the ACO, the solution space of a problem is indicated with a graph through which an ant produces a solution. Basically, a solution is a path that an ant produces in a test from its hill to a food source. Any decisions (nodes) taken by the ants along this path are considered a solution component. In the Ant-Miner, because the solution produced for the stratification problem is a rule that includes a set of idioms, the idioms themselves are regarded as the components of the solution. Thus, the construction graph has to include all idioms that can be used in generating a rule (solution). In the Ant-Miner, the nodes in the construction graph indicate the values of attributes present within the dataset. The set of nodes (N) in the construction graph are as follows (4):

$$N = \bigcup_{i=1}^{n} v_{ij} \ , \qquad j \in \{1, 2, \dots, l\} \tag{4}$$

where i indicates the ith attribute, n is the number of nodes, v_{ij} s the jth value related to the ith attribute, and l is the number of values related to the ith attribute.

Therefore, each node that has been selected in a term is in the form of $< A_i = V_{ij} >$. The set of idioms that an ant selects along its path can be represented in the form of the following rule:

IF <Term-1> AND <Term-2> AND . . . <Term-n> THEN <Class>

Each node in the construction graph has to have a particular amount of pheromone, and the initial values of pheromone for each term is set at the beginning of each iteration (5):

$$\tau_{ij}(t=0) = \frac{1}{\sum_{r=1}^{a} b_r} \tag{5}$$

where a is the total number of attributes, and b_r is the number of values in the range of attribute r. the construction graph does not include the values of class attributes, and only contains the idioms that take part in the construction of the rule. After the rule records were made, the consequence of a rule (i.e., class) is selected based on the value of a class with the highest frequency of occurrence in the samples conforming to the records of the law. Each node has a Boolean attribute that determines accessibility to the node. A node is inaccessible when all the values of its attributes have been covered by discovered rules. This Boolean attribute provides ants with the opportunity to consider or ignore a node while making decisions. Each node has a heuristic value that indicates the local quality of the selected term. This heuristic value influences the possibility of the selection of a particular term by the ants. This value is updated when rules are discovered and the size of the training set is reduced. The amount of pheromone is updated after each experiment; thus, the selection of idioms for other ants is influenced in future experiments [29], [30].

In the following sections, details related to this algorithm will be discussed.

B. The Construction of a Rule

A rule is created incrementally through the addition of a term to the previous ones. A $term_{ij}$ is in the form of $A_i = V_{ij}$ where A_i is the ith attribute and V_{ij} is the jth value from the A_i domain. The probability for the selection of $term_{ij}$ by an ant is calculated using (6):

$$P_{ij} = \frac{\eta_{ij} . \tau_{ij}(t)}{\sum_{r=1}^{a} \sum_{s=1}^{b_r} (\eta_{rs} . \tau_{rs}(t))} \tag{6}$$

where η_{ij} is the problem-dependent heuristic value for $term_{ij}$, $\tau_{ij}(t)$ is the current pheromone, *a* is the overall number of attributes, and b_r is the number of values in the domain of attribute *r*.

The probability of a term to be selected relies on two factors: the first is η_{ij} which indicates the value of the problem-dependent heuristic function; in addition, the second factor is τ_{ij} or the amount of pheromone of $term_{ij}$. The first factor measures the prediction power of $term_{ij}$; in other words, the greater this value is, the more qualified $term_{ij}$ will be, as selected with higher probability. The second factor τ_{ij} determines the amount of pheromone in $term_{ij}$ and is completely dependent upon the path that has been taken by the previous ants in making rules [30].

The amount of pheromone in the construction graph is a method to establish an indirect relationship between ants within a colony; in addition, it indicates the experience of previous ants in creating their solution and can be used as suggestions for the following ants to find better solutions. First, the amount of pheromone is the same in all idioms. Once an ant finishes its path, this amount is updated in idioms that have been selected by the ant. The amount of pheromone sprayed on a path depends on the quality of the constructed rule. The higher the quality of this rule is, the more the amount of pheromone will be, which is sprayed on the idioms of rule within the construction graph. After a while when several ants have generated their rules, the best path is more likely to be selected by the following ants. The selection and addition of a term to a rule under construction are faced with some limitations, which have been discussed below.

- If the rule under construction includes $term_{ij}$, the selection of $term_{ik}$ is impossible (i.e., two values of the same domain of attributes A_i cannot co-occur in one rule).
- If the number of the covered samples is smaller than the threshold value set via the min_cases_per_rule parameter, $term_{ij}$ cannot be added to a rule under construction.

During the construction process, the ant constructs a rule regardless of what class it belongs to. Determining a class for the rule occurs later. After the rule is constructed, the system chooses a class that can maximize the quality of the rule. This occurs through determining the number of samples this rule covers in each class [30].

C. The Heuristic Function

Each node of the construction graph has a current heuristic value that indicates the local quality of that node. This value is calculated using a problem-dependent heuristic function. The heuristic value is calculated through estimating the quality of a term according to its capability in the enhancement of the prediction accuracy of the rule. This heuristic function works based on the Information Theory presented in [31]. The value of the heuristic function is calculated by measuring the entropy (or the amount of obtained information) related to a particular term. For each term $term_{ij}$ in the form of $< A_i = V_{ij} >$, the entropy is calculated using (7):

$$\mathrm{Entropy}(\mathrm{T_{ij}}) = -\sum_{w=1}^{k}\left(\frac{\mathrm{Freq\ T_{ij}^{w}}}{|\mathrm{T_{ij}}|}\right) \times \log_2\left(\frac{\mathrm{Freq\ T_{ij}^{w}}}{|\mathrm{T_{ij}}|}\right) \quad (7)$$

where k is the number of classes, $|T_{ij}|$is the overall number of samples where attribute A_i has the value V_{ij}, and $Freq\ T_{ij}^{w}$ is the number of attributes where attribute A_i has the value V_{ij} belonging to the class *w*.

The larger amount of $Entropy(T_{ij})$ means that the A_i value has been distributed more homogeneously V_{ij}= among the classes, and the predictability power of $term_{ij}$ is lower. Idioms that are to be added to a rule under construction must have high prediction power. Thus, in the Ant-Miner, the greater the value of $Entropy(T_{ij})$, the less likely it will be to select $term_{ij}$ by the ants. The heuristic function of the information theory is normalized using (8):

$$\eta_{\mathrm{ij}} = \frac{\log_2(\mathrm{k}) - \mathrm{Entropy}(\mathrm{T_{ij}})}{\sum_{r=1}^{a}\sum_{s=1}^{b_r}\log_2(\mathrm{k}) - \mathrm{Entropy}(\mathrm{T_{rs}})} \quad (8)$$

where *a* is the total number of attributes, b_r is the number of values present in the domain of the *r*th attribute, and *k* indicates the number of classes. Regardless of the content of the rule under construction, $Entropy(T_{ij})$ for $term_{ij}$ is always the same. $Entropy(T_{ij})$ for each $term_{ij}$ is calculated in a preprocessing stage and before each external iteration in the Ant-Miner algorithm [31].

If the A_i attribute is not present within the training set, then $|\mathrm{T_{ij}}| = 0$. In such cases, $\mathrm{Entropy}(\mathrm{T_{ij}}) = \log_2(\mathrm{k})$ occurs, which shows the highest value for entropy. This corresponds to a situation where $term_{ij}$ is assigned with the lowest prediction power. If all the samples in partition T_{ij} belong to a similar class, then $\mathrm{Entropy}(\mathrm{T_{ij}}) = 0$. It is similar to a situation where the highest prediction value has been assigned to $term_{ij}$. Notably, the value of $Entropy(T_{ij})$ falls in the range of $0 \le \mathrm{Entropy}(\mathrm{T_{ij}}) \le \log_2(\mathrm{k})$.

D. Rule Pruning

The main objective of rule pruning is to eliminate extra idioms previously added to the rule. This process potentially increases the prediction power of a rule through extending its coverage without compromising its reliability. The simpler rules that are the result of the pruning stage are shorter and more general, and consequently easier to understand. This is another reason for pruning.

After the construction of a rule by each ant, the pruning process starts. The search strategy used in the pruning process of Ant-Miner is similar to the one proposed by Quinlan [30], [31] though the criterion of rule quality is quite dissimilar in these two studies. The main idea is to eliminate one term in each iteration, and this process continues until the quality of a rule is enhances satisfactorily. The first iteration starts with the complete rule; then, one term is eliminated in each iteration in a trial-and-error fashion, and the quality of the resulting rule is calculated using the quality function. In this stage, the class of the rule may change since the number of samples covered by the pruned rule may include another class. In each iteration, the term that results in the highest enhancement in the quality of the rule is eliminated. This process continues until there is only one term in the rule, or the elimination of no term can enhance the quality of the rule [31]. A process of pruning rules has been proposed by Chan and Freitas, which enhances the quality of the produced rules.

E. Pheromone Update

Each node in the construction graph indicates a term that has to be selected by an ant for rule construction. Each one of these nodes contains some pheromone. The amount of pheromone varies during the trial and errors of the ants and the selection of the nodes. First, all nodes contain a similar amount of pheromone. The amount of the early pheromone sprayed on each path is calculated using (5).

After an ant finishes the construction of a rule, the amount of pheromone in all nodes of the construction graph is updated. Pheromone update can be conducted using two methods:

1. Increasing the amount of pheromone in the idioms of the construction graph that has been selected during the process of rule construction. This is called pheromone enhancement.
2. The reduction of pheromone in the idioms of the construction graph that have not been selected during the construction process. This is called pheromone evaporation.

The possibility of selecting $term_{ij}$ by the following ants increases during the process of pheromone enhancement since the current ants have approved its usefulness. The amount of pheromone sprayed on $term_{ij}$ depends on the quality of the rule constructed by an ant. In other words, the higher the quality of the rule is, the more amount of pheromone will be sprayed on $term_{ij}$.

The quality of the rule generated by ants is indicated with Q (9).

$$Q(R_t) = \text{sensitivity}(R_t) \times \text{specificity}(R_t) \qquad (9)$$

In addition, it can be calculated using (10):

$$Q(R_t) = \frac{TP}{TP + FN} \times \frac{TN}{TN + FP} \qquad (10)$$

where R_t is a rule that has been produced through ant_t.

$TP(\mathit{True\ Positives})$ indicates the number of samples covered by the rule that is in the class predicted by the rule.

$FP(\mathit{False\ Positives})$ indicates the number of samples covered by the rule that is not in the class predicted by the rule.

$FN(\mathit{False\ Negatives})$ indicates the number of samples that are not covered by the rule, but are within the class predicted.

$TN(\mathit{True\ Negatives})$ indicates the number of samples that are not covered by the rule, and are not within the class predicted.

The greater the value of Q (in the range of $0 < Q < 1$) is, the higher quality the rule will have. Pheromone is enhanced in the following manner (11):

$$\tau_{ij}(t+1) = \tau_{ij}(t) + \tau_{ij}(t).Q \qquad (11)$$

The above equation is used for updating the pheromone of any term within the generated rule. Thus, according to (11), it can be concluded that the quality of the rule determines the amount of pheromone that is to be increased.

Pheromone begins to evaporate after pheromone enhancement for the idioms that were involved in the process of rule generation. During this process, the evaporation factors ρ is multiplied by the number of idioms that have not used τ_{ij}. The evaporation factor is in the range of $0 \le \rho \le 1$. Pheromone is evaporated based on (12):

$$\tau_{ij}(t+1) = \tau_{ij}(t).(1-\rho) + \rho \qquad (12)$$

In the early Ant-Miner, pheromone is evaporated after pheromone enhancement through normalizing the amount of pheromone τ_{ij} for each term $term_{ij}$. This normalization is carried out through dividing the amount of each τ_{ij} by total τ_{ij} of the nodes within the construction graph.

After the production of each rule, only the pheromone of those idioms that were involved in the process of rule production enhances. Thus, during normalization (after the pheromone enhancement of idioms that were involved) the amount of the pheromone of each unused term is divided by the total pheromone of all idioms. Finally, the amount of pheromone in the used idioms increases, while it decreases in the unused idioms [31].

The Proposed Methods

In this section, two methods have been proposed. The general form of the proposed method is given in Table 2, which is modified by the two methods GLNO and GEQC, which are described in sections A and B.

A. The Generalized Logical Negation Operator (GLNO) Method

Using the GLNO method in the construction of rule records, idioms that include the rules can be made in the form of <Not attribute = value>. Compared to the idioms of the early algorithm, these idioms are more compatible with constructing rules with high coverage. The advantage of this generalization is the reduction of the produced rules, which results in greater understandability of the output. A simple modification is made in the construction graph to apply this generalization.

Using the GLNO method not only reduces the set of produced rules but also enhances stratification accuracy. This section explains the details of this generalization and the modifications made to the algorithm to support it.

Table 2: NEW Ant – Miner algorithm

```
Begin NEW Ant-Miner
training_set ← all training cases;
discovered_rule_set ← ∅;
While (|training_set| > max_uncovered_cases)
    InitializePheromoneAmounts();
    CalculateHeuristicValues();
    $R_{best} \leftarrow \emptyset$;
    $i \leftarrow 0$;
    Repeat
        $SelectRuleClass(ant_i)$;
        $ConstructRuleAntecedent(ant_i)$;
        $R_{current} \leftarrow PruneRule(ant_i)$;
        $Q_{current} \leftarrow CalculateRuleQuality(R_{current})$;
        $UpdatePheromone(R_{current})$;
        If $Q_{current} > Q_{best}$ then
            $R_{best} \leftarrow R_{current}$;
        End if
        $i \leftarrow i + 1$;
    Until $i = \max\_trials$ OR $Convergence()$
    $discovered\_rule\_set \leftarrow discovered\_rule\_set + R_{bes}$;
    $training_{set} \leftarrow training_{set} - Cases(R_{best})$;
End while
End NEW Ant-Miner
```

As mentioned before, the rule will be produced in the following form:

$$\text{IF } < A_i = v_{ij} > AND\ < A_k = v_{kl} > AND \ldots \text{THEN Class}$$

Regarding the application of the GLNO method in the records of the produced rules, the positive and negative logical values per each attribute are added to the construction graph. The set of graphs (N) in the construction graph will be in the form of (13):

$$N = \bigcup_{i=1}^{n} v_{ij} \cup \bigcup_{i=1}^{n} \bar{v}_{ij} \quad , j \in \{1,2,\ldots,l\} \tag{13}$$

Thus, the components of the decision in the construction graph permit the rule to be produced in the following manner:

$$\text{IF } < A_i = v_{ij} > \text{AND } < A_k \text{ NOT } = v_{kl} > \text{AND} \ldots \text{THEN Class}$$

The negative values are added to an attribute that has more than two values in its domain. In addition, the construction of idioms in the form of < attribute = value > is supported. An instance of the rules produced based on through the GLNO method has been illustrated below:

$$\text{IF <Price = Low> AND <Condition NOT = Bad> THEN <Class = Buy>}$$

Idioms that apply the GLNO method are more likely to be compatible with ordinary phrases. This results in the generation of rules with higher convergence. If the GLNO method is adopted to produce grouping rules on the applied datasets, three rules will be required for the accurate classification of the overall dataset. These rules have been described below.

1) IF <Condition NOT = Bad> THEN <Class = Buy>
2) ELSE IF <Condition = Bad> AND <Safety = Very Good> THEN <Class = Wait>
3) ELSE <Class = Don’t Buy>

Since the produced rules have more convergence with the GLNO method, the size of the output rule will be smaller than the rule set produced without applying that method. The following rules have been produced without applying the GLNO method:

1) IF < Condition = Bad > AND <Safety = Very Good> THEN <Class = Wait>
2) ELSE IF < Condition = Bad > AND <Safety = Good> THEN <Class = Don‘t Buy>
3) ELSE IF < Condition = Bad > AND <Safety = Bad> THEN <Class = Don‘t Buy>
4) ELSE <Class = Buy>

At least four rules are required for correctly classifying the previously labeled samples (or three rules for partially classifying some samples). Though using negative attributes doubles the size of the construction graphs it can lead to the generation of rules with a higher convergence rate in training samples. As a result, a smaller number of rules is produced, and output understand ability is enhanced. In addition, the number of iterations required to reach the threshold of samples to be covered – and consequently the overall time for executing the algorithm – is reduced. The practical results of executing the algorithm indicate accurate performance of the algorithm in addition to the points mentioned above.

It is not necessary to make changes in the Ant-Miner to make it ready for the GLNO method. The process of pheromone update in negative values is like ordinary pheromone update, and a method similar to the one applied to calculate ordinary heuristic value is employed to calculate such values. For this purpose, (14) or any other heuristic function used in various versions of Ant-Miner can be applied.

$$\eta_{ij,k} = \frac{|term_{ij}, k| + 1}{|term_{ij}| + No\_of\_Classes} \tag{14}$$

where:

$\eta_{ij,k}$ is the heuristic value for $term_{ij}$ belonging to class k .

$|term_{ij}, k|$ is the number of training samples that contain $term_{ij}$ and belong to class k.

$|term_{ij}|$ is the number of training samples that contain $term_{ij}$.

$No\_of\_Classes$ indicates the number of classes.

It has to be pointed out that while using the GLNO method, the selection of the function for the evaluation of the rules affects output in idioms of classifier accuracy. That is because using the GLNO method produces rules with high coverage, which can damage the reliability of rules (being influential on the accuracy of a classifier). Using an evaluation function that puts greater emphasis on rule reliability, this obstacle can be removed and the relationship between rule coverage and classifier accuracy can be balanced. To implement the GLNO method, some minor details have to be made on the data structure of the early Ant-Miner. The first change is the addition of a new data field to the data structure of nodes in the construction graph. This field is of Boolean type and indicates that a particular node is the attribute or value of the GLNO method. The second change has to be made in the data structure of the construction graph. Every different value present in the domain of an attribute is added to the construction graph twice; once with the False value, and once again with the True value. Hence, the values of each attribute in the construction graph have two components: with and without the GLNO method. This is performed only on attributes that have more than two values in their domain.

B. *The Generalized Exacerbation of Quality Conflict (GEQC) Method*

During the process of pheromone update in the ordinary ACO algorithms, the amount of the sprayed pheromone is a function of the quality of rules. The objective of the GEQC method is to strengthen the conflict between not-found, weak, good, and superior solutions.

This method is a new strategy of pheromone update where ants with high-quality solutions are motivated through increasing the amount of pheromone sprayed on the trail that they have found; conversely, the ants that find weaker solutions are punished through eliminating pheromone from their trails.

To apply this concept to Ant-Miner, the quality of the produced rule including its reliability and support is taken into consideration (13). If the reliability of the produced rule exceeds the threshold $\varphi_{۱}$, the pheromone of this rule has to be strengthened. On the other hand, if the reliability is lower the threshold $\varphi_{۲}$, the pheromone for this rule has to be eliminated. This operation can be observed in (15):

$$\mathrm{PH_{IJ}(T)} = \begin{cases} 2.\,\mathrm{Q(R_T)}, & \text{IF CONFIDENCE}(\mathrm{R_T}) > \Phi_1 \\ \mathrm{Q(R_T)}, & \text{IF } \Phi_2 < Confidence(\mathrm{R_T}) < \Phi_1 \\ 2 - \mathrm{Q(R_T)}, & \text{IF CONFIDENCE}(\mathrm{R_T}) < \Phi_2 \end{cases} \qquad (15)$$

where:

$Ph_{ij}(t)$ indicates the amount of pheromone that has to be sprayed in *t* iterations.

$\mathrm{Q(R_t)}$ is the quality of the rule $\mathrm{R_t}$ produced by the function of rules quality assessment (9).

φ_1 and φ_2 are the upper and lower bounds of the reliability of rules where quality conflict exacerbates.

This strategy has several advantages. First, high-quality rules receive significantly more pheromone compared to the weaker or ordinary ones, and this results in faster convergence. Second, it ensures a balance between the output quality of the produced rules (influenced by the rule support) and the classificatory accuracy of such rules (influenced through the reliability of the rules). For instance, several values of attributes occur more frequently in the samples of training sets. This results in enhancing the role of support in quality evaluation, and an increase in the overall quality of the rules regardless of rule reliability. Thus, the GEQC method acts towards the reliability of the rules to prevent the production of rules with low quality in idioms of stratification. Third, weaker rules are punished through eliminating their pheromone so that the unsurveyed groups gain more chance of being selected in the following iterations. To explain, their pheromone may exceed that of the weaker groups, and the exploratory aspect of the algorithm can be enhanced in this manner. The GEQC method is used in a hybrid form with other methods. In other words, this method can be used as a pheromone update strategy along with other methods.

Datasets

For the purpose of evaluating the early and developed versions of the Ant-Miner algorithm, 8 public datasets that are accessible in the University of California website [32] were applied. The major characteristics of the selected datasets have been indicated in Table 3.

Table 3: The characteristics of the datasets used in the experiment

Dataset	Number of samples	Number of attributes	Number of classes
Car Evaluation	1728	6	4
Nursery	12960	8	5
Tic-Tac-To	958	9	2
Mushrooms	8124	22	2
Dermatology	366	33	6
Soybean	683	35	19
Contraceptive Method Choice	1473	9	3
BDS	1248	8	2

The developed Ant-Miner cannot be applied to continuous attributes, and such attributes have to be converted into discrete ones during a pre-processing stage. The selected datasets include only discrete attributes for the purpose of preventing the interference of discretization methods in the experiment.

The Experimental Approach

The ten-fold cross-validation technique was applied to divide the dataset into a training and a test set with a ratio of 90 to 10 percent. Each pair in the training and test sets was used 10 times with every combination of the developed and early Ant-Miner, and the obtained means were written. The process of ten-fold cross-validation was used for each dataset applying a different random division of training and test samples.

The number of the produced rules (indicating the understandability of the rules), the mean number of tests in each iteration (indicating the tests required to attain convergence), and the accuracy of the produced rules in quality evaluation were recorded.

Parameters Used in the Methods

The following parameters have the same values over the entire tests:

The number of ants (number_of_ants) = 5

The number of trials per ant (number_of_trials per_ants) = 100

The number of iterations needed for convergence (no_rules_converg) = 10

(This parameter is used for testing the convergence of the colony to a particular rule. If a similar rule is made by 10 ants, convergence is fulfilled and the iterations are stopped.)

The maximum uncovered samples (max_uncovered_cases) = 10%

(If the number of uncovered samples is larger than the value set for this parameter, the algorithm has to continue in order to explore more rules and cover more samples.)

The overall number of iterations (Number of Global Iterations) = 50

(the maximum number of overall iterations required to explore rules, which can bring about minimum coverage in training samples, is determined here.)

The Quality Contrast Intensifier Thresholds (φ_1, φ_2)

(values 0.75 and 0.35 are set for these parameters, respectively.)

The overall number of tests in each iteration is 500, which is obtained by multiplying the number of ants in the number of iterations per ants. In addition, this number indicates the maximum number of tests in each iteration, and the iteration process stops when no_rules_converg equals the produced rules.

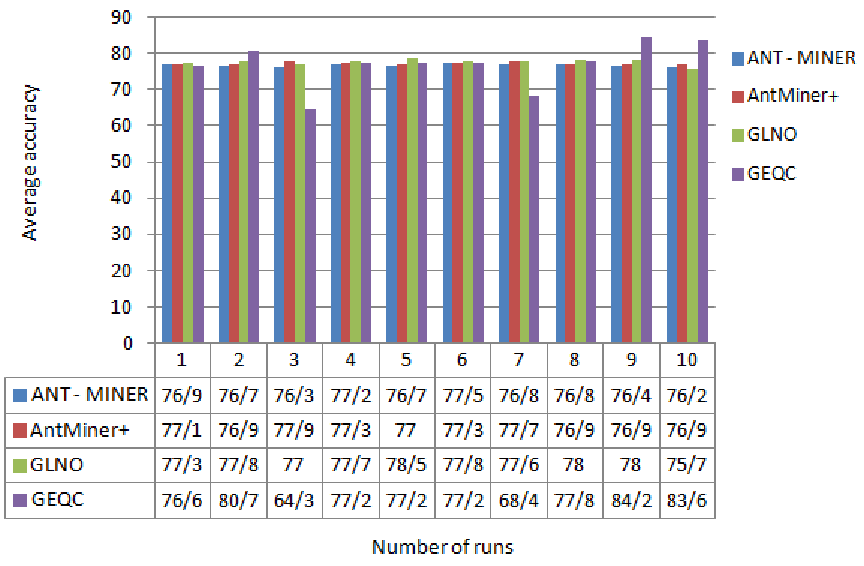
Test Results

In this section, the results of the tests have been presented. The results of each dataset have been presented separately, and each section includes a table (Table 4, Table 5, Table 6, Table 7, Table 8, Table 9, Table 10, and Table 11) which show the summary of test results after applying each development of Ant-Miner singularly or in a hybrid manner and for both early and new system, analysis of the results, and a diagram (Fig. 1, Fig. 2, Fig. 3, Fig. 4, Fig. 5, Fig. 6, Fig. 7 and Fig. 8).

Using the GEQC method in the first dataset reduced the mean number of rules produced through the algorithm since the produced results showed a higher convergence rate. In comparison to the early algorithm, this method enhanced the mean accuracy in most scenarios.

Table 4: Details of the results of the car evaluation dataset

Method	ANT – MINER		AntMiner+		GLNO		GEQC	
Execution	Rules	Accuracy	Rules	Accuracy	Rules	Accuracy	Rules	Accuracy
1	8	76.90	10	77.1	5	77.27	11	76.60
2	8	76.66	10	76.9	5	77.75	8	80.70
3	8	76.29	10	77.9	6	77.04	9	64.32
4	8	77.15	10	77.3	5	77.65	8	77.19
5	8	76.73	10	77	6	78.53	8	77.19
6	8	77.48	10	77.3	6	77.76	8	77.19
7	8	76.81	10	77.7	5	77.63	8	68.42
8	8	76.82	10	76.9	6	78.03	8	77.77
9	8	76.37	10	76.9	6	78.00	9	84.21
10	8	76.23	10	76.9	5	75.68	9	83.62



	1	2	3	4	5	6	7	8	9	10
🟦 ANT - MINER	76/9	76/7	76/3	77/2	76/7	77/5	76/8	76/8	76/4	76/2
🟥 AntMiner+	77/1	76/9	77/9	77/3	77	77/3	77/7	76/9	76/9	76/9
🟩 GLNO	77/3	77/8	77	77/7	78/5	77/8	77/6	78	78	75/7
🟪 GEQC	76/6	80/7	64/3	77/2	77/2	77/2	68/4	77/8	84/2	83/6

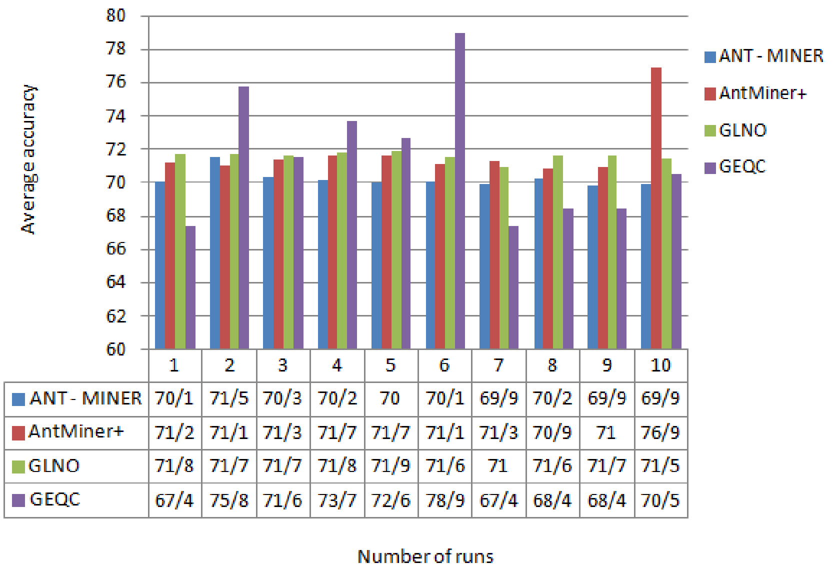
Fig. 1: Comparative diagram of the proposed methods with the initial method for the car evaluation dataset.

Using the GEQC method in the second dataset reduced the mean number of produced rules.

In comparison to the early methods, using newer methods resulted in the minimum number of rules.

Table 5: Details of the results of Tic-Tac-To dataset

Method	ANT – MINER		AntMiner+		GLNO		GEQC	
Execution	Rules	Accuracy	Rules	Accuracy	Rules	Accuracy	Rules	Accuracy
1	7	70.1	8	71.2	5	71.76	6	67.36
2	7	71.51	8	71.1	5	71.68	6	75.78
3	6	70.32	9	71.3	5	71.65	5	71.57
4	6	70.20	10	71.7	5	71.81	7	73.68
5	6	70.00	10	71.7	5	71.88	7	72.63
6	5	70.06	8	71.1	5	71.57	7	78.94
7	6	69.93	9	71.3	5	70.98	7	67.36
8	6	70.22	8	70.9	5	71.61	7	68.42
9	6	69.85	9	71	5	71.65	7	68.42
10	6	69.87	10	76.9	5	71.45	6	70.52



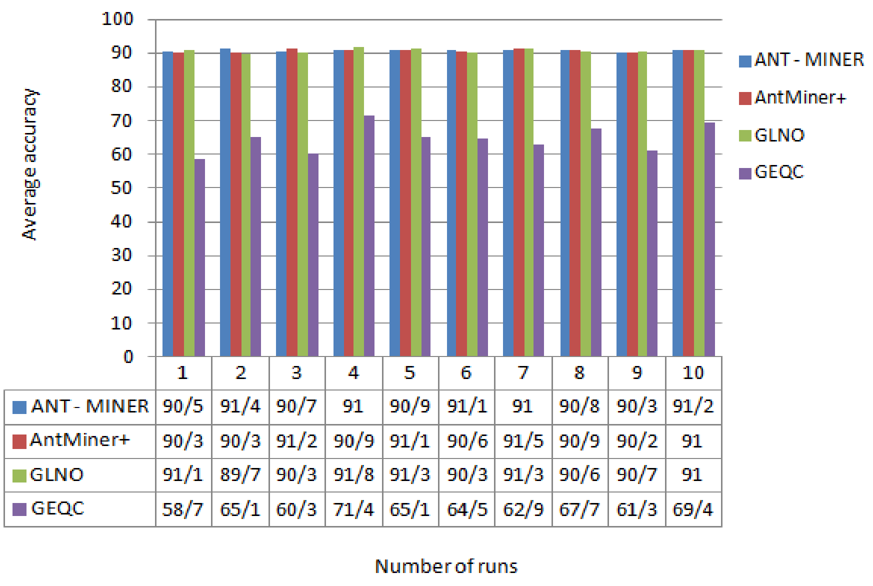
	1	2	3	4	5	6	7	8	9	10
🟦 ANT - MINER	70/1	71/5	70/3	70/2	70	70/1	69/9	70/2	69/9	69/9
🟥 AntMiner+	71/2	71/1	71/3	71/7	71/7	71/1	71/3	70/9	71	76/9
🟩 GLNO	71/8	71/7	71/7	71/8	71/9	71/6	71	71/6	71/7	71/5
🟪 GEQC	67/4	75/8	71/6	73/7	72/6	78/9	67/4	68/4	68/4	70/5

Fig. 2: Comparative diagram of the proposed methods with the initial method for the Tic-Tac-To dataset.

Using the GEQC method in the third dataset produced the minimum number of rules with very few tests and acceptable accuracy.

Table 6: Details of the results of mushrooms dataset

Method Execution	ANT – MINER		AntMiner+		GLNO		GEQC	
	Rules	Accuracy	Rules	Accuracy	Rules	Accuracy	Rules	Accuracy
1	6	90.51	5	90.3	4	91.13	9	58.73
2	6	91.38	5	90.3	4	89.73	9	65.07
3	5	90.67	9	91.2	4	90.31	9	60.31
4	6	91.03	10	90.9	4	91.79	9	71.42
5	6	90.92	10	91.1	4	91.28	9	65.07
6	6	91.08	10	90.6	4	90.27	9	64.51
7	5	90.98	10	91.5	4	91.29	9	62.90
8	6	90.78	10	90.9	5	90.55	9	67.74
9	5	90.26	10	90.2	4	90.65	9	61.29
10	5	91.19	9	91	4	90.96	9	69.35



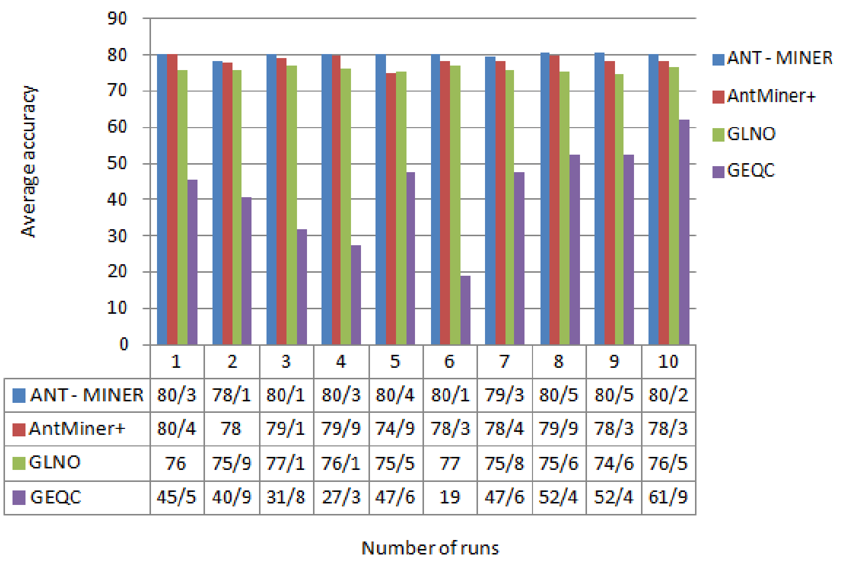
	1	2	3	4	5	6	7	8	9	10
🟦 ANT - MINER	90/5	91/4	90/7	91	90/9	91/1	91	90/8	90/3	91/2
🟥 AntMiner+	90/3	90/3	91/2	90/9	91/1	90/6	91/5	90/9	90/2	91
🟩 GLNO	91/1	89/7	90/3	91/8	91/3	90/3	91/3	90/6	90/7	91
🟪 GEQC	58/7	65/1	60/3	71/4	65/1	64/5	62/9	67/7	61/3	69/4

Fig. 3: Comparative diagram of the proposed methods with the initial method for the mushrooms dataset.

Using the GEQCmethod in the fourth dataset produced the minimum number of rules with few tests and acceptable accuracy.

Table 7: Details of the results of Nursery dataset

Method Execution	ANT – MINER		AntMiner+		GLNO		GEQC	
	Rules	Accuracy	Rules	Accuracy	Rules	Accuracy	Rules	Accuracy
1	8	80.29	8	80.4	5	75.98	3	45.45
2	7	78.14	9	78	4	75.91	2	40.90
3	8	80.13	9	79.1	5	77.14	3	31.81
4	8	80.26	9	79.9	5	76.06	3	27.27
5	8	80.37	7	74.9	5	75.48	3	47.61
6	8	80.08	9	78.3	5	77.00	2	19.04
7	8	79.31	9	78.4	5	75.80	3	47.61
8	8	80.53	9	79.9	5	75.55	3	52.38
9	8	80.51	9	78.3	4	74.63	3	52.38
10	8	80.19	9	78.3	5	76.54	3	61.90



	1	2	3	4	5	6	7	8	9	10
🟦 ANT - MINER	80/3	78/1	80/1	80/3	80/4	80/1	79/3	80/5	80/5	80/2
🟥 AntMiner+	80/4	78	79/1	79/9	74/9	78/3	78/4	79/9	78/3	78/3
🟩 GLNO	76	75/9	77/1	76/1	75/5	77	75/8	75/6	74/6	76/5
🟪 GEQC	45/5	40/9	31/8	27/3	47/6	19	47/6	52/4	52/4	61/9

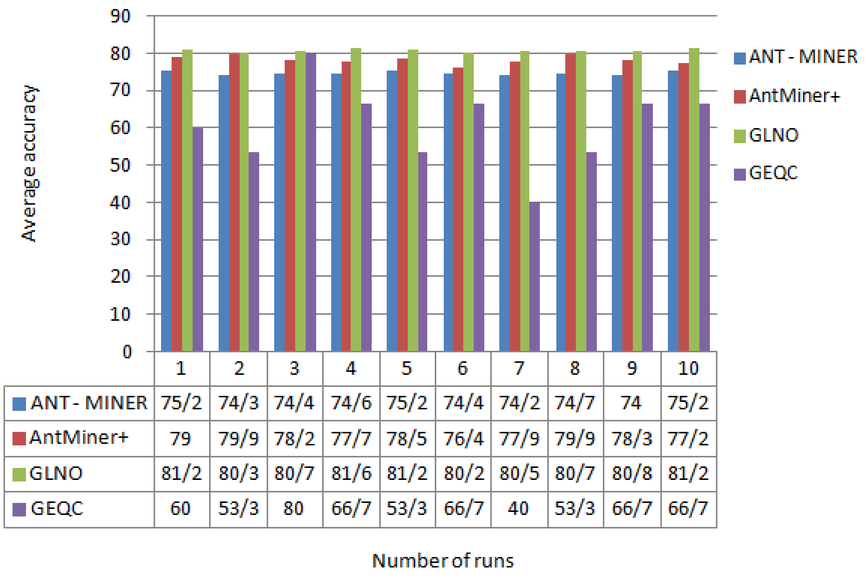
Fig. 4: Comparative diagram of the proposed methods with the initial method for the Nursery dataset.

The employment of the GEQC method in the fifth dataset, in comparison to the early algorithm, produced a smaller number of rules with significant accuracy.

The hybrid form of the GEQC method and the early algorithm produced the minimum number of rules with an acceptable level of accuracy.

Table 8: Details of the results of the Dermatology dataset

Method	ANT – MINER		AntMiner+		GLNO		GEQC	
Execution	Rules	Accuracy	Rules	Accuracy	Rules	Accuracy	Rules	Accuracy
1	8	75.18	9	79	7	81.15	4	60
2	9	74.29	10	79.9	7	80.27	4	53.33
3	8	74.43	8	78.2	7	80.72	3	80
4	8	74.62	8	77.7	7	81.56	4	66.66
5	8	75.24	9	78.5	7	81.16	4	53.33
6	8	74.43	8	76.4	7	80.24	4	66.66
7	8	74.21	8	77.9	7	80.54	2	40
8	8	74.67	9	79.9	7	80.72	2	53.33
9	8	74.02	9	78.3	7	80.78	4	66.66
10	9	75.17	8	77.2	7	81.24	3	66.66



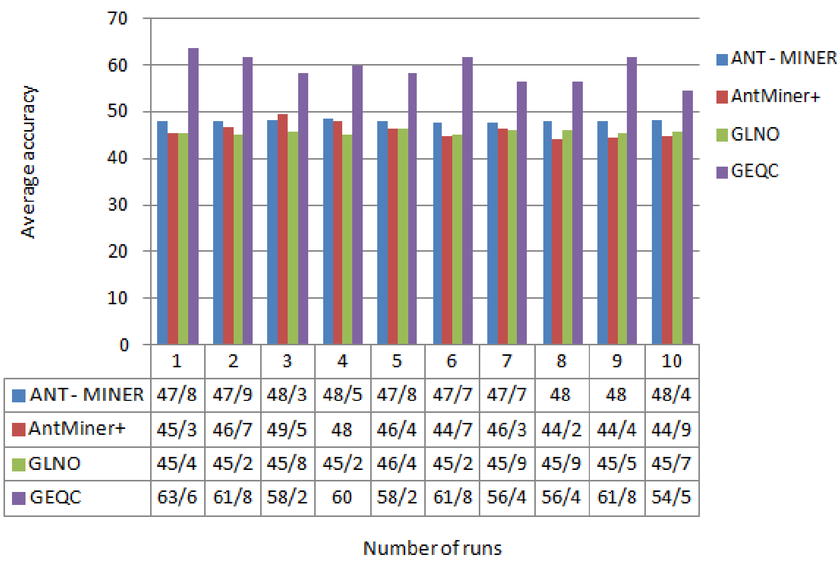
	1	2	3	4	5	6	7	8	9	10
🟦 ANT - MINER	75/2	74/3	74/4	74/6	75/2	74/4	74/2	74/7	74	75/2
🟥 AntMiner+	79	79/9	78/2	77/7	78/5	76/4	77/9	79/9	78/3	77/2
🟩 GLNO	81/2	80/3	80/7	81/6	81/2	80/2	80/5	80/7	80/8	81/2
🟪 GEQC	60	53/3	80	66/7	53/3	66/7	40	53/3	66/7	66/7

Fig. 5: Comparative diagram of the proposed methods with the initial method for the Dermatology dataset.

The GEQC method in the sixth dataset helped reduce the number of produced rules. Using this method, alongside the other method, and the quality contrast intensifier resulted in the minimum number of rules.

Table 9: Details of the results of the Soybean dataset

Method Execution	ANT – MINER		AntMiner+		GLNO		GEQC	
	Rules	Accuracy	Rules	Accuracy	Rules	Accuracy	Rules	Accuracy
1	11	47.79	9	45.3	8	45.35	6	63.63
2	11	47.89	9	46.7	9	45.23	6	61.81
3	10	48.31	9	49.5	9	45.77	5	58.18
4	11	48.52	9	48	9	45.17	6	60
5	11	47.78	9	46/4	9	46.42	7	58.18
6	11	47.71	9	44/7	9	45.17	7	61.81
7	11	47.65	9	46/3	9	45.93	7	56.36
8	11	47.96	9	44/2	9	45.93	6	56.36
9	11	48.03	9	44/4	9	45.51	6	61.81
10	11	48.38	9	44/9	9	45.67	7	54.54



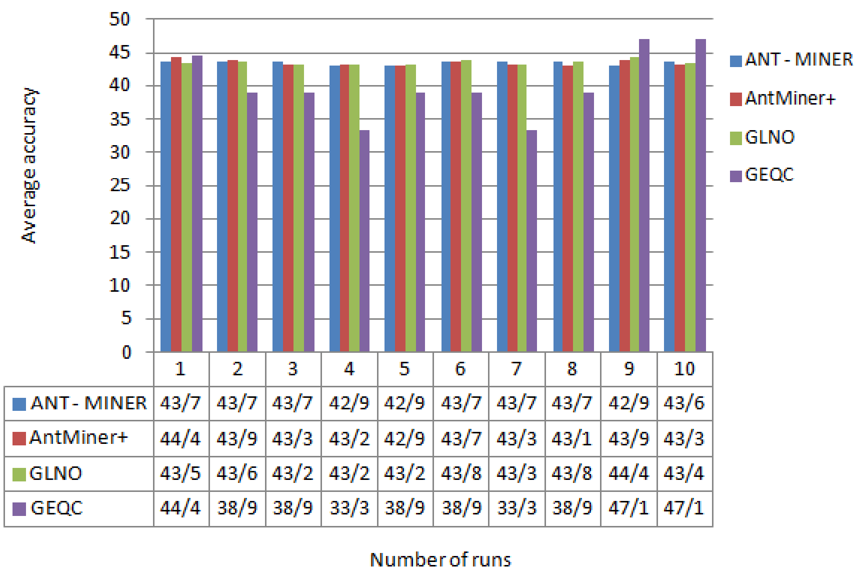
	1	2	3	4	5	6	7	8	9	10
🟦 ANT - MINER	47/8	47/9	48/3	48/5	47/8	47/7	47/7	48	48	48/4
🟥 AntMiner+	45/3	46/7	49/5	48	46/4	44/7	46/3	44/2	44/4	44/9
🟩 GLNO	45/4	45/2	45/8	45/2	46/4	45/2	45/9	45/9	45/5	45/7
🟪 GEQC	63/6	61/8	58/2	60	58/2	61/8	56/4	56/4	61/8	54/5

Fig. 6: Comparative diagram of the proposed methods with the initial method for the Soybean dataset.

The GEQC method in the seventh dataset, in comparison to the early algorithm, led to the production of a smaller number of rules. This method, alongside the quality contrast intensifier, produced the minimum number of rules.

Table 10: Details of the results of the Contraceptive Method Choice dataset

Method Execution	ANT – MINER		AntMiner+		GLNO		GEQC	
	Rules	Accuracy	Rules	Accuracy	Rules	Accuracy	Rules	Accuracy
1	8	43.66	9	44.4	7	43.47	2	44.44
2	9	43.68	9	43.9	6	43.62	3	38.88
3	9	43.71	9	43.3	6	43.17	1	38.88
4	8	42.87	9	43.2	7	43.24	2	33.33
5	9	42.93	9	42.9	6	43.18	2	38.88
6	8	43.74	8	43.7	6	43.82	2	38.88
7	9	43.67	9	43.3	7	43.28	2	33.33
8	9	43.70	9	43.1	7	43.75	1	38.88
9	8	42.94	8	43.9	7	44.40	2	47.05
10	9	43.64	9	43.3	6	43.37	4	47.05



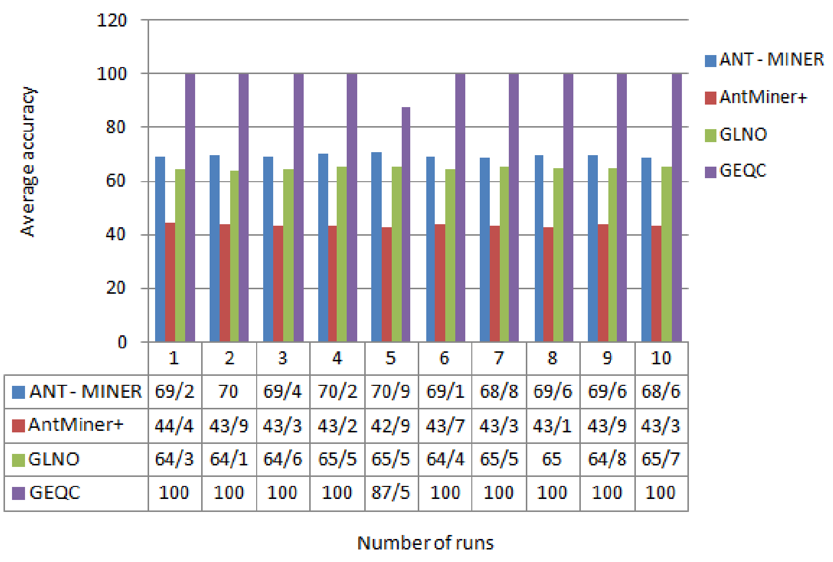
	1	2	3	4	5	6	7	8	9	10
🟦 ANT - MINER	43/7	43/7	43/7	42/9	42/9	43/7	43/7	43/7	42/9	43/6
🟥 AntMiner+	44/4	43/9	43/3	43/2	42/9	43/7	43/3	43/1	43/9	43/3
🟩 GLNO	43/5	43/6	43/2	43/2	43/2	43/8	43/3	43/8	44/4	43/4
🟪 GEQC	44/4	38/9	38/9	33/3	38/9	38/9	33/3	38/9	47/1	47/1

Fig. 7: Comparative diagram of the proposed methods with the initial method for the Contraceptive Method Choice dataset.

Using the GEQC method in the eighth dataset produced a smaller number of rules. The typical method significantly increased the number of the produced rules and decreased their size (particularly when combined with the GEQC method).

Table 11: Details of the results of the BDS dataset

Method	ANT – MINER		AntMiner+		GLNO		GEQC	
Execution	Rules	Accuracy	Rules	Accuracy	Rules	Accuracy	Rules	Accuracy
1	11	69.17	8	44.4	4	64.31	5	100
2	11	69.96	8	43.9	4	64.09	5	100
3	11	69.42	8	43.3	4	64.57	5	100
4	11	70.22	8	43.2	4	65.53	5	100
5	11	70.92	8	42.9	4	65.53	5	100
6	11	69.07	8	43.7	4	64.40	5	87.50
7	11	68.76	8	43.3	4	65.47	5	100
8	11	69.58	8	43.1	4	64.96	5	100
9	11	69.68	8	43.9	4	64.76	5	100
10	11	68.57	8	43.3	4	65.69	4	100



	1	2	3	4	5	6	7	8	9	10
🟦 ANT - MINER	69/2	70	69/4	70/2	70/9	69/1	68/8	69/6	69/6	68/6
🟥 AntMiner+	44/4	43/9	43/3	43/2	42/9	43/7	43/3	43/1	43/9	43/3
🟩 GLNO	64/3	64/1	64/6	65/5	65/5	64/4	65/5	65	64/8	65/7
🟪 GEQC	100	100	100	100	87/5	100	100	100	100	100

Fig. 8: Comparative diagram of the proposed methods with the initial method for the BDS dataset.

As observed in the results, the efficiency of the algorithm can be significantly enhanced through using the proposed optimization methods.

Conclusion

In the present study, two novel approaches were proposed for improving the Ant-Miner algorithm. As observed in the experiments conducted on public datasets, employing these methods increased the overall quality of the algorithm.

In other words, the GLNO method produced a smaller number of rules but increased the number of construction graphs and the number of trials in each trial.

In addition, the GEQC method prevented the production of low-quality rules (i.e., it enhanced the reliability of the output rules). In addition, it provided the unsurveyed nodes a greater chance to be selected in the following iterations (i.e., it increased the exploratory power of the algorithm). It is clear from the results that the two proposed methods have increased the classification accuracy compared to the previous methods, except in some datasets where the reduction of the classification accuracy has had the advantage of reducing the number of production rules, which are considered as the whole advantage of the proposed methods.

Future suggestions

It is recommended that the combination of the new system with the Ant-Miner algorithm (a version of Ant-Miner capable of working with continuous attributes) be considered if desired. In addition, the use of multiple pheromones for clustering can be considered. There is also a need to strike a balance between coverage and predictive accuracy, especially when the generalized method of logical negation operator is used with multiple pheromones.

Author Contributions

H. Nosrati Nahook designed the experiments, analysis the data and wrote the manuscript. S. Tabatabaei carried interpreted the results and revised the manuscript.

Acknowledgment

This work is completely self-supporting, thereby no any financial agency's role is available. This work is completely self-supporting, thereby no any financial agency's role is available.

Conflict of Interest

The authors declare no potential conflict of interest regarding the publication of this work. In addition, the ethical issues including plagiarism, informed consent, misconduct, data fabrication and, or falsification, double publication and, or submission, and redundancy have been completely witnessed by the authors.

Abbreviations

ACO	Ant Colony Optimization
TSP	Traveling Salesman Problem
COP	Combinational Optimization Problems
MuLAM	Multi-Label Ant-Miner
MMAS	MAX-MIN system
GLNO	Generalized Logical Negation Operator
GEQC	Generalized Exacerbation of Quality Conflict

References

[1] A. Abraham, C. Grosan, M. Chis, Swarm Intelligence in Data Analysis, Studies in Computational Intelligence, vol. 34, Springer: 1-20, 2006.

[2] M. Dorigo, A. Colorni, V. Maniezzo, "The ant system: optimization through a colony of cooperating agents," IEEE Trans. Syst. Man Cybern. Part B Cybern., 26: 29–41, 1996.

[3] S. Tabatabaei, H. Nosrati Nahook, "A new routing protocol in MANET using Cuckoo optimization algorithm," J. Electr. Comput. Eng. Innovations, 9(1): 75-82, 2021.

[4] M. Medland, F.E.B Otero, A.A. Freitas, "Improving the cAnt-Miner PB classification algorithm," presented at the International Conference on Swarm Intelligence. Springer, Berlin, Heidelberg, 2012.

[5] M.G.H. OMRAN, S. AL-SHARHAN, "Improved continuous Ant Colony Optimization algorithms for real-world engineering optimization problems," Eng. Appl. Artif. Intell., 85: 818-829, 2019.

[6] R.S. Parpinelli, H.S. Lopes, A.A. Freitas, "Data mining with an ant colony optimization algorithm," IEEE Trans. Evol. Comput., 6(4): 321-332, 2002.

[7] W.J. Jiang, Y.H. Xu, Y.S. Xu, "A novel data mining algorithm based on ant colony system," in Proc. 2005 International Conference on Machine Learning and Cybernetics, 3: 1919-1923. 2005.

[8] B. Liu, H.A. Abbas, B. McKay, "Classification rule discovery with ant colony optimization," in Proc. IEEE/WIC International Conference on Intelligent Agent Technology (IAT): 83-88, 2003.

[9] J. He, D. Long, C. Chen, "An improved ant-based classifier for intrusion detection," in Proc. Third International Conference on Natural Computation (ICNC 2007), 4: 819-823, 2007.

[10] A. Chan, A. Freitas, "A new classification-rule pruning procedure for an ant colony algorithm," in Proc. International Conference on Artificial Evolution (Evolution Artificielle): 25-36, Springer, Berlin, Heidelberg, 2005.

[11] K. Thangavel, P. Jaganathan, "Rule mining algorithm with a new ant colony optimization algorithm," in Proc. International Conference on Computational Intelligence and Multimedia Applications (ICCIMA 2007), 2: 135-140, 2007.

[12] D. Martens, M. D. Backer, R. Haesen, J. Vanthienen, M. Snoeck, B. Baesens, "Classification with ant colony optimization," IEEE Trans. Evol. Comput., 11(5): 651-665, 2007.

[13] Y. Chen, L. Chen, L.Tu, "Parallel ant colony algorithm for mining classification rules," in Proc. 2006 IEEE International Conference on Granular Computing: 85-90, 2006.

[14] B. Jyoti, A.K. Sharma, "AntMiner: Bridging the gap between data mining classification rule discovery and bio-inspired algorithms," in Proc. International Conference on IoT Inclusive Life (ICIIL 2019), NITTTR Chandigarh, India: 269-277, 2020.

[15] B.K. Nanda, S. Dehuri, "Ant Miner: A hybrid pittsburgh style classification rule mining algorithm," Int. J. Artif. Intell. Mach. Learn. (IJAIML), 10(1): 45-59, 2020.

[16] A. Singh, Akansha, G. Gupta, "ANT_FDCSM: A novel fuzzy rule miner derived from ant colony meta-heuristic for diagnosis of diabetic patients," J. Intell. Fuzzy Syst. 36(1): 747-760, 2019.

[17] M. Durgadevi, R. Kalpana, "Medical distress prediction based on classification rule discovery using ant-miner algorithm," in Proc. 2017 11th International Conference on Intelligent Systems and Control (ISCO): 88-92, 2017.

[18] H.N.K. Al-Behadili, K.R. Ku-Mahamud, R. Sagban, "Ant colony optimization algorithm for rule-based classification: Issues and potential solutions," J. Theor. Appl. Inf. Technol., 96(21): 7139-7150, 2018.

[19] J. Kozak, Decision Tree and Ensemble Learning According to Ant Colony Optimization. Springer, Cham: 29-44, 2019.

[20] L. Wan, C. Du, "An approach to evaluation of environmental benefits for ecological mining areas according to ant Colony algorithm," Earth Sci. Inf., 14: 797–808, 2021.

[21] B. Shuang, J. Chen, Z. Li, "Study on hybrid PS-ACO algorithm," Appl. Intell., 34(1): 64-73, 2011.

[22] R.S. Parpinelli, A. Freitas, "Data analysis with an ant colony optimization algorithm," IEEE Trans. Evol. Comput., 6: 321–332, 2002 .

[23] B. Liu, H. Abbass, B. McKay, "Density-Based exploratory for rule discovery with Ant-Miner," in Proc. 6th Australasia-Japan Joint Workshop on Intell. Evol. Syst.: 180–184, 2002.

[24] B. Liu, H. Abbass, B. McKay, "Categorizing rule discovery with ant colony optimization," in Proc. IEEE/WIC Int. Conf. Intell. Agent Technology: 83–88, 2003.

[25] A. Chan, A. Freitas, "A new classification-rule pruning procedure for an ant colony algorithm, Artificial Evolution," Lect. Notes Comput. Sci., 3871: 25–36, 2005.

[26] A. Chan, A., Freitas, "A new ant colony algorithm for multi-label grouping with applications in bioinformatics," in Proc. Genetic and Evolutionary Computation Conf.: 27-34, 2006.

[27] D. Martens et al., "Categorizing with ant colony optimization," IEEE Transactions on Evolutionary Computation. vol. 11, pp. 651–665, 2007.

[28] T. Stützle, M. Dorigo, "ACO algorithms for the traveling salesman problem," Evolutionary Algorithms in Engineering and Computer Science, Wiley: 163-183, 1999.

[29] H.N.K. Al-Behadili, K.R. Ku-Mahamud, R. Sagban, "Rule pruning techniques in the ant-miner classification algorithm and its variants: A review," in Proc. 2018 IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE): 78-84, 2018.

[30] U. Ayub, H. Naveed, W. Shahzad, "PRRAT_AM—An advanced ant-miner to extract accurate and comprehensible classification rules," Appl. Soft Comput.: 106326, 2020.

[31] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufman, 1993.

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

Biographies

Hassan Nosrati Nahook received his bachelor's and master's degrees in Sistan and Baluchestan and Islamic Azad universities, Research Sciences Branch, Tehran, Iran in 2010 and 2013, respectively. His research interests include machine learning, Fuzzy systems and methods, evolutionary processing and bioinformatics.

Shayesteh Tabatabaei received her B.S. and M.S. degrees in Computer Engineering from Islamic Azad University, Shabestar, Iran, in 2006 and 2008 and then continued her study for Ph.D. in Computer Science Branch Software in Islamic Azad University, Science and Research Branch, Tehran, Iran. Her research interests include routing, QOS, security, and clustering for mobile ad hoc networks and wireless sensor networks.

How to cite this paper:
H. Nosrati Nahook, S. Tabatabaei, "Presenting new methods for improving the ant-miner algorithm," J. Electr. Comput. Eng. Innovations, 10(1): 57-74, 2022.

DOI: 10.22061/JECEI.2021.7856.441

URL: https://jecei.sru.ac.ir/article_1556.html