AN ADAPTIVE ANT COLONY OPTIMIZATION ALGORITHM FOR RULE-BASED CLASSIFICATION

HAYDER NASER KHRAIBET AL-BEHADILI

DOCTOR OF PHILOSOPHY
UNIVERSITI UTARA MALAYSIA
2020

Awang Had Salleh
Graduate School
of Arts And Sciences

Universiti Utara Malaysia

PERAKUAN KERJA TESIS / DISERTASI
(Certification of thesis / dissertation)

Kami, yang bertandatangan, memperakukan bahawa
(We, the undersigned, certify that)

HAYDER NASER KHRAIBET AL-BEHADILI

calon untuk Ijazah **PhD**
(candidate for the degree of)

telah mengemukakan tesis / disertasi yang bertajuk:
(has presented his/her thesis / dissertation of the following title):

"AN ADAPTIVE ANT COLONY OPTIMIZATION ALGORITHM FOR RULE-BASED CLASSIFICATION"

seperti yang tercatat di muka surat tajuk dan kulit tesis / disertasi.
(as it appears on the title page and front cover of the thesis / dissertation).

Bahawa tesis/disertasi tersebut boleh diterima dari segi bentuk serta kandungan dan meliputi bidang ilmu dengan memuaskan, sebagaimana yang ditunjukkan oleh calon dalam ujian lisan yang diadakan pada : ***27 Februari 2020.***
That the said thesis/dissertation is acceptable in form and content and displays a satisfactory knowledge of the field of study as demonstrated by the candidate through an oral examination held on: ***February 27, 2020.***

Pengerusi Viva: *(Chairman for VIVA)*	**Assoc. Prof. Dr. Siti Sakira Kamaruddin**	Tandatangan *(Signature)*
Pemeriksa Luar: *(External Examiner)*	**Prof. Dr. Salwani Abdullah**	Tandatangan *(Signature)*
Pemeriksa Dalam: *(Internal Examiner)*	**Assoc. Prof. Dr. Yuhanis Yusof**	Tandatangan *(Signature)*
Nama Penyelia/Penyelia-penyelia: *(Name of Supervisor/Supervisors)*	**Prof. Dr. Ku Ruhana Ku Mahamud**	Tandatangan *(Signature)*
Nama Penyelia/Penyelia-penyelia: *(Name of Supervisor/Supervisors)*	**Dr. Rafid Sagban Abbood Allwawi**	Tandatangan *(Signature)*

Tarikh:
(Date) ***February 27, 2020***

Permission to Use

In presenting this thesis in fulfilment of the requirements for a postgraduate degree from Universiti Utara Malaysia, I agree that the University Library may make it freely available for inspection. I further agree that permission for the copying of this thesis in any manner, in whole or in part, for a scholarly purpose may be granted by my supervisor(s) or, in their absence, by the Dean of Awang Had Salleh Graduate School of Arts and Sciences. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain is not allowed without my written permission. It is also understood that due to recognition shall be given to me and to Universiti Utara Malaysia for any scholarly use, which may be made of any material from my thesis.

Requests for permission to copy or to make other uses of materials in this thesis, in whole or in part, should be addressed to:

Dean of Awang Had Salleh Graduate School of Arts and Sciences
UUM College of Arts and Sciences
Universiti Utara Malaysia
06010 UUM Sintok

Abstrak

Klasifikasi adalah tugas perlombongan data yang penting dengan aplikasi yang berbeza dalam banyak bidang. Pelbagai algoritma klasifikasi telah dikembangkan untuk menghasilkan model klasifikasi dengan ketepatan yang tinggi. Berbeza dengan model klasifikasi kompleks dan sukar yang lain, algoritma klasifikasi berdasarkan peraturan menghasilkan model yang dapat difahami oleh pengguna. Ant-Miner adalah varian pengoptimuman koloni semut dan algoritma pintar yang terkenal yang banyak digunakan dalam klasifikasi berdasarkan peraturan. Walau bagaimanapun, Ant-Miner mempunyai masalah terlebih padan dan mudah jatuh ke dalam optima tempatan yang mengakibatkan ketepatan klasifikasi yang rendah dan peraturan klasifikasi yang kompleks. Dalam kajian ini, pengkelasan Ant-Miner baru dikembangkan, dinamakan Adaptive Genetic Iterated-AntMiner (AGI-AntMiner) yang bertujuan untuk mengelak masalah optima tempatan dan terlebih padan. Komponen AGI-AntMiner meliputi: i) AntMiner Adaptive yang merupakan teknik pra-pemangkasan untuk memilih ambang yang sesuai secara dinamik berdasarkan kualiti peraturan; ii) Genetik AntMiner yang meningkatkan pasca pemangkasan dengan menambah/mengurang istilah dalam dwi cara; dan, iii) Search-AntMiner Tempatan Berterusan yang meningkatkan eksploitasi berdasarkan struktur kawasan berganda. Algoritma AGI-AntMiner yang dicadangkan dinilai pada 16 kumpulan data penanda aras bidang perubatan, kewangan, permainan dan sosial yang diperoleh dari repositori University California Irvine. Prestasi algoritma dibandingkan dengan varian lain dari algoritma klasifikasi berasaskan peraturan Ant-Miner dan state-of-the-art berdasarkan ketepatan klasifikasi dan kerumitan model.Hasil eksperimen membuktikan bahawa algoritma AGI-AntMiner yang dicadangkan lebih unggul dalam dua (2) aspek. Hibridisasi pencarian tempatan di AGI-AntMiner telah meningkatkan mekanisme eksploitasi yang membawa kepada penemuan peraturan klasifikasi yang lebih tepat. Teknik pra-pemangkasan dan pasca pemangkasan baru telah meningkatkan kemampuan pemangkasan untuk menghasilkan peraturan klasifikasi yang lebih pendek yang lebih senang ditafsirkan oleh pengguna. Oleh itu, algoritma AGI-AntMiner yang dicadangkan mampu melakukan pencarian yang cekap dalam mencari peraturan klasifikasi terbaik yang mengimbangkan ketepatan klasifikasi dan kerumitan model untuk mengatasi masalah terlebih padan dan optima tempatan. Induksi peraturan, Pembelajaran mesin, Perlombongan data, Metaheuristik, kecerdasan kawanan.

Kata Kunci: Induksi peraturan, Pembelajaran mesin, Perlombongan data, Metaheuristik, kecerdasan kawanan.

Abstract

Classification is an important data mining task with different applications in many fields. Various classification algorithms have been developed to produce classification models with high accuracy. Differing from other complex and difficult classification models, rules-based classification algorithms produce models which are understandable for users. Ant-Miner is a variant of ant colony optimisation and a prominent intelligent algorithm widely use in rules-based classification. However, the Ant-Miner has overfitting and easily falls into local optima problems which resulted in low classification accuracy and complex classification rules. In this study, a new Ant-Miner classifier is developed, named Adaptive Genetic Iterated-AntMiner (AGI-AntMiner) that aims to avoid local optima and overfitting problems. The components of AGI-AntMiner includes: i) an Adaptive AntMiner which is a pre-pruning technique to dynamically select the appropriate threshold based on the quality of the rules; ii) Genetic AntMiner that improves the post-pruning by adding/removing terms in a dual manner; and, iii) an Iterated Local Search-AntMiner that improves exploitation based on multiple-neighbourhood structure. The proposed AGI-AntMiner algorithm is evaluated on 16 benchmark datasets of medical, financial, gaming and social domains obtained from the University California Irvine repository. The algorithm's performance was compared with other variants of Ant-Miner and state-of-the-art rules-based classification algorithms based on classification accuracy and model complexity. Experimental results proved that the proposed AGI-AntMiner algorithm is superior in two (2) aspects. Hybridization of local search in AGI-AntMiner has improved the exploitation mechanism which leads to the discovery of more accurate classification rules. The new pre-pruning and post-pruning techniques have improved the pruning ability to produce shorter classification rules which are easier to interpret by the users. Thus, the proposed AGI-AntMiner algorithm is capable in conducting an efficient search in finding the best classification rules that balance the classification accuracy and model complexity to overcome overfitting and local optima problems.

Keywords: Rule induction, Machine learning, Data mining, Metaheuristic, Swarm intelligence.

Acknowledgement

First and foremost, I would like to express my gratitude to ALLAH S.W.T, the Almighty, for giving me the opportunity, determination and strength to do my research.

I would like to particularly thank my supervisor Prof. Dr. Ku Ruhana Ku Mahamud, for the continuous support, encouragement, patience, generosity and advice. I would also like to thank my second supervisor Dr. Rafid Sagban from the University of Babylon, for his care, kindness and thoughtful consideration.

To my parents, Naser Khraibet and Majida Ghaeb, and lovely brothers and sisters, I am forever indebted to you all. Your unconditional and selfless support have been the foundation of my life.

To my beloved wife, Noor Habeeb, and prince Mohammed, I would like to appreciate your understanding and wisdom. This journey would not have been possible without your support.

A part of this project would not exist without the help of my friends, as they have given me encouragement and strength. Thank you Ayad Mohammed, Salah Hadi, Hussein Almazini, Hassan Almazini, Salah Mortada, Ammar, Jawad, and Hassan.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

A-AntMiner	Adaptive–AntMiner Classifier
ACO	Ant Colony Optimisation
ACS	Ant Colony System
AGI-AntMiner	Adaptive Genetic Iterated AntMiner
ANN	Artificial Neural Networks
AS	Ant Systems
CART	Classification and Regression Tree
CHID	Chi-Squared Automatic Interaction Detector
CO	Combinatorial Optimisation
DM	Data Mining
FURIA	Fuzzy Unordered Rule Induction Algorithm
GA	Genetic Algorithm
GA-AntMiner	Genetic-Based Ant-Miner
ID3	Iterative Dichotomiser 3
ILS	Iterated Local Search
ILS-AntMiner	Hybridization Ant-Miner Classifier with Iterated Local Search
KNN	K-Nearest Neighbour
MDL	Minimum Description Length
MMAS	Max–Min Ant System
PSO	Particle Swarm Optimisation
QAP	Quadratic Assignment Problem
RBC	Rules-Based Classification
SA	Simulated Annealing
SVM	Support Vector Machines
TS	Tabu Search
TSP	Traveling Salesman Problem
UCI	University of California Irvine Machine Learning Repository

CHAPTER ONE
INTRODUCTION

Data transcend traditional concepts that have been popular in previous decades. Traditional concepts have considered data only for government computer transactions and scientific or business purposes until the beginning of the digital revolution and the dawn of the information age. The amount of data rapidly increases daily (Wu, Zhu, Wu, & Ding, 2014) and data can be found and transferred through the massive proliferation of smart devices, the World Wide Web, remote satellite systems and telecommunications. Data collection can be performed in different domains, such as social networking applications, industry, commercial, medical and text resources. These domains offer massive amounts of data. Thus, the gap between data generation and its understanding continues to increase (Abdel-Basset, Mohamed, Smarandache & Chang, 2018; Fan & Bifet, 2013). Understanding the hidden value of this explosive growth of data is urgently required to determine its specific characteristics. Moreover, different types of techniques and methods must be developed to transform these data into helpful information and knowledge (Gamarra, Guerrero & Montero, 2016).

Data mining (DM) utilizes artificial intelligence, statistics and machine learning power to analyse massive amounts of data in order to discover knowledge and information from data for further use (Fürnkranz, Gamberger, & Lavrač, 2012; Kesavaraj & Sukumaran, 2013; Sarkar, Sana, & Chaudhuri, 2012; I. Witten, Eibe, Mark, & Christopher, 2016). In principle, DM can be effective with any type of data, such as relational databases, data warehouses, time-series data, web data or text

databases. However, the challenges and methods for extracting knowledge may differ from each other, thereby making DM diverse and distinct. Common DM tasks can be categorised into classification, regression, clustering and association (Gavrilovski et al., 2016; Mukhopadhyay, Maulik, Bandyopadhyay & Coello, 2014).

Classification is an important DM task with broad applications in many fields. Examples of classification applications include medical diagnoses, spam email and intrusion detection and bankruptcy. The classification task has different techniques, such as rules-based classification, decision tree, linear models, nonlinear models, probabilistic and lazy evaluation, to represent knowledge (Gavrilovski et al., 2016; Ghahramani, 2015; Lotte, Congedo, Anatole, Lamarche & Arnaldi, 2007; Romero, Ventura, Espejo & Hervás, 2008; Wu et al., 2008). In many application domains (i.e., medical diagnosis, protein function prediction and credit approval) the comprehensibility of the model is important (Otero, Freitas & Johnson, 2013). For example, the other techniques aim to introduce high classification accuracy and ignore the degree of human understandability, thereby causing numerous problems in interpreting and explaining relationships between features. An increase in the understandability of the model will help to find hidden discrimination behind the data. On the other hand, classification rules have high interpretation, thus explaining the relationships between features, and can be easily applied in an expert system (Durgadevi & Kalpana, 2017; Katsis, Goletsis, Boufounou, Stylios & Koumanakos, 2012; Wahid & Al-Mazini, 2018; Wu, 2008). The classification accuracy of the rules-based classifier and its size (rule comprehensibility) are the two (2) evaluation measurements for classifier quality. Moreover, rules-based classification (RBC) is

the easiest decision making technique given its comprehensibility and outstanding performance (Cepukenas, Lin & Sleeman, 2015; He, Long & Chen, 2007; Holzinger et al., 2017; Mohammad, Thabtah & McCluskey, 2014). From this perspective, RBC is superior to other methods in terms of minimising the effort of knowledge achievement which requires significant expertise to function effectively. However, an issue that frequently arises in RBC is the overfitting problem with extremely complex classification rules (Battiti & Brunato, 2014; Liu, 2015).

Overfitting is a common problem in DM classification tasks and occurs when many terms are added to the rule. The rule has a perfect fit (high classification accuracy) for specific data instances from which they are generated but generalising the rule to different instances in the dataset is inapplicable. Overfitting may increase computational cost and affect the accuracy rate of the prediction rule in hidden instances (Liu, 2015; Murthy & Meenakshi, 2015). In addition, any excess of pruning in the rule may lead to a very simple rule that does not have the ability to capture the underlying structure of the data. This problem is known as underfitting. The rule will not be suitable and lead to poor predictive performance on the data. Therefore, the ability of the RBC classifier to find global optimal classification rules depends on its capacity to find a good balance between the overfitting and underfitting. Accordingly, the optimal balance between overfitting and underfitting is the prime challenge faced by any RBC. Thus, rule pruning is an important technique to avert overfitting and underfitting problems based on the measurement of comprehensibility and accuracy of the classification rules.

In terms of complexity, the rule-learning process will be affected by computational complexity when handling large and complex data being classified. In computational complexity theory, problems can be classified into the following two types: polynomial problems called P and non-deterministic polynomial (NP) problems. The former refers to problems that can be solved in polynomial time, whereas the latter represents problems that cannot be solved in polynomial time. A problem as difficult as any problem in the non-polynomial class is called NP-hard. Algorithms can be classified into exact and approximate to solve combinatorial optimisation (CO) problems. Exact algorithms have failed in the majority of NP-hard problems because they cannot identify optimal solutions in polynomial time. Thus, the optimal solutions are replaced with an improved solution through approximate algorithms because no exact algorithms can find the optimal solution in a limited amount of time (Dorigo & Stützle, 2004). In fact, the approximate algorithms can be categorised into two major classes, namely, heuristic and metaheuristic algorithms. The former simply means to discover or find and aim to search for a favourable solution for all instances of the problem. Heuristic algorithms are classified into two types, namely, constructive and local search. Constructive algorithms build a solution step-by-step from scratch and are considered the fastest among approximate algorithms. However, the solutions obtained through these algorithms are frequently of low quality and, in certain cases, are inferior to those obtained through local search algorithms. The local search algorithm begins with a suitable solution which is improved iteratively until no accessible improvement can be obtained. However, its defects include stopping at poor-quality local optima and being dependent on initial solutions. The disadvantages of heuristic algorithms have led to new general approaches called

metaheuristic algorithms which aim to bypass the above-mentioned complexity problems. Correspondingly, a metaheuristic algorithm guides the heuristic using different intelligent strategies and concepts to explore a large search space and accept solutions that are worse than previous iterations (Beheshti & Shamsuddin, 2013). The drawbacks of metaheuristic algorithms are presented as follows: i) The approach does not move deep into the search space quickly; and, ii) it requires long computational time. The hybridisation of the advantages of a metaheuristic algorithm over local search is used to solve these problems. The metaheuristic aspect is used as a tool to find favourable solutions from the search space, whereas local search is used as a technique for improving this solution.

In addition, classification problems can be viewed as optimisation problems which aim to identify the optimal model that represents the classification relationships in the data (Otero & Freitas, 2013; Otero, Freitas & Johnson, 2012). Accordingly, metaheuristic algorithms are extensively applied to DM classification, such as ant colony optimisation (ACO), Tabu search (TS), particle swarm optimisation (PSO), simulated annealing (SA) and genetic algorithm (GA) (Asadi & Shahrabi, 2016). Most metaheuristic algorithms are inspired by natural phenomena, such as biological–physical systems or swarms (Yang, 2010). The field of swarm intelligence has been derived from observing swarms' behaviour of real ant colonies, flocking birds, schools of fish or any other insects (Sakthipriya & Kalaipriyan, 2015; Selvi & Umarani, 2010).

In recent years, the swarm intelligence paradigm has received extensive attention in DM, mainly the ACO algorithm (Nayar, Ahuja, & Jain, 2019). Furthermore, ACO which emulates the foraging behaviour of ant colonies in their search for food (Liao, Stützle, Marco & Dorigo, 2013; Martens, Baesens & Fawcett, 2011) is a successful swarm intelligence metaheuristic search algorithm for optimisation problems and DM. This algorithm uses a stochastic aspect and an iterative adaptation procedure based on positive feedback, thereby helping in finding effective solutions for various optimisation problems (Dorigo & Stützle, 2004).

ACO was created by Dorigo in early 1991 in accordance with the observation of ant colony behaviour in searching for nest food (Dorigo & Stützle, 2004). The ant looks for a food source randomly and drops its own pheromone trail whenever it walks. Pheromones act as an indirect communication method between ants. Consequently, with other ants looking for food, overall paths are affected by the amount of pheromone laid by these ants. The pheromone density increases along the route that is visited by numerous ants (short route). Conversely, the pheromone substance in the long path evaporates and decreases over time. Therefore, the algorithm adapts the concepts of individual cooperation and the stochastic policy based on local information (Negulescu, Negulescu & Dzitac, 2017).

In terms of data classification, ACO has been confirmed to be an effective technique for DM classification and knowledge discovery (Salama, Abdelbar, Otero, & Freitas, 2013; Uthayakumar, Vengattaraman, & Dhavachelvan, 2017). Experiments have shown that ant mining is a superior rule induction algorithm that exhibits

performance similar to other classifiers and performs better in various application domains (Arif-Ul-Islam & Ripon, 2019; Liang et al., 2016; Uthayakumar et al., 2017). The ACO algorithm for RBC (Ant-Miner) introduced by Parpinelli, Lopes and Afreitas (2002) is inspired by the foraging behaviour of a real ant colony. Ant-Miner is a metaheuristic, swarm-based, stochastic and separate-and-conquer approach. This approach consists of three major stages, namely, construction rule, pruning rule and updating pheromone (Yang, Li, Zhang, & Ke, 2016).

In the construction rule stage, each ant begins to add terms to be included in the rule. The term represents a particular (attribute, value) duo from the attribute in the dataset, and each pair can be used only once under the construction rule. The ant adds terms that increase the classification accuracy based on its pheromone concentration and amount of information. In addition, Lopez-Ibnez, Stutzle and Dorigo (2016) emphasise the inclusion of local search as the main component of any ACO algorithm. As an ACO variant, the Ant-Miner faces the local optimisation problem; that is, the searching strategy is restricted in global search, thus producing a vulnerable rule set (Liu, 2014; Saian & Ku-Mahamud, 2011). Therefore, many studies (Guan & Lin, 2016; Lin, Dai, Contreras & Zhang, 2017; Mavrovouniotis, Müller, Yang & Yang, 2016) have confirmed that the integration between ACO and local search is compulsory, rather than optional. Nevertheless, local optimisers frequently suffer from an initialisation problem where the main idea of including local search is to allow the ACO algorithm to initialise its search and improve the generated solutions through an aggressive search in the neighbourhood structure of the solutions. The performance of a local optimiser is frequently a function of the

initial solution. Initial solutions are a poor option because the local search procedure focuses significantly on improving the initial low-quality solution (Dorigo & Stützle, 2010). However, the local search is rarely used with ant-mining.

In rule pruning, the overfitting problem can be avoided by decreasing the length of the constructed rules and increasing their simplicity. The procedure removes one term at a time whilst enhancing quality. The pruning repeats until no more improvement exists. In pheromone update, there are two main stages, that of updating the pheromone amount for all terms in the current rule on the basis of its quality and updating all terms that do not appear in the current rule (Parpinelli et al., 2002a).

1.1 Problem Statement

The importance of interpretation in data classification arises whenever knowledge discovery is used in supporting decisions made by humans. Increasing the understandability of the classification model will allow us to find hidden discrimination behind the data (Holzinger et al., 2017). The Ant-Miner is a prominent ACO-based rule classification framework with high interpretation ability to find the relationships between features and outstanding performance in a simple manner (Holzinger, Palade, Rabadan, & Holzinger, 2014; Salama et al., 2011a). Nevertheless, it has several drawbacks that may reduce classification accuracy.

In the construction stage of the Ant-Miner, the fixed selection of threshold (i.e., pre-pruning) is insufficient to overcome the over- and under-fitting of data as it is critical

and highly dependent on data. The threshold criteria in the literature vary from each other (Chan & Freitas, 2006a; Holden & Freitas, 2008; Thangavel & Jaganathan, 2007; Tripathy, Hota, & Satapathy, 2013). In such cases, practitioners can be confused concerning selecting an appropriate threshold for a specific dataset. Consequently, if a high value is chosen, the constructed rule will under-fit the data. Otherwise, it will not overcome the overfitting of data. Equally important, is that the predefined threshold does not take any consideration of the different stages of search spaces. Therefore, the pre-pruning technique can be improved in such a way that threshold value is automatically adapted rather than kept constant.

In Ant-Mining, post-pruning suffers from the problem of nesting effect origins from the method of greedy Sequential Backward Selection (SBS) in feature selection. The pruning starts from a complete set of terms and removes one term at a time with no ability to refresh the eliminated terms, thereby depriving the opportunity of obtaining a good pruned rule by adding/removing the terms during the pruning process (Abdoos, Mianaei, & Ghadikolaei, 2016; Venkatesh & Anuradha, 2019; Wah, Ibrahim, Hamid, Abdul-Rahman, & Fong, 2018). To overcome such a drawback of the post-pruning technique, a more flexible technique can be implemented to add / remove the terms during the pruning process.

The Ant-Miner classifier also suffers from premature exploitation because of the absence of any local search in its structure (Liu, 2014; Saian & Ku-Mahamud, 2011, 2012). In such cases, Ant-Miner search strategy is restricted in global search, thus producing a vulnerable rule set. Therefore, the Ant-Miner was not designed to

explore the neighbourhoods of the current rule and does not consume more time in improving it iteratively. The neighbourhood structures are not fully covered. In this way, this type of search is over-explorative because it is either a single neighbourhood structure movement as exemplified in Saian and Ku-Mahamud (2012), or it does not profit from local search at all (Martens et al., 2011). Therefore, various neighbourhood structures can be developed to catapult the search to another point which gives the possibility of completely exploiting the neighbourhood.

1.2 Research Questions

This research aims to answer the following questions:

1 How can the pre-pruning threshold value be varied on-the-fly based on the need of the Ant-Miner classifier?

2 How can the greedy post-pruning method be improved?

3 Can local search technique solve the problem of the premature exploitation in Ant-Miner classifier?

4 Will the proposed adaptive ACO classifier produce better results?

1.3 Research Objectives

The main objective of this research is to develop an adaptive ACO algorithm for RBC (Ant-Miner) that balance between classification accuracy and model complexity. The specific objectives of this research are presented as follows:

1 To develop an adaptive pre-pruning technique that has the ability to select the appropriate threshold value based on the quality of the solution.

2 To develop a genetic-based post-pruning technique for inclusion/removal of terms in post-pruning process.

3 To propose a hybridise Iterated Local Search (ILS) with the Ant-Miner classifier algorithm.

4 To evaluate the performance of the proposed adaptive ACO classifier.

1.4 Significance of the Research

The proposed adaptive threshold control method can be considered as a new study and a new variant of the Ant-Miner algorithm. In addition, the proposed post-pruning method overcomes the limitations and drawbacks of the exiting post-pruning procedure by flexibly adding and removing terms during the pruning process. These methods are a contribution to knowledge in the area of DM classification tasks. The hybridisation between the Ant-Miner classifier and ILS algorithms aims to enhance the performance of the classifier. Furthermore, the proposed mechanism that combines pre-pruning and post-pruning techniques aims to include the good features of both techniques and integrate them into the Ant-Miner. Thus, this research is significant in providing a classification rule list that is entirely simple and accurate.

In particular, the proposed classification algorithm can be used in real applications and allows experts to review the result. In fact, the algorithm significantly provides a deep insight into the relationship between the key factors that lead to predicting the class labels that may previously have been concealed. For example, in disease diagnoses, several variables affect illnesses. Therefore, finding the specific variables is important and can assist in the control of disease. This classifier algorithm will be

the future direction for different domains with an overall goal to extract information from the data. These domains may include business, economics classification, library classification, biological classification, medical classification, health care systems, education systems and manufacturing engineering.

1.5 Scope of the Research

This research has focused on a prominent variant of the ACO algorithm for RBC algorithm called Ant-Miner. The classifier algorithm aims to generate classification rules from the data using some DM concepts, principles and the search behaviour of ACO. Thus, this research has focused on the following three research directions to improve the performance of Ant-Miner: (1) use an online adaptation method to select the appropriate terms to be included in the rule; (2) evolve a post-pruning technique with the flexibility to add/remove terms during pruning; and, (3) hybridise the ILS and Ant-Miner classifier algorithms. Consequently, these directions will enhance the classification performance of the Ant-Miner algorithm. In addition, there are two types of classification based on the number of prediction classes; single-label classification and multi-label classification. This research has focused on the classical classification task which is the single-label classification.

All experiments in Chapter Five have been conducted using 16 datasets obtained from the University of California Irvine machine learning repository (UCI). The datasets vary in terms of field, instances, attributes and classes. The experiments have been performed using a 10-fold cross-validation procedure. The final result of

the classification accuracy and the simplicity (number of rules and number of terms per rule) have been used to indicate the performance of the proposed classifier.

1.6 Thesis Organisation

This thesis is divided into six chapters. Chapter One introduces the background information related to the problem that this research has solved. Chapter Two covers reviews on various classification tasks and its techniques to represent knowledge, classification using rule induction, Ant-Miner classifier and its variants, hybridisation with local search, rule pruning techniques, and parameter control. Chapter Three addresses the research framework, methods, techniques and experimental procedures used in conducting this research. Chapter Four presents three new classification algorithms, namely, A-AntMiner, GA-AntMiner and ILS-AntMiner. These classifiers are implemented to overcome the limitation of the Ant-Miner classifier and the classifiers are integrated into AntMiner to form the AGI-AntMiner algorithm for rules-based classification. Chapter Five compares the performance of the proposed classifiers with other hybrid ant-mining classifiers and state-of-the-art RBC algorithms. Chapter Six presents the conclusions on the classification algorithms and emphasises the contributions of this research. In addition, Chapter Six provides some recommendations as guidelines for future works on ant-mining classification algorithms.

CHAPTER TWO
LITERATURE REVIEW

2.1 Introduction

In this chapter, reviews of the previous studies on ACO in RBC are presented. The focus of the reviews is on the main issues relevant to efficacy of ACO in building classification models. The chapter also draws attention to the research directions to make the ACO more appropriate in constructing a classification algorithm. Section 2.2 presents the data classification and its common techniques. Section 2.3 explains several existing classification algorithms related to rules-based classification. Section 2.4 focuses on the algorithmic components and variants of the Ant-Miner classification algorithm. Section 2.5 presents the hybridisation with local search methods. Section 2.6 discusses the related work on rule pruning as an important part of rules-based classification methods. Section 2.7 presents the previous studies on parameterisation as an autonomous search inside the search algorithm. Section 2.8 summarises this chapter.

2.2 Classification Technique

Data classification is the most important DM task with broad applications in nearly every field. Examples of classification include medical diagnosis and detection of spam emails, intrusion and bankruptcy. Data classification is considered a supervised learning approach of classifying each item in a set cluster to one category or class based on its characteristics. Classification is regarded as a NP-hard problem due to the large search space of different data sources collected from mobile apps, medical

agencies, video and photo systems, networks and online forum platforms (Hota, Satapathy & Jagadev, 2015; Jiang, Yang, Yan & Miao, 2016; Tripathy et al., 2013). For example, environmental scientists use data classification in satellite technology to monitor oil slicks from images for providing early warnings in different weather conditions. In the diagnosis domain, which is a principal application area of expert systems, data classification is used to detect diseases at the early stage (Kubat, Holte & Matwin, 1998; Liu et al., 2014; Waal, 2017). Countless applications exist but medical diagnosis and remote sensing domains present the importance of data classification.

The classification process consists of two procedures: training and testing. Training builds the classification model based on a part of the dataset called training data. Testing indicates the performance of the classifier based on other sections of the testing dataset, which are never seen during training (Farid, Zhang, Rahman, Hossain & Strachan, 2014). Classification model (classifier) performance is measured in terms of accuracy. The classifier predicts the target class for each instance. If the prediction is successful, then the result will be counted as correct. Otherwise, the result will be counted as an error. The overall accuracy is measured based on the number of correctly and incorrectly predicted classes over all instances in the dataset (Patil & Sherekar, 2013). Data classification can be presented in different ways depending on the representation of knowledge by using machine learning, statistics and artificial intelligence (Ghahramani, 2015). The taxonomy of common techniques in data classification based on knowledge representation includes linear models,

decision tree (DT), rules-based methods, non-linear models, lazy evaluation and probabilistic approaches (refer Figure 2.1) (Gavrilovski et al., 2016).

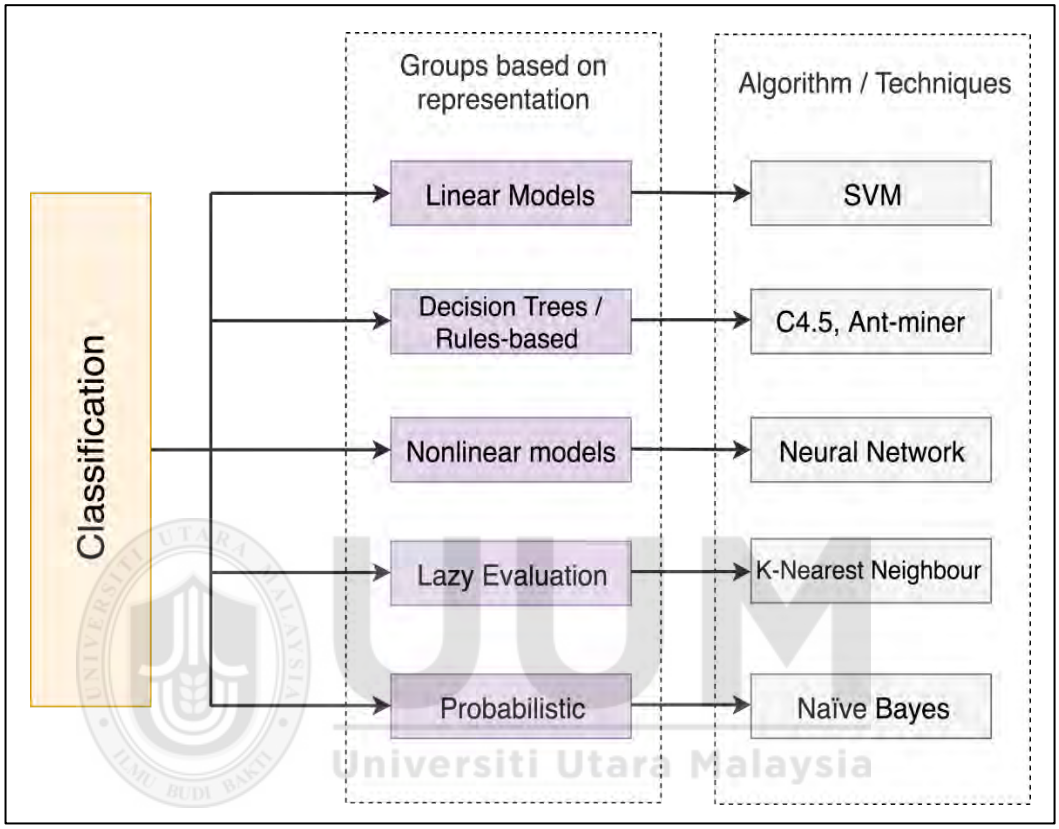

Figure 2.1. Taxonomy of DM Classification Task *(Gavrilovski et al., 2016)*

The techniques and algorithms provide an overview of the fundamental concepts of the supervised techniques mostly used for classification (Han & Kamber, 2011), such as DT classifier (Azar & El-Metwally, 2013), Artificial neural networks (ANN) technique (Lam et al., 2014), Support vector machine (SVM) (Danenas & Garsva, 2015), K-nearest neighbour (KNN) classifiers (Kanj, Abdallah, Denœux & Tout, 2016; Liao & Vemuri, 2002) and rules-based classifier (Liu, Gegov & Stahl, 2015). One important related concept is Wolpert and Macready's 'no free lunch' theorem,

which states that the overall performances of different classification algorithms are equivalent (Ali & Smith, 2006; Li, Li & Wang, 2015). This theorem implies that no algorithm is capable of always achieving the best performance result for all datasets (Datta & Saha, 2016).

The SVM is a binary classifier that applies to different domains (Azar & El-Said, 2014; Mangasarian, 2003) and can classify data instances by separating the classes (Kong & Zhang, 2008). The SVM's basic ideas are in accordance with the concept of a hyper-plan, which defines the decision boundaries of the dataset. The hyper-plan separates dataset instances that have different class attributes (Harvey, 2009). The kernel functions are used to disjoin the instances linearly. Fundamentally, the hyper-plan is used to classify instances on the basis of the margin (i.e., the amount of space) it creates between the classes by reducing the balance between the difficulty caused by classification and the experimental mistakes to avoid overfitting. If the margin is large, then the chance of misclassifications is few. Figure 2.2 shows the basic structure of the SVM classifier. Many kernel functions are available, such as linear function and RBF. The latter is common due to its good performance in different fields (Moustakidis & Theocharis, 2010). The SVM has been expanded to handle multiclass datasets by dividing multiclass problems into binary problems with their own classifiers (Weston & Watkins, 1998).

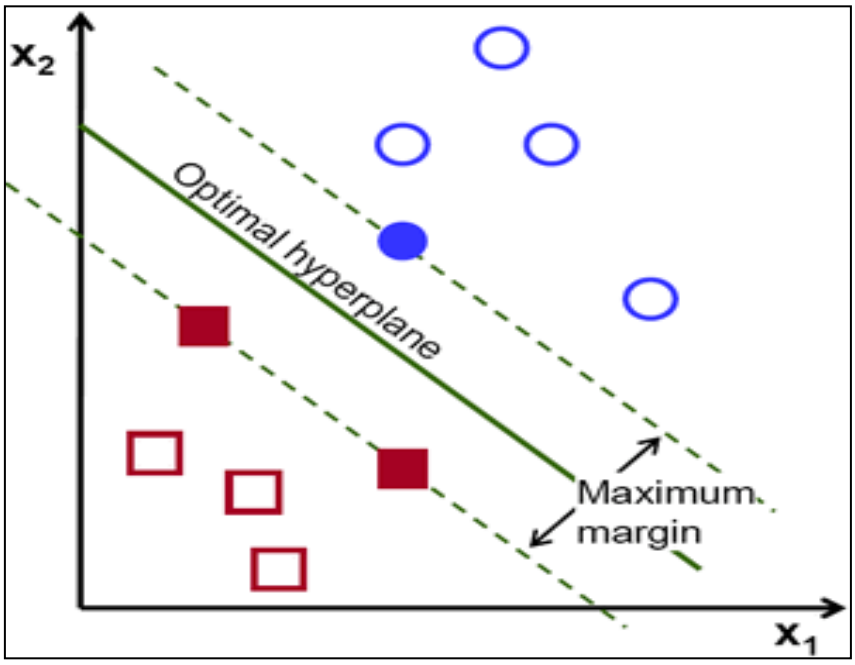

Figure 2.2. Basic Structure of the SVM Classifier (Azar & El-Said, 2014)

The DT classifier is commonly used in classification domains. This classifier uses the divide-and-conquer concept to build the classification model (Rokach & Maimon, 2005). The learning model can be built into different algorithms that split the training dataset into branch-like segments. The DT algorithm selects an attribute to be the root of the tree, makes one branch for all possible values and then repeatedly iterates this process for each branch (Peng, Chen & Zhou, 2009). Conventional DT algorithms include chi-squared automatic interaction detector (CHID), classification and regression tree (CART), iterative dichotomiser 3 (ID3), C4.5 and C5.0 (Strecht, 2015). The most basic difference amongst the algorithms and their usages can be briefly defined as follows: ID3 cannot handle continuous attributes or missing values in the dataset, and it is time-consuming compared with other DT algorithms. Therefore, ID3 does not handle overfitting when no pruning procedure exists in its components. C4.5 and C5.0 can work with continuous and

categorical attributes. Both algorithms use a pre-pruning procedure and are considered faster than ID3. However, only the C5.0 algorithm can handle missing values in the dataset because C4.5 is limited. Similarly, CART works with continuous and categorical attributes. This algorithm handles the missing value in the attribute and uses a post-pruning procedure, but it is not faster than C4.5 and C5.0 (Hssina, Merbouha, Ezzikouri & Erritali, 2014; Lavanya & Rani, 2011).

The ANN is a classification technique based on models of the brain. The main idea is to let the brain learn, and then things are transmitted via neurons (Maojo & Sanandres, 2000). This method considers one of the well-classified techniques in different real-world problems (Amato et al., 2013; Tkáč & Verner, 2015). The ANN consists of input, output and hidden layers and connection weight, activation function and summation node (Yadav & Chandel, 2013). In classification tasks, the ANN can learn from training data. The knowledge, which is represented as neurons, spreads in one or more layers of the network architecture. The main objective of the ANN is to transmit the activation values from the input to the significant output. The ANN consists of a set of rules to define the mechanism of the number of hidden layers, transmission weighing, connection pattern and node activation. The variation settings of these rules will produce the ANN variations (Almana, Aksoy & Alzahrani, 2014). An example of an ANN in classification data is the classical and multilayer perceptron (Kazi & Mubarak, 2014). Figure 2.3 displays the general structure of a neural network with one (1) hidden layer.

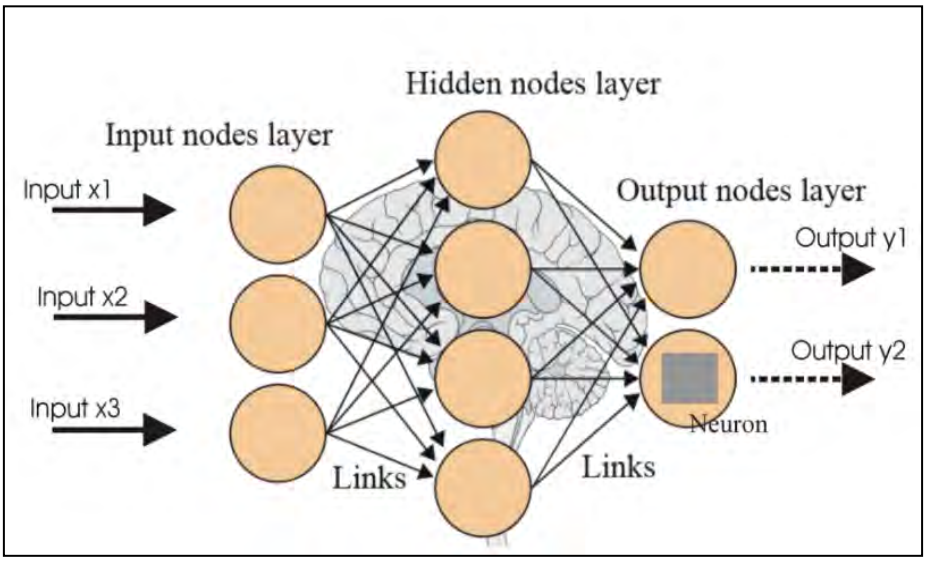

Figure 2.3. ANN Multilayer Perceptron (Jain, Mao & Mohiuddin, 1996)

The KNN classifier is a traditional approach in the DM classification task (Kanj, Abdallah, Denœux & Tout, 2015). It is based on learning by analogy. This classifier represents the instance as a point in an n-dimensional space. All training instances are stored in a dimensional numeric attribute. In this way, the class label of the unknown instance is assigned by searching the pattern space for the KNN instances from the training dataset (Xu et al., 2013). The nearest instances can be measured by using the Euclidean distance between points. Commonly, the unseen instances are assigned to the most common class in the dataset. In addition, the KNN classifier is considered to be a lazy learner because it stores all training instances and does not build a classification model until some instances are classified. This feature contradicts other classification methods, such as ANN and DT, that build a classification model before receiving unknown instances (Gillis & Morsi, 2016).

However, the above-mentioned techniques have drawbacks and limitations. For instance, the ANN hardly uncovers patterns in an easily understandable manner. In addition, this technique requires numerous instances for learning, can hardly determine the number of necessary layers and neurons and has time-consuming calculations (Au, Chan & Yao, 2003; Kotsiantis, 2007; Kumar, Aggarwal & Sharma, 2015; Lazarov & Capota, 2007; Olden & Jackson, 2002). Meanwhile, the KNN is expensive because of the required large computational costs, which will be worsened by the large number of neighbours, compared with unlabelled instances. Therefore, the KNN requires efficient indexing techniques. In addition, noisy or irrelevant attributes may be present in the data, and the KNN assigns equal weight to each attribute. Consequently, the classification results may cause confusion (Phyu, 2009). DT algorithms are inefficient for non-linear and complex relationships between attributes (Almana et al., 2014) and demonstrate poor performance when the training dataset is small. The SVM has a low training speed when selecting efficient kernel functions for given classification problems. This method needs practice and validates the performance over the training dataset to find the best function. The selection of the best kernel setting can be similar to determining an appropriate number of hidden layers in the ANN (Kotsiantis, 2007). In addition, the kernel function fails when the number of attributes is greater than the number of instances in the dataset (Alwan, 2013). Furthermore, the SVM is extremely sensitive to noise and discrete data, which can dramatically decrease the performance.

In summary, these techniques have a common limitation i.e they build the classification models as a black box. In addition, they aim to introduce high

prediction accuracy but neglect the degree of human comprehensibility. Consequently, problems in interpretation and explanation of the relationships amongst the features may exist. However, increasing the understandability of the model will allow the determination of the hidden discrimination behind the data. Nevertheless, rules-based classification is considered the easiest decision making method due to its simplicity and outstanding performance (Cepukenas et al., 2015; He et al., 2007; Holzinger et al., 2017; Mohammad et al., 2014).

2.3 Rules-based Classification

Rules-based classification algorithms are common supervised machine learning techniques (Wu et al., 2008). The classification is introduced in a different formalism consisting of a list of prediction rules in the form of

If <term1> and <term2> and ….. then <target class>.

Rules-based classification algorithms have different main groups of rule learning algorithms (AlMana & Aksoy, 2014; Chennupati, 2014; Elgibreen & Aksoy, 2013; Fürnkranz, Gamberger & Lavrač, 2012; Yildiz, 2013). The first type is known as a divide-and-conquer approach, and various algorithms have been proposed to convert different DT classification algorithms into a group of rules, such as C4.5, CART and ID3. These algorithms initially create and then transfer a DT to a list of rules by converting each path from the root to a leaf as a rule.

The second type is known as the separate-and-conquer approach, which has the ability to generate if–then rules directly from a given dataset by using various rules-based classifiers, such as RIPPER, PART and ant-mining algorithms. The PART

algorithm was introduced by Frank and Witten (1998). In each iteration, this algorithm builds a partial DT by using C4.5 and converts the best leaf that has the best coverage into a rule before discarding the tree (Datta & Saha, 2016; Frank & Witten, 1998).

The RIPPER algorithm (also known as JRip) was implemented in 1995 by Cohen as an optimised version of an IREP algorithm. The algorithm builds rules by adding terms based on the heuristic in accordance with the minimum description length (MDL) criteria until no negative instance is covered by the rule. The algorithm will repeatedly add construction rules to the list until all positive instances are covered (Cohen, 1995; Seerat & Qamar, 2015).

The Fuzzy Unordered Rule Induction Algorithm (FURIA) is a modification variant of the RIPPER classifier, which uses fuzzy instead of conventional rules. This classifier introduces a simple and comprehensible classification model. Moreover, FURIA utilises an efficient rule-stretching method to handle uncovered examples. Experimental results show that the performance of the FURIA classifier outperforms that of the original RIPPER classifier (Hühn & Hüllermeier, 2009).

The conjunctive rule classification algorithm is a single rules-based algorithm that can predict discrete and continuous class labels. The general output of this classifier is a classification model that captures all generalizable knowledge within the training data. The construction rule consists of two parts: antecedent (‘AND’) and the consequent (class value). The antecedent’s part of the rule is selected using the

information gain of each antecedent, whereas the consequent part is selected using the distribution of the available classes in the dataset. Then, the construction rule prunes by using a pre-pruning technique based on the number of antecedents or the reduced error pruning procedure that computes the weighted average of the accuracy rates of the pruning data (Fauzi bin Othman & Moh Shan Yau, 2007).

The decision table is a machine learning classification algorithm that summarises the instances in the dataset by using the decision tables that consist of the same number of features in the dataset. Therefore, to classify the instances, the classifier will find the row in the decision table that meets the new instances. This algorithm utilises the wrapper method used in feature selection to identify good attribute combinations for the classification table. In this way, the classifier can introduce small decision tables, as well as overcome the problem of data overfitting (Freitas, 2014).

Other classifiers learn rules from the one-attribute-rule classification algorithm, which is a simple and accurate classification algorithm that generates one classification rule for each attribute in the dataset. This classifier uses the minimum-error attribute for predicting a specific class. Then, the greedy algorithm selects the best rule that has the lowest error rate. Additionally, if the classifier has discovered more than one rule that has the same error rate, then it will randomly select one (Ali & Smith, 2006).

The existing rule induction algorithms are typically have the greedy characteristic. They may only determine local optimal rather than global optimal classification

rules. In contrast, ant-mining algorithms can mitigate this drawback by using a combination of two basic ideas. Firstly, these algorithms have a stochastic aspect, which helps them to explore a large area of search space. Secondly, they use an iterative adaptation procedure based on positive feedback (Freitas & Parpinelli, 2008). Ant-mining algorithms are rules-based classification that follows a separate-and-conquer strategy to generate rules from the data (conquer) and then remove the covered instances (separate). The algorithms consist of three major steps: sequential rule construction, rule pruning and pheromone update. Experiments have shown that ant-mining algorithms achieved similar performance to and even outperformed other classification techniques in various application domains (Agravat et al., 2010; Al-Abadi, 2017; Arif-Ul-Islam & Ripon, 2019; Baig & Shahzad, 2012; Chorbev, Mihajlov & Jolevski, 2009; Durgadevi & Kalpana, 2018; Lai et al., 2016; Martens, Baesens & Fawcett, 2011; Mashhour et al., 2018; Ramalingam & Sujatha, 2018; Sabri & Saian, 2017; Salama, Abdelbar & Freitas, 2011; Salama, Abdelbar, Otero & Freitas, 2013; Uthayakumar et al., 2017; Wu, 2008).

2.4 Ant-Mining Classifiers

ACO is a nature-inspired metaheuristic algorithm proposed to solve hard CO problems. The inspiration of ACO is stigmergy, which follows real ants' behaviour by using indirect communication via the ants' pheromone deposits in the environment. The pheromone consists of numerical information, which increases the probability and guides other ants in the colony to construct solutions for the problem (Dorigo, Birattari, & Stützle, 2006; Falaghi & Haghifam, 2007). ACO has largest use in different NP-hard CO problems; for example, network routing, scheduling and DM

(Chang, Chang, & Lin, 2009; Dorigo et al., 2006). This algorithm has become an effective method for extracting useful and interesting classification rules from data through the Ant-Miner classification algorithm. Artificial ants can find solutions to large and complex search spaces in the context of rule discovery. The strategy of searching space combines exploration and exploitation (Sagban, 2016).

The general outline of the ant-mining classifier (Figure 2.4) includes algorithmic procedures that govern the search or building strategy for a classification model (López-Ibáñez, Stützle, & Dorigo, 2016). Each procedure includes different phases, depending on the implemented classifier. These differences involve specific building blocks that interact with each other; such blocks are referred to as algorithmic components. On the basis of an extensive investigation of existing ant-mining classifier algorithms, a component-wise insight is missing. Nevertheless, determining which amongst the relevant building blocks for the perspective algorithm (e.g., chosen ACO variant, rule construction, pruning choice, pheromone update strategy or parameter setting) contributes to the improvement of the performance is necessary. The subsequent sections discuss a component-wise overview of these ant-based methodologies and the popular ant-mining techniques based on their building blocks. Furthermore, they summarise the key characteristics of such algorithms, regardless of their specific classification domain.

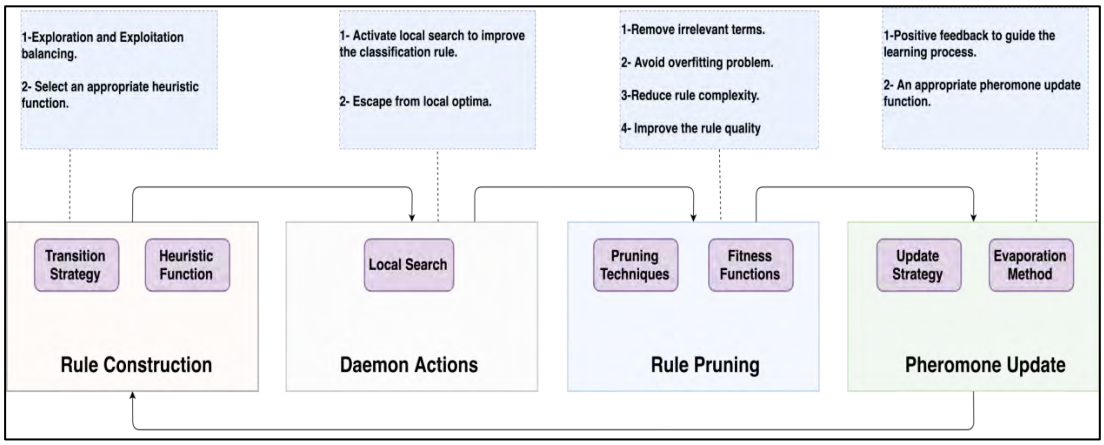

Figure 2.4. ACO Metaheuristic Components for Rule-based Classification

2.4.1 Rule Construction

In Ant-Miner variants, useful information is expressed in two parts: rule antecedent (IF) and rule consequent (THEN). Hence, the knowledge will present in the following form: IF <conditions> THEN <class>. The former contains a set of conditions, which are usually connected by a logical conjunction operator (AND) in the following form: IF term1 AND term2 AND…. Each term contains three factors <attribute, operator, value> (e.g., <Blood pressure = high>). The ant will add one term at a time to the current partial rule constructed. The current partial rule is similar to the current partial path followed by that ant. Meanwhile, the selection of the term to be added to the rule will be based on the transition strategy and on the problem-dependent heuristic function and associated pheromone amount. Local search is optional; several studies have shown that local search is useful in improving the performance of ACO algorithms. The transition rule, heuristic function and pheromone update used in ant-mining variants, as well as other components, will be discussed in detail in the subsequent discussions.

2.4.2 Transition Strategy

The transition rule governs the search strategy to achieve the balance between the exploration of the search and exploitation of the obtained best solutions (Dorigo & Stutzle, 2010; Sakthipriya & Kalaipriyan, 2015). The variants of ant-mining classifiers are developed to improve the exploration/exploitation balance and produce effective classification rules. The traditional ant-mining classifiers are developed on the basis of ant systems (AS). Afterwards, some classifiers use the ant colony system (ACS) and the max–min ant system (MMAS). The majority of algorithms based on the ACO framework utilise the formula introduced by Dorigo and Stützle (2004), which represents the probability of selecting a term to be added to the rule.

The AS was proposed in 1991 as a result of experiments in the foraging behaviour of real ants. This system is used to determine the shortest path in the traveling salesman problem (TSP). The construction graph in this algorithm is fully connected; each city represents a node and the possible paths connect the cities (Dorigo & Colorni, 1996). Ant-mining classifiers are inspired by AS. The AS has two main steps: rule construction and pheromone updates. The transition rules in all ant-mining classifier algorithms are dependent on the AS transition rules, which add terms to the current rule on the basis of the heuristic information and pheromone associated with each term (Chan & Freitas, 2006a; He et al., 2007; Jaganathan, Thangavel, Pethalakshmi & Karnan, 2007; Liang, Lee & Lee, 2011; Liu, Abbass & McKay, 2002; Madhusudhanan, Karnan & Rajivgandhi, 2010; Michelakos Papageorgiou, & Vasilakopoulos, 2010; Parpinelli Lopes, & AFreita, 2002b; Prabha & Balraj, 2014; Salama & Abdelbar, 2010; Salama & Otero, 2013; Shahzad & Baig, 2010; Smaldon,

2006; Thangavel & Jaganathan, 2007; Tripathy et al., 2013). A different strategy is proposed by Jin et al. (2006) and Wu and Sun (2012), whose algorithms aim to avoid the dependence on the initial term due to the random selection of a term at the beginning of the interior loop.

Chan and Freitas (2006b) designed the transition rule for multi-label classification tasks, in which each ant can produce a set of candidate rules equal to the number of the class attributes. In some cases, an ant discovers one rule if and only if it can predict all the class attributes. Each ant selects a term to add to the current rule by using a roulette wheel selection technique (Freitas, 2002). The roulette wheel consists of slots with different sizes that represent the probability of selecting a term on the basis of its heuristic and pheromone amounts, as in AS. The performance of this algorithm was compared with three classifiers: the original Ant-Miner, C5.0 and a simple majority classifier that uses a bio-informatics dataset to predict protein functions. The results show that the proposed algorithm's performance is better than the that of C5.0 and the simple majority classifier, but demonstrated no difference from the original Ant-Miner.

Other ant-mining classifiers propose a new transition rule to handle the continuous attributes proposed by the cAnt-Miner algorithm (Otero et al., 2008; Rajpiplawala & Singh, 2014). Each ant initially selects a continuous attribute. The term will not be represented by three factors (i.e., attribute, operator, value), but of the form (attribute $< v$ and attribute $\geq v$). The selection of the best threshold value follows a dynamic discretisation method, namely, entropy-based discretisation, and is evaluated on the

basis of the cases in the training dataset that are covered by the current rule. The pheromone update procedure is extended to deposit an amount of pheromone that can manage continuous attributes. Another extension is the cAnt-Miner 2 transition rule for handling continuous attributes, as proposed by Otero et al. (2009) by applying two new methods. The first method recursively applies a cAnt-Miner discretisation method and relies on the MDL criterion to accept or reject a threshold value, thereby creating intervals with lower and upper bound values in the following form: lower value <= attribute < upper value (Irani & Fayyad, 1993). The second method emphasises the importance of the attributes' interaction by considering the previously selected vertices in the threshold evaluation. The pheromone concentration is extended to handle continuous attributes. Another extension of the cAnt-Miner algorithm, namely, cAntMinerPB, a new sequential covering strategy for ACO classification algorithms to mitigate the problem of rule interaction, where the order of the rules is implicitly encoded as pheromone values and the search is guided by the quality of a candidate list of rules (Otero, Freitas, & Johnson, 2013b).

In the work of Saian and Ku-Mahamud (2012), each ant discovers one best rule based on the hybrid ACO with the SA algorithm. SA is a method for searching an optimal solution to the local optimisation problems. The SA algorithm depends on a variable, that is, temperature (Aarts, Korst, & Laarhoven, 1997). The proposed hybrid algorithm uses the iteration on the temperature variable by starting with a high value. Therefore, all rules have the same probability to be selected at the beginning. Then, the temperature decreases, and all rules will have a chance to be selected as the best rule. In this way, the algorithm can avoid the local optimisation problem of the

solution. Then, the best rule in the iteration will be selected and the best amongst all iterations will be added to the rule set (Saian & Ku-Mahamud, 2012).

ACS is different from AS in three aspects (Dorigo & Gambardella, 1997). Firstly, ACS uses more random actions to construct the rule than AS. Secondly, the global pheromone updates only the global-best solution. Finally, ACS performs local pheromone updates whilst ants move between nodes to increase the exploration. In Ant-Miner classifier algorithms, an adaptive state transition strategy and the pheromone update the strategy of the ACS used in the four classifiers (Agravat et al., 2010; Fakhar, 2014; Ji, Zhang, Liu, & Zhong, 2006; Liu, Abbass, & Mckay, 2004; Zhang & Sun, 2016) for the term selection to predict a classification rule that aims to provide an explicit control over the exploitation and exploration balance.

The enhanced Ant-Miner algorithm uses a new pheromone update mechanism and mutation strategy (Ji et al., 2006). The pheromone updates on the basis of a rule-punishing operator and self-adapting evaporation, whereas the mutation strategy mutates the value of one attribute node at a time and measures the rule quality; if the quality improves, then the mutation operation proceeds. Another ACS state transition rule uses the same ACS state transition rule with a new parameter β, namely, relative importance, to trade-off and control the trail versus visibility in the probability rule (Jiang, Xu, & Xu, 2005; Wang & Feng, 2004).

The first algorithm, developed on the basis of the better performing MMAS (Stutzle & Hoos, 1997, 2000), was AntMiner+ (Martens et al., 2007). Firstly, this algorithm

uses the MMAS concept as follows: it utilises the best solution during the iteration, and the pheromone update is only applied to the best ant with two update possibilities. The pheromone update can be an iteration of the best or global-best ant. Secondly, AntMiner+ uses the pheromone limitation boundaries within $[\tau_{min,}\tau_{max}]$ to avoid stagnation. Thirdly, the pheromone is initialised with $[\tau_{max}]$ to increase the exploration of the search space in the early stage. AntMiner+ algorithm modifies the original Ant-Miner by defining the environment as directed on an acyclic graph, and all the other algorithms are in a fully connected graph. This mechanism decreases the selection amongst all nodes in all previous algorithms to select the nodes in one variable. Moreover, AntMiner+ has a new heuristic function, new pheromone update strategy and new self-adopting mechanism to weight the α and β parameters. Furthermore, this algorithm can work with two types of nominal and ordinal attributes and uses different rule pruning procedures.

However, the most popular transition rule used in the ant-mining variant procedure is based on the original Ant-Miner, which overshadows the other rules (Baig & Shahzad, 2012; Chan & Freitas, 2006a; He et al., 2007; Jin et al., 2006; Liang et al., 2011; Liu et al., 2002; Madhusudhanan et al., 2010; Michelakos et al., 2010; Parpinelli et al., 2002b; Prabha & Balraj, 2014; Salama & Abdelbar, 2010; Salama & Otero, 2013; Shahzad & Baig, 2010; Smaldon, 2006; Thangavel & Jaganathan, 2007; Tripathy et al., 2013; Wu & Sun, 2012). In addition, some modifications on the transition rule have been applied to the original transition rule to handle continuous attributes (Otero et al., 2008, 2009; Rajpiplawala & Singh, 2014), as well as multi-label classification tasks (Chan & Freitas, 2006b). Several studies have used the ACS

transition rule. The proposed transition strategy improves only the exploration strategy. Meanwhile, the original ACS algorithm was introduced to improve the exploration and exploitation strategies. In the original ACS algorithm, a strong elitist strategy was proposed to enhance exploitation (i.e., local and global pheromone updates), which is absent in the ACS-based classification algorithms. Hence, when the search space is excessively large, the imbalance between exploration and exploitation will occur and introduce non-optimal classification rules (Agravat et al., 2010; Fakhar, 2014; Ji et al., 2006; Jiang et al., 2005; Liu et al., 2004; Wang & Feng, 2004). Other studies have utilised the SA transition rule (Saian & Ku-Mahamud, 2012). The algorithm performance obtains improved accuracy rates, but a worsened result in simplicity, wherein the numbers of rules and terms per rule abnormally increased compared with the original Ant-Miner. The algorithm proposed by Martens et al. (2007) still uses the same transition rule but has added a new self-adopting mechanism to weigh the α and β parameters; this mechanism will be discussed in detail in Section 2.7. The subsequent subsections will discuss the heuristic function and the pheromone update used in ant-mining classifiers.

2.4.3 Heuristic Function

The heuristic function measures the terms with probability to be selected and their importance. In addition, this function aims to add terms for each rule to improve the classification accuracy.

The typical heuristic function used in the Ant-Miner and its variant algorithms is inspired from information theory (Cover & Thomas, 2006). The function measures

the amount of information contained in each term (entropy) (Agravat et al., 2010; Chan & Freitas, 2006a, 2006b; Jaganathan et al., 2007; Ji et al., 2006; Liang et al., 2011; Madhusudhanan et al., 2010; Parpinelli et al., 2002b; Saian & Ku-Mahamud, 2011; Thangavel & Jaganathan, 2007; Tripathy et al., 2013; Wu & Sun, 2012). Then, an extra condition is added to select the terms to each candidate rule (Thangavel & Jaganathan, 2007); in this way, a high amount of information that must exceed or equal a pre-determined threshold value (0.6) can be selected. Tripathy et al. (2013) determined a threshold value of 0.7. Such a threshold value not only benefits the selection of a proper term in the rules but also decreases the computational complexity of the rule pruning procedure. This heuristic function is modified in some algorithms. Wu et al. (2012) proposed a minor modification in this function by removing the denominator and varying the heuristic function to simplify the heuristic function by applying a slight modification to the denominator. Both modifications are proposed to resolve the problem of local optima.

Unlike discrete attributes, the entropy of continuous attributes requires a customised heuristic function. Otero et al. (2008) proposed an enhanced heuristic function in cAnt-Miner to handle continuous attributes. This heuristic function incorporates the dynamic entropy-based discretisation procedure to create thresholds on continuous attribute domain values dynamically during the selection of the candidate rule in two intervals: $a_i < v$ and $a_i \geq v$. The function then selects the best partition (i.e., purest) that possesses the maximum number of examples from the same class. This selection is based on the entropy values of the two generated partitions. The partition with low entropy values will be selected. Furthermore, this heuristic function is adopted by

Rajpiplawala and Singh (2014). Salama and Otero (2013) proposed a modification on the discretisation method by selecting the partitions with high relevance for prediction by utilising pre-selected class attributes.

A cAnt-Miner2 heuristic function is used to handle continuous attributes (Otero et al., 2009). This function incorporates the MDL principle (Irani & Fayyad, 1993) and the cAnt-Miner's entropy-based discretisation method to generate multiple discrete intervals. Moreover, intervals are created in the form of $vlower \leq yi < vupper$ by applying the attributes' interaction by considering previously selected vertices in the creation of partitions. Michelakos et al. (2010) proposed a new algorithm that involves a hybrid heuristic function between the cAnt-Miner2 heuristic function and the minimal-redundancy–maximal-relevance proposed by Ding and Peng (2003). This algorithm aims to select the best possible attributes that have minimum redundancy and maximum relevance to be added in the candidate rule. The maximum relevance measures the dependency amongst the attributes. Attributes in the target class with high dependence on one another are added to the candidate rule. The minimum redundancy aims to retain one attribute from the attributes that depend highly on each other in the construction rule; the attributes' reduction should not affect the classification power. Another heuristic function, which was introduced by He et al. (2007) and Liu et al. (2002, 2004), is an easily computable density estimation heuristic function that is vastly used in the Bayesian classification and Lazy Bayes rule.

The Laplace-corrected confidence heuristic function has been used by Fakhar (2014), Salama and Abdelbar (2010) and Smaldon (2006) to select the terms for specific class attributes, wherein ants have previous knowledge about the class attribute of the candidate rule. In addition, the algorithm used by Salama and Otero (2013) adopted two heuristic functions. The first is used to select the rule consequent class on the basis of the occurrences of the class in the training set prior to the rule antecedent. The second is still used in the Laplace-corrected confidence function to build the rule antecedent.

In cAntMiner and AntMiner-C algorithms, the first term in the candidate rule is selected on the basis of the Laplace-corrected confidence function (Baig & Shahzad, 2012; Shahzad & Baig, 2010). The other terms are selected using new heuristic functions. When an ant is on term i and wants to move to term j, the function will consider the compatibility of both terms and the overall importance of term j to the current class. This heuristic function considers the relationship between terms and overall importance and penalises the terms that lead to specific rules. The rule is extremely complex and contains the details of the training set and the failures on the test set (overfitting).

The distance-based heuristic function in the hmAnt-Miner classifier (Prabha & Balraj, 2014) is further suitable for hierarchical multi-label classification structures. This concept is inspired from the CLUS-HMC algorithm and is based on the paradigm of DT induction. The heuristic function of this algorithm produces a set of instances that accept the condition represented by the term.

ACO-Miner (Wang & Feng, 2004) uses the heuristic function produced by the ELEM2 algorithm (An, 2003). This function evaluates attributes on the basis of the degree of their relevance to target class and guides the search to select the most relevant terms for each candidate rule.

In the construction graph proposed by AntMiner+ (Martens et al., 2007), each ant knows the class attribute for the candidate rules and selects the heuristic value for each term on the basis of a specific class. Each term will have heuristic values as many as the number of class attributes. The heuristic function for each term is computed by the fraction of training cases that are correctly covered (described) by this term.

However, the majority of Ant-Miner variations use the original heuristic function introduced by the original Ant-Miner (Agravat et al., 2010; Chan & Freitas, 2006a, 2006b; Jaganathan et al., 2007; Ji et al., 2006; Liang et al., 2011; Madhusudhanan et al., 2010; Parpinelli et al., 2002b; Saian & Ku-Mahamud, 2011; Thangavel & Jaganathan, 2007; Tripathy et al., 2013; Wu & Sun, 2012). Moreover, some heuristic functions have been introduced to handle continuous attributes (Michelakos et al., 2010; Otero et al., 2008, 2009; Rajpiplawala & Singh, 2014; Salama & Otero, 2013). Other variations have utilised simple heuristic functions that are not expensive in terms of computational costs (Baig & Shahzad, 2012; Fakhar, 2014; He et al., 2007; Liu et al., 2002; Liu et al., 2004; Martens et al., 2007; Prabha & Balraj, 2014; Salama & Abdelbar, 2010; Salama & Otero, 2013; Shahzad & Baig, 2010; Smaldon, 2006; Wang & Feng, 2004). However, these simple functions are less accurate than the

heuristic function of the Ant-Miner; the use of pheromones should compensate for this disadvantage. In conclusion, the time complexity of the heuristic function of the Ant-Miner is not a serious problem in the context of the entire algorithm because the heuristic function values for all terms can be computed in linear time with respect to the number of instances and attributes.

2.4.4 Pheromone Update

The main objective of the pheromone is to communicate with, and reinforce, other ants to identify the best set of rules. Ants cooperate by adding information in the search spaces. The pheromone trails depend on the quality of the discovered rule, which increases with the amount of good rules and decreases with the amount of bad ones. This step can be usually performed in Ant-Miner classification algorithms via pheromone update mechanisms. The pheromone update mechanism consists of two main operations: pheromone deposit and pheromone evaporation.

Firstly, the pheromones are initialised with equal amounts of pheromone on the basis of the attribute values in the dataset (Liu et al., 2002; Parpinelli et al., 2002b). AntMiner+ uses high amounts of pheromones to increase exploration at the early stage of the search. This algorithm also uses boundaries [τ min, τ max] to avoid the early stagnation of the search, which has been inspired from MMAS (Martens et al., 2007).

The pheromone concentration increases for all terms occurring in the rule simulating the pheromone deposit along the path determined by each ant. The terms that have

high pheromone concentrations will garner high selection probability. The quality ratio is used to decide the amount of pheromone to be deposited. Following the ACO variant, two types of update in Ant-Miner classification algorithms are available. On one hand, the majority of Ant-Miner variants use the classical AS system to update the pheromone for each ant construct rule. On the other hand, conventional algorithms use the original equation proposed by Parpinelli et al. (2002), which increases the pheromone on the basis of the quality of the construction rule and then decreases the pheromone by normalising all pheromone values.

Other Ant-Miner classification algorithms aim to use the best rule identified during the search. Only one ant will be allowed to deposit pheromone with two possibilities: best (Liang et al., 2011; Martens et al., 2007; Prabha & Balraj, 2014; Rajpiplawala & Singh, 2014; Saian & Ku-Mahamud, 2011; Wu & Sun, 2012) or global-best iteration (Martens et al., 2007). This idea has been adopted from ACS and MMAS. The second mechanism aims to reduce the computation time and utilise the best rule.

Some algorithms use the pheromone update based on the rule quality and a new parameter called ρ to indicate the pheromone evaporation rate and avoid unlimited accumulation, thereby controlling exploration and exploitation (Agravat et al., 2010; Baig & Shahzad, 2012; Fakhar, 2014; Ji et al., 2006; Jiang et al., 2005; Liu et al., 2004; Martens et al., 2007; Shahzad & Baig, 2010; Wang & Feng, 2004).

However, the above-mentioned new pheromone evaporation and self-adaptive parameters are unnecessary in the Ant-Miner classification variants because evaporation is conducted by normalising all pheromone values. However, algorithms

that do not have this type of evaporation mechanism should add an explicit evaporation parameter to avoid the unlimited concentration of pheromone trails.

2.5 Hybridisation with Local Search

CO problems arise in various application domains, such as scheduling problems, routing problems, network routing and data classification. These problems have simple definitions and complex solutions (Dorigo et al., 2006). The algorithms used to solve such problems are classified into exact and approximation algorithms. The exact algorithms always ensure the discovery of the optimal solution. With the large instance size of the problem, the run time often increases abnormally and, in some cases, the exact algorithms produce unsatisfactory solutions for hard optimisation problems. The approximation algorithms aim to introduce a trade-off between solution quality and run time. These algorithms can efficiently determine the optimal (approximate) solution for hard optimisation problems (Stutzle, 1998).

The approximation algorithms can be classified into two groups: heuristic and metaheuristic algorithms. Heuristic algorithms have two categories: constructive and local search algorithms. Constructive algorithms build the solutions from scratch, whereas local search algorithms search initial solutions and iteratively use the neighbourhood search for improvement. The high-level framework and strategy that guide the local search and constructive algorithms are the metaheuristic algorithms (Dorigo & Stutzle, 2003).

Metaheuristic algorithms are high-level strategies or general heuristics designs that explore the search space and search for a high-quality solution for the optimisation problem. Some well-known metaheuristic algorithms include ACO (López-Ibáñez et al., 2016), SA (Franzin & Thomas Stutzle, 2018), TB (Hoos & Stutzle, 2015), GA (Jacobson & Kanber, 2015) and ILS (Stützle & Ruiz, 2017). Metaheuristic algorithms can be classified into two groups: constructive or local search based (iterative).

Constructive metaheuristic algorithms build the optimal solution from their constituent elements. The greedy algorithm concept is often used to supplement the best available element to the solution. The iterative metaheuristic algorithms determine the optimal solution by iteratively replacing the current solution with its neighbourhood to achieve further improvement (Blum & Roli, 2001). The ACO belongs to the former (Levine & Ducatelle, 2004), whereas the others belong to the latter. ACO has been determined to use stochastic movements, heuristic information, and memory in the search process and lead to high quality solutions (López-Ibáñez et al., 2016). These features will allow ACO to explore large search spaces without becoming trapped in local minima.

However, ACO suffers from a local optimisation problem, i.e., the searching strategy is restricted to a global search only, which produces vulnerable solutions. Local search algorithms are fast and efficient. They start with an initial solution and spend a long time achieving improvement using the available neighbourhood space. However, local search algorithms cannot escape from local minima. Hence,

hybridisation between the ACO and local search algorithm is required to solve the disadvantages of both algorithms. The ACO is used to explore the new area within the search space and produce the initial solution, whereas local search algorithms use this solution to utilise and improve the search in the neighbourhood space. Many studies have reported the improvement of hybridisation between ACO and local search.

Abuhamdah (2018) adopted the hybrid elitist ant system variant of the ACO algorithm for DM clustering tasks. This study combined the elitist ant system with the ILS algorithm and tuned the importance of the constraints' parameter for each dataset. The goal of the study is to introduce an elitist ant system that can utilise the search space in the clustering domain. Experiments were conducted using six medical benchmark datasets from UCI. The computational results show that the proposed algorithm produced solutions with higher quality than other approaches (Abuhamdah, 2018).

Lin et al. (2017) combined ACO with an 1-opt local search procedure for forest transportation planning problems (FTPPs). The proposed mechanism introduced to improve the solution quality obtained by ACO further. The 1-opt local search procedure is applied at the end of the iteration to search for potential shortcuts at each vertex and its adjacent vertices along the individual origin–destination routes to improve the quality. The performance of the proposed approach was tested in a hypothetical FTPP and then tested using 10 FTPP instances generated using the same FTPP. Afterwards, the proposed method was applied in a real, large-scale FTPP. The

results show that the proposed approach can match the feasible solutions for all cases of all FTPP tests.

Another study produced by Jaradat (2016) combined elitist–AS, a variant of ACO, with the ILS algorithm to solve the well-known symmetric traveling salesman NP-hard problem. The case study involved different instances of TSP, including two instances that consist of 26 cities and 1094 locations in Jordan. This research aims to maintain the elitist–AS exploitation mechanism by using ILS to intensify the search around the elite solution. The results display that the proposed hybridisation can produce optimal results compared with other algorithms (Jaradat, 2016).

A successful hybridisation between the original ACO and ILS algorithms for the forecasting of Turkey's domestic electricity consumption has been achieved by Toksari (2016) whose research aimed to combine the advantages of both algorithms to perform forecast estimations. The study applies ILS algorithm to each single rule discovered by ACO algorithm. The performance evaluation was conducted on Turkey's datasets between 2004 and 2013 using different economic indicators (population, export, GDP and import). The evaluation exhibited high quality results when compared to the actual and predicted electricity consumption results. Therefore, the proposed hybridisation is used to forecast Turkey's domestic electricity consumption until 2030 (Toksari, 2016).

Ezzat et al. (2015) proposed an extension of the EigenAnt ant colony system (EAAS) algorithm called probabilistic EAAS (PEAAS). The algorithm applied to the

sequential ordering problem (SOP), which is used to model real-world vehicle routing and transportation problems. The enhancement has been applied using two procedures. Firstly, the stochastic degree in the solution construction process was increased. Secondly, the SOP-3-exchange local search procedure was added to enhance all solutions in the solution construction steps. The result indicated that PEAAS is better than the original EAAS and state-of-the-art enhanced ACS (EACS) algorithms.

Liao et al. (2011) introduced an incremental ant colony algorithm with local search (IACORLS) to solve a continuous optimisation problem. IACORLS is an improved version of the ACOR algorithm with an extra procedure for search diversification, which consists of a growing solution archive, followed by the hybridisation between the ACOR algorithm and the local search procedure to increase the intensified capabilities. The experiment used three local search procedures: Powell's BOBYQA, Lin-Yu Tseng's Mtsls1 and Powell's conjugate directions set. The results were tested using two benchmark functions in SOCO and the special session on continuous optimisation. The results showed that the combination of the ACOR algorithm and Mtsls1 obtains the best result amongst the possible combinations. Moreover, IACROLS is considered one of the best continuous optimisation algorithms.

Drias et al. (2015) proposed a hybridisation of ACO and TS in medical web information foraging, the framework of which consisted of two main steps. The first phase performed learning on several web pages' instances for localising the related pages of interest to the user. The second phase considered the dynamicity and

openness of the web by using the results obtained from the first step. Then, the changes that occurred on the web were observed. The experiment used the website of the US National Library of Medicine (MedlinePlus), which consists of 1903 web pages for 900 diseases. The results from this experiment were promising from two aspects: response time and strong web relatedness.

Eswaramurthy and Tamilarasi (2009) presented the global optimisation of the hybrid ACO with TS in job shop scheduling problems. The aim of this coupling was to utilise pheromone trails and dynamic tabu in selecting the best available neighbours to enhance the quality of the solution. This coupling helps escape from the local optima, avoids cycling and accelerates the convergence of the search process. The performance was tested using two matrices: quality of solution and CPU run time. The results showed that the proposed algorithm performs better than well-known algorithms.

Another study introduced different approaches based on the hybridisation between ACS and local search procedures in different optimisation problems. The study, conducted by Gambardella (2015) in a PhD thesis, consisted of the proposed approaches published with other co-authors. The first approach consisted of parallel stochastic ACS coupling with restricted three-opt procedure as a local search procedure in a TSP optimisation problem. The study showed that ACS provides good starting solutions for local search optimisers. The experiment results displayed that ACS performs better than other nature-inspired algorithms, i.e., evolutionary computation and SA. In addition, this paper was the second most cited in *IEEE*

Transactions on Evolutionary Computation. The second approach proposed the coupling between multiple ACS with CROSS exchanges as local searches for time window vehicle routing optimisation problems. The study introduced two colonies: the first reduces the number of the vehicles, whereas the second reduces the travel distances, subsequently sharing information between colonies through pheromone updating. The performance of this approach was tested in terms of quality and execution time, showing that it is comparable with well-known methods. The third approach highlighted other couplings between ACS and a new local search mechanism to work with a sequential ordering optimisation problem called SOP-3-exchange. The approach introduced the SOP-3 exchange local optimiser, which was designed for the sequential ordering problem regarded as the contribution of this study. The experiment evidence showed that the implementation of the local search greatly improves ACS performance. The performance results show that the proposed approach is better than existing methods in the sequential ordering optimisation problem. Another approach was proposed by Gambardella, Montemanni and Weyland (2012) by using hybrid ACS with strong local searches for three transportation optimisation problems. The proposed approach aims to improve the performance of the ACS algorithm in the constructive stage and integrates it with a strong local search optimiser. In this mechanism, the local search runs if and only if the obtained solution in the constructive stage is within 20% of the best quality solution to accelerate the ACS algorithm. The evaluation results display that EACS obtains a higher quality of solutions than classical ACS methods. Finally, the ACS algorithm was applied with local search procedure in different real-world vehicle routing problems. The performance in these real problems showed that ACO is one

of the best and most successful metaheuristics in vehicle routing optimisation problems.

Guan and Lin (2016) proposed the hybridisation between ACO and variable neighbourhood search for single-row facility layout as an optimisation problem. The enhancement approach modified the original ACO in different directions. The first modification, conducted in the construction stage, proposed new objective function formulas to compute the values in different neighbourhoods. A feedback criterion based on edit distance was measured to avoid local optima. Then, a new pheromone updated the strategy on the basis of the best and worst solutions introduced. In addition, an extra variable neighbourhood search procedure was added as a local search mechanism, which uses the first improvement via three different neighbourhood structures to change in a random search. The experiment results showed that the proposed algorithm is comparable with other algorithms in single-row facility layout optimisation problems.

De La Cruz, Paternina-Arboleda, Cantillo and Montoya-Torres (2013) combined the ACS algorithm with TS in the Heterogeneous Vehicle Routing Problem with Multiple Products and Time Windows (HVRPTWMP). The approach is based on two pheromone strategies to accelerate the ants in the searching process. In addition, HVRPTWMP uses two characteristics, namely, recent and frequent event memories and diversification. This approach utilises regency and frequency memories from TS to search for further improvement in the quality of solutions. Meanwhile, diversification is applied using the best solution of the current neighbourhood instead

of the standard TS that uses the best solutions to guide search diversification. In the second stage, the standard TS stores the best solution explicitly, whereas this approach stores the attributes. The performance uses instances from the literature and shows that the quality of the proposed approach is comparable with the best-known solutions from the literature.

Sagban, Ku-Mahamud and Abu Bakar (2016) proposed overcoming the problem of the exploitation in ACO when hybrids are included in local searches. Their approach applies two mechanisms to solve this drawback, namely, recursive local search method and reactive heuristics based on information recorded in two auxiliary memories in the search process to guide the search in the future. The first auxiliary memory indicates the arcs with low pheromone density during the pheromone update procedure at each step for further significance. The recursive local search mechanisms use population-based memory to handle the solution that has been generated in the previous populations and used iteratively to utilise the neighbourhood of the current solution. These proposed techniques were implemented in the MMAS variant of ACO in two optimisation problems: TSP and quadratic assignment problem (QAP). The results were evaluated with different ACO variants coupled with local search procedures, displaying that the enhanced algorithm is better than six ACO variants.

Saian and Ku-Mahamud (2012) proposed Ant-Miner coupled with SA to generate a list of classification rules. In Ant-Miner, each ant discovers a rule. The study proposed SA as a local search procedure to improve this rule iteratively. The SA

works for each rule on the basis of the temperature variable, which starts at a high value and then decreases on the basis of predefined factors. The search runs a certain number of iterations and selects the best rule from the available neighbourhood depending on its quality. Using the SA mechanism, which starts with high temperatures, allows the rule with low quality to be selected. Then, temperature will be decreased, and the difference between the current and previous qualities will be crucial for selecting the rule with high quality. The performance matrix is indicated on the basis of the rule quality, the number of discovered rules and the terms per rule. The performance of this approach was tested using 13 datasets from a UCI repository, showing that it is comparable with the original Ant-Miner in the predicative accuracy.

A hybridisation of ACO and ILS algorithms based on new pheromone-update rules on TSP and job shop scheduling NP-hard problems was proposed by Fox et al. (2007). The proposed pheromone-update rules are as follows: (1) rank-based; and, (2) IB/GB pheromone updates. The experiment result carried out on 16 TSP problems indicated that the hybrid ACO/ILS-rank-based outperforms the non-hybrid rank-based method. Similarly, the hybrid ACO/IB/GB method outperforms that of the non-hybrid IB/GB-based method. In addition, the experimental results on job shop scheduling for FT06, FT10 and FT20 problems were promising (Fox et al., 2007).

Ji et al. (2006) combined the ACO and GA as a local search procedure. The Ant-Miner algorithm generates the list of rules in each iteration and selects the best rule

to be added in the final discover rule list. The proposed mechanism uses the best rule and the mutation mechanism used in GA to mutate the attribute value one at a time and then test the quality of the newly generated and original rules if and only if the quality increases. As such, the new rule will replace the original. The experiment was conducted using six datasets from the UCI repository, showing slight improvement in quality and the comprehensibility of the rule.

In the above-reviewed studies, two types of local search procedure are hybrid with ACO. The first type is the heuristic method based on the iterative exploitation of neighbourhoods to improve the current solution by local changes (Ezzat et al., 2015; Gambardella, 2015; Lin et al., 2017). Although the solution quality increases with a large number of neighbourhoods, these methods take exponential time. The second type, namely, metaheuristics, also uses strong local search, which increases the performance of heuristic methods by guiding the search with high-level principles and strategies (i.e., based on objective function, memory and prior performance) (Abuhamdah, 2018; De La Cruz et al., 2013; Drias et al., 2015; Eswaramurthy & Tamilarasi, 2009; Fox et al., 2007; Gambardella, 2015; Gambardella et al., 2012; Guan & Lin, 2016; Jaradat, 2016; Ji et al., 2006; T. Liao et al., 2011; Sagban et al., 2016; Saian & Ku-Mahamud, 2011; Toksari, 2016). These algorithms overcome the drawbacks of the heuristic methods in a further intelligent way, which allows the local search to escape from local minima by generating a new initial solution or allows the worsening moves rather than simply providing random starting solutions. Table 2.1 summarises the recent successive works of the ACO hybrid with local search in different application domains. For each study we list the ACO types, local

search method and problem name. Overall, the results show that the local search plays a crucial role in improving the quality of the solution in different ACO variants. In addition, the study observes two types of local search heuristic and metaheuristic methods. The subsequent sub-sections discuss the basic concept of TS, SA, GA and ILS as metaheuristic local search-based algorithms in detail.

Table 2.1

Summary of the Successful Hybridisation Between ACO and Different Local Search Methods

References	ACO variant	Local search method	Problem name
(Abuhamdah, 2018)	elitist-AS	ILS	Data clustering.
(Lin et al., 2017)	AS	1-opt local search	Forest transportation planning problems
(Jaradat, 2016)	elitist-AS	ILS	TSP
(Toksari, 2016)	ACO	ILS	Turkey's electricity domestic consumption
(Ezzat et al., 2015)	PEAAS	SOP-3-exchange local search	Sequential ordering problem
(Liao et al., 2011)	ACO R	Multiple Trajectory Search	Continuous optimization problems
(Drias et al., 2015)	AS	TS	Medical data management
(Eswaramurthy & Tamilarasi, 2009)	ACS	TS	Job shop scheduling problems
(Gambardella, 2015)	ACS	SOP-3-exchange CROSS Exchanges SOP-3-exchange Strong local searches	TSP Vehicle routing problems Sequential ordering problem Transportation problems
(Gambardella et al., 2012)	ACS	Strong Local Search	Transportation problems
(Guan & Lin, 2016)	AS	variable neighbourhood search	The single row facility layout problem

(De La Cruz et al., 2013)	ACS	TS	Vehicle routing problem
(Sagban et al., 2016)	MMAS	Recursive local search	TSP and QAP
(Saian & Ku-Mahamud, 2011)	Ant-Miner	SA	Data classification
(Fox et al., 2007)	ACO	ILS	TSP Job shop scheduling
(Ji et al., 2006)	Ant-Miner	GA	Data classification

2.5.1 Tabu Search

TS, first introduced by Glover (1986), is a global optimisation technique used to escape from local optima and implement an explorative strategy. TS uses memory structure to consider the solution that has been previously visited in a short-term period or has violated a rule (Blum & Roli, 2001). The algorithm marks these as tabu (forbidden); thus, the TS will not repeatedly provide the same solutions given the use of memories. The main components of the TS algorithm are neighbour choice tabu list and aspiration criterion. The first component is short-term memory, namely, tabu list. This memory tracks the most recently visited solutions, forbids moves towards them and prevents the search from endless cycling. The second component is aspiration criterion. Solutions in the tabu list are sometimes considered powerful; the aspiration criteria are used to construct solutions even though they lead to an overall stagnation of the searching process. The common use of the aspiration criterion is selecting a solution from the tabu list if it is better than the currently constructed best solution. Moreover, the global information (i.e., recency, frequency, quality and influence) collected in the run process can be implemented as strategic guidance for the search process. This type of information is presented as long-term memory to

achieve a trade-off between exploration and exploitation of the search process. Figure 2.5. shows the algorithmic skeleton of TS algorithm.

```
1   InitialMemoryStructure, GenerateInitialSolution   S , Sbest = S
2       While termination condition is not met
3           A= GenerateAdmissibleSolution (S)
4           S= SelectBestSolution (A)
5           If (f(S) < f(Sbest))
6           Sbest =S
7       End
8       Return Sbest
9   End
```

Figure 2.5. TS pseudocode

2.5.2 Simulated Annealing

In SA (Aarts et al., 1997), as metaheuristics is local search-based, the method of searching is inspired from annealing in metallurgy; its technique involves material temperature by controlling heating and cooling to increase the size of its crystals and reduce their defects. Through the search, the worse solution is accepted to avoid the local minima. The algorithm starts with an initial solution and iteratively improves an initial solution. Two components in SA control exploration and exploitation. The first is the cooling schedule in which the temperature starts at a high degree and decreases in the search process to the final degree. At each temperature, the search precedes a certain number of steps to select the best solution from available neighbourhoods. The temperature start value, end value, decreasing value and number of search steps at each temperature are the parameters of the cooling schedule. Many studies are attempted to derive an appropriate cooling schedule in SA. The second component is the acceptance criterion for guiding the search on the basis of the metropolis algorithm, which uses a probability function to accept or

reject movement to the next step. Figure 2.6 shows the pseudocode of the Simple SA algorithm.

```
1   GenerateInitialSolution  S , S_best = S, initial value for T_0 , n=0,
2    While termination condition of outer-loop is not met
3        While termination condition of inner-loop is not met
4            S'= GenerateRandomSolution (S)
5            S= Accept Solution (T_n,S,S')
6            If (f(S) < f(S_best))
7            S_best =S
8         End
9         T_n+1=UpdateTemp(T_n), n=n+1
10    End
11  return S_best
12  End
```

Figure 2.6. SA pseudocode

2.5.3 Genetic Algorithm

GA is naturally inspired by Darwin's biological evolution theory, including the principles of natural selection, recombination, mutation and reproduction (Srinivas & Patnaik, 1994). The high-level algorithmic outline of the GA is shown in Figure 2.7. GA is a metaheuristic population-based method for solving NP-hard problems. The GA starts by randomly creating and iteratively improving many initial solutions. The basic idea creates an initial population, namely, chromosomes, that represents the solutions to the optimisation problem. The solutions are then evaluated on the basis of fitness function to create a new generation by combining two solutions, called parents, using a crossover operator. The crossover operator exchanges useful information between solutions to generate improved productions, namely, offspring. A mutation operator is used to alter small modifications randomly in the offspring. In this operator, the solution may change from the previous one.

```
1  Initial population;
2  Evaluate population;
3      While termination condition is not met
4          Select solutions for next population;
5          perform crossover;
6          perform mutation;
7          Selection;
8      End
9  End
```

Figure 2.7. GA pseudocode

Finally, the selection operator selects some solutions from the parents and offspring on the basis of the fitness function (solutions with high fitness will survive); this operator aims to keep the population at a constant size. The complete cycle of these three operators (crossover, mutation and selection) is called generation. The algorithm will converge on the best solutions, which represent the optimal solution for the optimisation problem, after several generation steps.

2.5.4 Iterated Local Search

ILS is a metaheuristic algorithm with the basic idea of using perturbation for an incumbent solution to generate other solutions for the next iteration. ILS is an iterative approach that combines two components: intensification and diversification (Lourenco, Martin, & Stutzle, 2003). Intensification has deterministic and quick capabilities to bring solutions to a local minimum. Diversification is a stochastic mechanism, which can explore the search space of all possible solutions. The algorithmic outline of ILS is presented in Figure 2.8 below. The ILS starts by generating and subsequently improving the initial solution with a local search procedure. The ILS has an iterative loop that consists of three steps. The first step

randomly perturbs the current solution to explore new areas of search spaces globally.

```
1   S0= Generate Initial Solution
2   S*= LocalSearch(S0)    % optional
3   Repeat
4       S'= perturbation
5       S*'= LocalSearch(S')
6       S*= AcceptanceCriterion (S*,S*')
7   Until termination condition met
8   End
```

Figure 2.8. ILS pseudocode

The second step is immediately applied to the local search procedure for the new solution. The last step is the acceptance criterion, which can be in two different behaviours, greedy (accepting the best one between the solutions) or probabilistic, similar to SA (worsened solution has chances to be selected). ILS will complete iterations from one local minimum to the next.

2.5.5 Discussion on Metaheuristic Hybrid with Local Search

The metaheuristic algorithms have been inspired by naturally occurring phenomena. The inspiration is embodied into some computational characteristics. The characteristics of metaheuristic algorithms include whether to use a single point or population of search. Another important feature of metaheuristics is using memory to guide the direction of future searches. In addition, some metaheuristic algorithms use single- or multiple-neighbourhood structures. The majority of local search algorithm use a single-neighbourhood structure that determines the type of allowed moves. This type of move is used in TS and SA algorithms. Conversely, the ILS uses multiple-neighbourhood structures (N and N′). In the ILS, the local search procedure

starts with the neighbourhood of N solution until reaching local optimum. Perturbation then catapults the search to another point. For subsequently repeating the local search, the primary neighbourhood is used. The GA uses a mutation operator, which has the same effect as perturbation. The crossover operator moves in hyper-neighbourhoods. Unlike the local search algorithm, the solution construction in ACO is not based on a specific neighbourhood structure. Table 2.2 shows the main characteristics of metaheuristic algorithms. '√' means the feature is present, and '–' indicates otherwise.

Table 2.2

Characteristics of Metaheuristic Algorithms

Characteristics	ACO	SA	TB	GA	ILS
Population	√	-	-	√	-
Memory	√	-	√	-	√
Multiple neighbourhoods	-	-	-	√	√
Constructive	√	-	-	-	-
Iterative	-	√	√	√	√

Recent studies have shown that the performance of ACO improves when hybridisation is performed with a local search algorithm. The hybridisation must consider the characteristics of metaheuristic algorithms, as shown in Table 2.4. These characteristics must be further studied to determine which algorithmic component is to be integrated. For example, the multiple-neighbourhood structures of ILS are used to guide the search for new regions (Blum & Roli, 2001). An in-depth investigation is needed where local search improves the classification task; otherwise, it will not have any additional performance improvement. In addition, the original Ant-Miner

and its variants do not use any local search to explore more terms and edges except the two studies in the literature, which also have drawbacks. In first study, Ji et al. (2006) combined Ant-Miner and GA as a local search procedure by using only a mutation mechanism to mutate one attribute value of the best rule found in the iteration; if and only if the quality improves, the new will replace the original rule. The combination is only used in the mutation operator; it will not have a high-quality solution whilst it is not able to explore and does not iteratively improve the neighbourhoods of the current solution. In a further study, Saian and Ku-Mahamud (2012) coupled Ant-Miner with SA as a local search procedure. Each ant in each colony uses the SA to discover one rule. The SA works for each rule on the basis of the temperature variable, which starts with a high value then decreases on the basis of pre-defined factors. The search runs a certain number of iterations and selects the best rule from available neighbourhoods on the basis of quality. The move results in minimal improvement because the local optimisers do not forbid the current neighbourhood structures from moving to the next iterations. Thus, the Ant-Miner algorithm will consume additional time in improving the same search regions to determine the best improvement. To the best of our knowledge, no research has been conducted on Ant-Miner hybrid and ILS algorithm as a local search. The diversification of Ant-Miner and the intensification of ILS can be coupled to balance local exploitation and global exploration via multiple-neighbourhood structures. In addition, restricting the ILS to the best iteration will save computation time whilst providing a strong possibility to improve the best solution obtained by the ants.

2.6 Rule Pruning

Rule pruning, a common technique in rule-based classifiers, reduces the size of the discovered rules by avoiding the overfitting noisy data (Bramer, 2002). Noise in a dataset is caused by certain reasons (incorrect input, programming errors and hardware failures). Such noisy data have a detrimental effect by misguiding the learning algorithm and producing a poor classifier performance. In learning, the algorithm adds terms to the rule to increase its classification accuracy by fitting the instances closely. In this way, the rule will cover positive instances (instances correctly predicted by the rule) and then remove them from the training set; the new existing instances are subjected to a new rule generation step. Subsequently, this type of rule is a perfect fit for instances from which they were generated. By contrast, rules generated from noisy instances are highly complicated, lack usefulness and exhibit low classification accuracy in classifying unseen instances. This problem is known as overfitting, which can occur when the constructed rules fit well, or exactly, a particular training instance and cannot apply on unseen data. These rules negatively affect the entire performance of the training model. An example of the overfitting rule picked up from the Ant-Miner algorithm without using rule pruning occurred in an experiment undertaken on a breast cancer dataset from the Ljubljana UCI repository, as shown in Figure 2.9. This dataset consists of nine attributes, all of which appear in the rule. Moreover, the rule is a perfect fit to the data instances from where it is generated.

IF age ='50-59' AND menopause ='ge40' AND tumor-size ='20-24' AND inv-nodes ='3-5' AND node-caps= 'yes' AND deg-malig ='2' AND breast ='right' AND breast-quad ='left_up' AND irradiat ='no' THEN 'no-recurrence-events'

Figure 2.9. Example of Overfitting Rule

The above-mentioned problem can be solved by detecting the significant terms in the generation rule and pruning the irrelative terms that provide minimal quality to classify the instances. This mechanism aims to improve the accuracy of the rule and reduce its complexity (Fürnkranz, 1997). Post-pruning, pre-pruning and hybrid pruning are three strategies used in Ant-Miner rules-based classifiers. In the pre-pruning strategy, the rule discovery algorithm is halted before creating a full rule. The stopping condition handles irrelevant terms during learning (i.e., stop selecting the term when the impurity measured for some terms is less than the pre-determined value). However, post-pruning handles irrelevant terms after an overfitting rule has been constructed; in this strategy, the rule grows to maximum size. The irrelevant term is deleted from the rule. Hybrid pruning combines the characteristics of post-pruning and pre-pruning. To observe the influence of rule pruning, an experiment was undertaken with the same Ljubljana breast cancer dataset using the same parameter setting on the Ant-Miner algorithm with the traditional post-pruning procedure. A different rule was obtained from similar instances, as shown in Figure 2.10. This rule involves only two attributes in its structure. Thus, the rule is simple and has few terms. Conversely, this rule covers more instances and is more accurate than the rule in the previous example in Figure 2.9.

IF menopause = 'ge40' AND irradiat = 'no' THEN 'no-recurrence-

Figure 2.10. Example of Pruned Rule

In all pruning strategies, any excess of pre-pruning and post-pruning in the rule may lead to a simple rule that cannot capture the underlying structure of the data. This problem is known as under-fitting. Thus, the rule will be unsuitable and lead to poor predictive performance on the data. Figure 2.11 shows overfitting and under-fitting rules based on predictive error and model complexity.

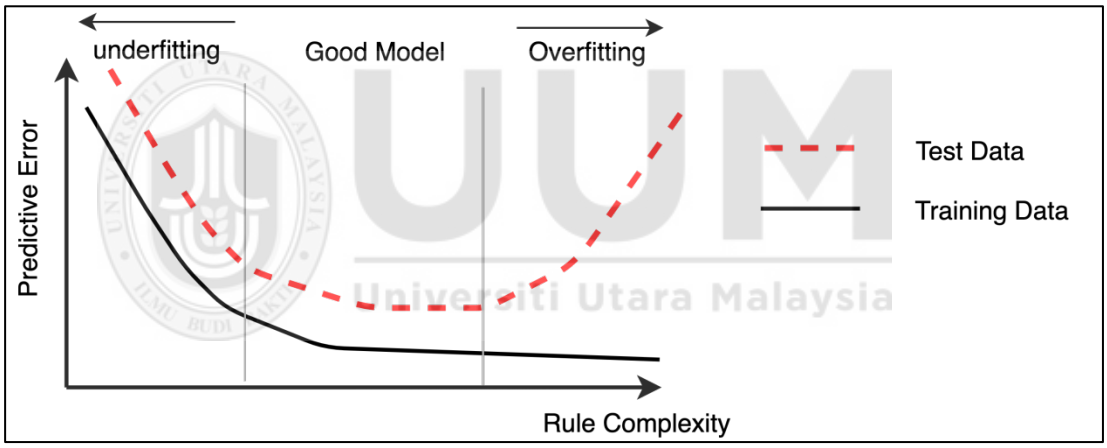

Figure 2.11. Overfitting and Under-fitting Effects on the Classification Accuracy and Complexity of the Rule (Al-behadili, Ku-Mahamud, & Sagban, 2018; Lean, Wang, & Kin, 2006)

Rule pruning is included in the Ant-Miner variant to avoid the problem of overfitting, reduce rule complexity and improve predictive performance. Therefore, designing the algorithm without pruning techniques will introduce complex rules and face overfitting problems (Mashhour et al., 2018). The next sub-section discusses the pruning strategies in Ant-Miner variants in detail.

2.6.1 Post pruning Technique in Ant-Mining Classifiers

The traditional technique used in the majority of Ant-Miner variants is the post-pruning technique. The pruning procedure is inspired from the method proposed by Quinlan (1987). This technique removes one term at a time from the rule, thereby improving the quality of the rule. This procedure iterates until no further improvement occurs or only one term is left in the rule. Post-pruning is then terminated to avoid the under-fitting rule. The class value of the dataset can change during this procedure because the majority of classes in the instances covered by the pruned rule might be changed compared with those covered by the original rules (Al-Abadi, 2017; Arif-Ul-Islam & Ripon, 2019; Ayub, Ikram & Shahzad, 2019; Durgadevi & Kalpana, 2018; Fakhar, 2014; Holden & Freitas, 2008; Jaganathan et al., 2007; Ji et al., 2006; Jiang et al., 2005; Jin et al., 2006; Lai et al., 2016; Liang et al., 2011; Liu et al., 2002, 2004; Madhusudhanan et al., 2010; Mashhour et al., 2018; Otero et al., 2008; Parpinelli et al., 2002; Rajpiplawala & Singh, 2014; Saian & Ku-Mahamud, 2011; Uthayakumar, Vengattaraman & Dhavachelvan, 2017; Wang & Feng, 2004; Wu & Sun, 2012).

Numerous algorithms proposed by (Agravat et al., 2010; Salama & Abdelbar, 2010; Salama & Otero, 2013; Smaldon, 2006; Yang, Li, Zhang & Ke, 2016) still use the same traditional procedure to prune the rule, but introduce a new fitness function to test the accuracy. In addition, a dubbed threshold-aware pruning mechanism and new fitness function are used in algorithms (Michelakos et al., 2010; Otero et al., 2009) sensible to the order of terms consisting of continuous values. This mechanism removes the last term added to the rule for simplification until the rule quality

decreases when the last term is removed, or the rule has only one term left. Another algorithm provides a new fitness measurement function and simplifies the traditional pruning by allowing the consequent part of the rule to remain unchanged during pruning (Smaldon, 2006).

A new post-pruning method is introduced by several studies to prune only the best rule discovered by all ants instead of pruning each rule constructed by each ant (Baig & Shahzad, 2012; Martens et al., 2007; Shahzad & Baig, 2010). Furthermore, the rule quality functions are changed by using new functions in those algorithms.

The algorithm that handles hierarchical multi-label classification, which was proposed by Prabha and Balraj (2014), considers the replacement of consequent rules during each single pruning. Pruning removes one term and recalculates the resulting class because the set of covered instances may change after the pruning of the last term. This step removes one term and replaces the class consequent. Then the quality of the candidate rule is measured and compared with the original rule. If the candidate rule has a higher quality than the original rule, then the former replaces the latter. This procedure is iterated until no further quality improvement occurs or only one term is left in the rule. In addition, this method uses a distance-based measure as a quality function.

2.6.2 Pre-pruning Technique in Ant-Mining Classifiers

The pre-pruning criteria were proposed by Thangavel and Jaganathan (2007) and Tripathy et al. (2013) in which the construction step accepts or rejects a term to be added to the rule-based method on its strength or importance rather than the post-

pruning method. This step will reduce the number of irrelative terms in the rule. However, this criterion, based on threshold value, aims to reduce the complexity of post-pruning by disallowing the irrelative term to be part of discovered rules.

The algorithm proposed by Martens et al. (2007b) removes pruning and introduces a new mechanism that extends the domain of each attribute with a dummy value 'any'. In the rule construction stage, the selection term of 'any' value means that a term is absent in the rule antecedent, leading to a shortened rule. ACO/PSO2 computes the number of terms included in the best rule discovered; if, and only if, the constructed rule entails more than 20 terms for each rule, then the pruning iterates to remove the unimportant or detrimental terms from the classification rule until the number is decreased to 20 (Holden & Freitas, 2008). In these mechanisms, the removal of post-pruning will make the algorithm less expensive.

2.6.3 Hybrid Pruning Technique in Ant-Mining Classifiers

Rule pruning is sensitive to the number of attributes in the dataset being mined because many terms will be included in the constructed rule in the construction stage, followed by numerous iterations of rule pruning to check rule quality during pruning. A hybrid rule pruning technique proposed by Chan and Freitas (2006) combines traditional rule pruning on the basis of the rule quality with a new procedure based on information gain. This hybrid pruner depends on a threshold user pre-defined value called r, which represents the total number of terms in the constructed rule. If the number of terms in the constructed rule does not exceed the value of r, then the traditional rule pruning is directly applied. However, if the constructed rule exceeds

the value of r, then the procedure first reduces the number of terms in the constructed rule to the r value by selecting the number of terms using the pre-calculated term's information gain. Subsequently, the value of heuristic function is computed in the rule construction stage with respect to specific class. The procedure will select r number of terms using the well-known roulette wheel selection technique, and the new selected rule is placed straight into the traditional rule pruning.

The rule pruning review provides a year-wise distribution over 38 research studies of ant-mining algorithms from 2002 to 2019. The usage of the traditional post-pruning that is produced by the original Ant-Miner overshadows other methods. In addition, experiments were conducted in the literature using three methods to observe the influence of rule pruning (Chan & Freitas, 2006a; Parpinelli et al., 2002b; Thangavel & Jaganathan, 2007). Datasets from the UCI repository were used by those studies, and the overall comparison results show that using the Ant-Miner algorithm without pruning leads to decreased classification accuracy and produces a complex rule (Mashhour, ElHouby, Wassif & Salah, 2018). Thus, pruning is an important component of any Ant-Miner algorithm. Overall, the results show that the pre-pruning technique can perform better than traditional post-pruning in terms of prediction accuracy and simplicity. The pre-pruning methods eliminate irrelevant terms on the basis of the threshold value in the rule generation stage. Table 2.3 shows a comprehensive overview of the strategies of rule pruning used in Ant-Miner variants. For each ant-mining algorithm, we review if the algorithm pruning using pre-pruning, post-pruning or hybrid pruning. '√' means the strategy is present, whereas '–' means otherwise.

Table 2.3

Techniques of Rule Pruning in Ant-Mining Classifiers

No	Reference	Classifier Name	Pre-Pruning	Post-Pruning	Hybrid Pruning
1	(Ayub et al., 2019)	AMclr	-	√	-
2	(Arif-Ul-Islam & Ripon, 2019)	Ant-Miner	-	√	-
3	(Durgadevi & Kalpana, 2018)	Ant-Miner	-	√	-
4	(Mashhour et al., 2018)	Ant-Miner	-	√	-
5	(Al-Abadi, 2017)	Ant-Miner	-	√	-
6	(Uthayakumar et al., 2017)	Ant-Miner	-	√	-
7	(Yang et al., 2016)	Ant-MinerPAE	-	√	-
8	(Lai et al., 2016)	Ant-Miner	-	√	-
9	(Prabha & Balraj, 2014)	hmAnt-Miner	-	√	-
10	(Otero, Freitas, & Johnson, 2013b)	cAntMinerPB	-	√	-
11	(Fakhar, 2014)	Ant-Miner 4	-	√	-
12	(Salama & Otero, 2013)	cAnt-Miner + µAnt-Miner	-	√	-
13	(Tripathy et al., 2013)	MTACO - Miner	√	-	-
14	(Wu & Sun, 2012)	mAnt-Miner	-	√	-
15	(Wu & Sun, 2012)	mAnt-Miner +	-	√	-
16	(Baig & Shahzad, 2012)	AntMiner-C	-	√	-
17	(Saian & Ku-Mahamud, 2012)	Hybrid ACO + SA	-	√	-
18	(Liang et al., 2011)	P-Ant-Miner	-	√	-
19	(Shahzad & Baig, 2010)	CAnt Miner	-	√	-
20	(Madhusudhanan et al., 2010)	FACO	-	√	-
21	(Michelakos et al., 2010)	Hybrid cAnt-Miner 2 with mRMR	-	√	-
22	(Salama & Abdelbar, 2010)	µAnt-Miner	-	√	-

23	(Agravat et al., 2010)	MACO -I	-	√	-
24	(Otero et al., 2009)	cAnt-Miner 2	-	√	-
25	(Otero et al., 2008)	cAnt-Miner	-	√	-
26	(Holden & Freitas, 2008)	ACO/PSO2	√	√	-
27	(Thangavel & Jaganathan, 2007)	TACO - Miner	√	-	-
28	(Jaganathan et al., 2007)	Ant-Miner + Improved Quick Reduct	-	√	-
29	(Martens et al., 2007)	AntMiner+	√	√	-
30	(Chan & Freitas, 2006a)	Ant-Miner +Hybrid Rule Pruner	-	-	√
31	(Smaldon, 2006)	Unordered Rule Set Ant-Miner	-	√	-
32	(Jin et al., 2006)	SIMiner	-	√	-
33	(Ji et al., 2006)	Enhanced Ant-Miner	-	√	-
34	(Jiang et al., 2005)	Ant-Miner (I)	-	√	-
35	(Liu et al., 2004)	Ant-Miner 3	-	√	-
36	(Wang & Feng, 2004)	ACO -Miner	-	√	-
37	(Parpinelli et al., 2002b)	Ant-Miner	-	√	-
38	(Liu et al., 2002)	Ant-Miner 2	-	√	-

2.6.4 Discussion on Rule Pruning Techniques

Rule pruning has the advantage of selecting the subset of terms from the set of terms in the rule. This type of procedure can simplify the rules and make them easy to interpret, increase generalisation by decreasing overfitting and improve classification accuracy. However, the disadvantage of pre-pruning methods proposed by Thangavel and Jaganathan (2007) and Tripathy et al. (2013) is selecting the correct threshold to

eliminate the irrelevant terms. If a high threshold value is selected, then the discovered rule will under-fit the data. By contrast, a low threshold will not overcome the problem of overfitting. The threshold value is critical and dependent on data, and the selection of an inappropriate value fails to overcome overfitting and under-fitting which then worsen the classification performance. In addition, the extension of the domain of each attribute with dummy value, which was proposed by Martens et al. (2007), will not guarantee that the constructed rule does not consist of any irrelative terms. However, post-pruning used in the majority of ant-mining classification algorithms has limitation to find the best pruning classification rule due to the insufficiency in the sequential backward selection search strategy, which suffers from the problem called 'nesting effect'. Hence, the terms that were once deleted cannot be later re-added. Simplifying post-pruning proposed by Baig and Shahzad (2012), Martens et al. (2007) and Shahzad and Baig (2010) which only prune the best rule discovered by all ants is unideal. In such a case, pruning runs once at a time for the best rule which will not allow high-quality rules to appear because it does not explore all rules or, at least, an elite set of rules. In addition, the hybrid pruning method proposed by Chan and Freitas (2006) is insufficient when a user must set the threshold value or the number of r parameters (number of terms per rule) as well as the problem of original post-pruning technique. The value of this parameter tends to be critical and dataset dependent; user determination may result in a small rule that is not considered an intelligent way for overcoming the problems of over- and under-fitting.

The most popular rule pruning techniques used in ant-mining classification algorithms, as shown in Table 2.5, are the traditional post-pruning techniques. The utmost advantage of the pruning techniques is that they improve the quality of the candidate solutions, whereas the elimination strategy is insufficient to find the optimal pruning rule. Overall, some classification results showed that the pre-pruning technique performs better than traditional post-pruning and hybrid pruning in terms of prediction accuracy and simplicity. Finally, to overcome these disadvantages, the characteristics of pre-pruning and post-pruning can be combined to produce a new hybrid rule pruning technique that features rule accuracy and comprehensibility. This hybrid method may involve a parameter control mechanism to select an appropriate pre-pruning threshold value (i.e., number of terms per rule, term strength or importance), wherein the threshold value can automatically be instantly adapted, rather than being statically determined by the user. The search strategy of post-pruning can be improved by using the characteristic of metaheuristic population-based optimisation (e.g. GA). The GA was explained in detail in Section 2.5. The parameter control mechanisms will be discussed in the next section.

2.7 Parameter Control

Parameter selection is an optimisation problem because it selects the optimal value from large search spaces (Geem, Kim & Loganathan, 2001; Hao, Cai & Huang 2006; Schwefel, 2004). The optimal selection is crucial for balancing exploration and exploitation in rule construction. The literature of parameter control in rule classification does not provide an explicit mechanism for configuring the parameters of Ant-Miner variants (Agravat et al., 2010; Parpinelli et al., 2002; Salama & Otero,

2013), although Martens et al. (2007) considered a self-adaptation mechanism and ants with a personality approach, Salama & Abdelbar (2010) proposed to select the value of two parameters α and β, which determine the relative importance of heuristic information and pheromone trails to increase the classification performance. This section reviews the available research on parameter setting in Ant-Miner variants followed by a review of recent parameter control mechanisms in ACO algorithms because the parameter setting in Ant-Miner variants is still in its infancy stage. This review follows the two stages of parameter setting strategies discussed by Eiben et al. (2007). Parameter setting consists of two types of method. The first method, namely, parameter tuning, sets the parameters in advance and predefines the best value for each parameter before executing the algorithm. The second method modifies the parameters during the construction of the solution (Nallaperuma, Wagner & Neumann, 2014). Different methods are available for changing the parameters during the run. Many methods have been widely studied in evolutionary algorithms (EAs) (Eiben et al., 2007). The global taxonomy of parameter setting in EAs shown in Figure 2.12 is also used in ACO parameter setting (Stutzle et al., 2010).

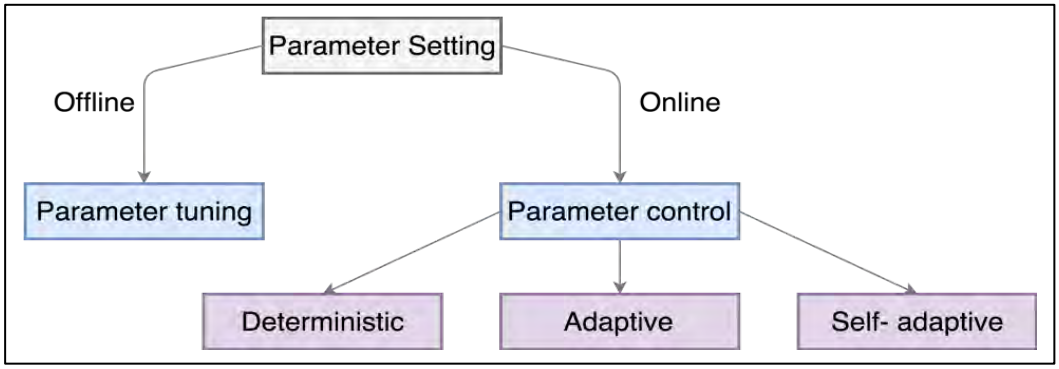

Figure 2.12. Global Taxonomy of Parameter Setting (Eiben et al., 2007; Stutzle et al., 2010)

The issue of setting values of different parameters in ACO-based classification is a promising area of research. This section provides a classification of such setting mechanisms. Some of these techniques exist in the literature of Ant-Miner variants and others in ACO variants.

A study of parameters, by Robu et al. (2016) in the literature of Ant-Miner algorithms, is classified as tuning, showing that values of five parameters, namely, NA, MCR, MUC, RC and NI, vary in different studies. The NA parameter is within [200,7000], the MCR [1,10], MUC [5-15] and RC[5,10]. This study presented 32 combinations of values run on 15 datasets. The dataset was obtained from the UCI repository. The results were analysed on the basis of the accuracy rate of prediction, number of rules discovered and implementation time. The results showed that different parameter values considerably enhance prediction accuracy.

The alternative to parameter tuning is parameter control. Parameter control methods can increase the algorithm's robustness depending on the stage of the search whilst it changes online (Karafotias, Hoogendoorn & Eiben, 2015). Deterministic, self-adaptive and adaptive are different methods that can be used to control the parameters during the run of the algorithm (Brest et al., 2006; J. Liu & Lampinen, 2005).

2.7.1 Deterministic Parameter Control Strategy

The deterministic strategy uses deterministic rules to change the parameters similar to the manner used in simulating annealing algorithms. Those rules are, priori,

calculated to an optimal schedule to change the parameters during the solution construction (Liu & Lampinen, 2005). Merkle et al. (2002) proposed a schedule for the values of β and ρ, changing them linearly. They proposed that a schedule for β parameter linearly decreases the value during the run of the algorithm from the initial value to zero. Additionally, they proposed a small value of ρ at the early stage of search space to increase exploration. They increased the value to the maximum value to determine the best solution. Another study in deterministic parameter control strategy (Meyer, 2004) in α –annealing proposed a variant of AS. The study introduced the annealing schedule to control α parameter during the run. The mechanism slowly increased α parameter in the early stage to maintain the diversity of the solution and gradually increased the parameter value to discover other regions of the search space. Salama and Abdelbar (2010) proposed a new mechanism, namely, the μAnt-Miner algorithm (ants with personality). This mechanism was used to control the values of α and β. The values of α and β parameters were drawn from a random strategy using a Gaussian distribution with a mean of 2 and a standard deviation (σ) in the range [0, 1]. The high values of the standard deviation introduced a high range of diversity in selecting a term. Zhang and Sun (2016) proposed a pheromone update mechanism based on the value evaporation factor called ρ, which was deterministically updated in the search process. This mechanism aimed to control the value of ρ and was called adaptive design. Firstly, the value of ρ will be within $0 < \rho \leq 0.75$. Secondly, as the search process expands, the value of ρ will incrementally increase. The increasing amount will then slow down. Finally, the maximum value of ρ will be 0.75. This mechanism aims to prevent the ant from converging on a specific rule.

2.7.2 Self-adaptive Parameter Control Strategy

The self-adaptive strategy inserts the parameter values into the environment itself (Kramer, 2010; Wang, Cai & Zhang, 2011). Martens et al. (2007) proposed the self-adaptive strategy in the AntMiner+ algorithm by adding two extra vertex groups in the construction graph and allowing the ants to select appropriate values for α and β parameters. The boundaries of α and β parameters are limited within [1,3]. Another study of self-adaptive strategy, proposed by Khichane, Albert and Solnon (2009) in the MMAS algorithm controls α and β parameters. In this study, different mechanisms were proposed. One parameter is independently controlled in the entire colony after iteration rather than in the individual ant level. The authors introduced two mechanisms for a self-adaptive strategy. The first, namely, global parameter learning ant-solver, involves each ant in the colony using the same parameter setting in the search process. In the second mechanism, namely distributed parameter learning ant-solver, each ant in the colony selects new values of α and β parameters in each step of the solution construction.

2.7.3 Adaptive Parameter Control Strategy

The adaptive strategy adjusts on the basis of the feedback from the search behaviour of the algorithm. The feedback mechanism uses some rules that consider the result quality (Qin, Huang & Suganthan, 2009); the adaptive strategy in an ACO algorithm and its variant are based on the pheromone concentration and the quality of solutions. Wang et al. (2015) investigated the online parameter control strategy of ACS and AS variants of the ACO algorithm in DM feature selection. The research used three mechanisms, namely, SI, SII and SIII, based on fuzzy logic control. The study

controlled three parameters: q, q0 and m. The experimental result on a UCI dataset of 10 showed that the proposed algorithms outperform other algorithms in classification accuracy as well as introducing a small subset of features (Wang et al., 2015).

Another study proposed online machine learning strategies to control the parameter of the MMAS algorithm in TSP and QAP problems. This research proposed three mechanisms, namely, QRA, ERA and URQ. The QRA mechanism used the quality of solutions as implicit exploration behaviour. ERA and URQ used the swarm intelligence behaviour to collect feedback and guide the search (Sagban, Ku-Mahamud & Shahbani Abu Bakar, 2015).

Amir, Badr and Farag (2007) proposed a different approach using a fuzzy logic controller in the TSP problem using the ACS algorithm, which also belongs to the adaptive strategy. This approach aimed to adapt the value of β and q_0 on the basis of two measurement factors on the performance of the algorithm. The first is the variance between the best-known solution and the best solution obtained. The second is the variance solutions visited by the population of ants.

Huang et al. (2006) proposed an adaptive parameter control strategy for the variant of ACS in TSP problem. PSO was introduced to explore the parameter space of the ACO. The parameters β, ρ and q0 were considered in this study. The PSO mechanism changes adaptively on the basis of the predefined range for a parameter (Huang, Yang, Hao & Cai, 2006).

Another research study used an artificial fish swarm algorithm for online control of the parameter space of the ACS algorithm. The research considered a different parameter of ACS, i.e., α, ρ and q0 (Abbas, Juan & Mahdi, 2007). The main difference between this mechanism and that of Huang et al. (2006) is that the same parameter setting is kept for all ants.

Another study proposed the usage of GA for the online control of the parameter of ACS variant. This method evolves values of ACS parameters by using the evolutionary algorithm in the search process. This study controlled three ACS parameters in the TSP problem, namely, β, ρ and q0 (Gaertner & Clark, 2005).

Pilat and White (2002) introduced a mechanism that changes the ACS parameter in the search process in TSP problems by using an EA algorithm. The mechanism proposes different values of the parameters for each ant before construction solutions. This study selected three parameters for adaptation, namely, β, q0 and ξ. Their results were somewhat inconclusive (Pilat & White, 2002).

Table 2.4 summarises the parameter setting techniques existing in ACO and Ant-Miner variants. For each study, we review the parameter setting strategy, ACO variants and parameters. Some studies proposed general adaptation strategies that could be used for several parameters. Here, we present the parameters that were experimentally adapted.

Table 2.4

Summary of Parameter Setting Techniques in ACO Variants

References	Setting Strategy	ACO Variants	Parameters
(Robu, Vacar, Robu, & Holban, 2016)	Tuning	Ant-Miner	NA, MCR, MUC, RC
(Merkle et al., 2002)	Deterministic	variant of AS	β , ρ
(Meyer, 2004)	Deterministic	AS	A
(Salama & Abdelbar, 2010)	Deterministic	μAnt-Miner	α , β
(Zhang & Sun, 2016)	Deterministic	Adaptive ACO	ρ
(Martens et al., 2007)	Self-adaptive	AntMiner+	α , β
(Khichane et al., 2009)	Self-adaptive	MMAS	α , β
(Wang et al., 2015)	Adaptive	ACS, AS	q, q0 and m
(Sagban et al., 2015)	Adaptive	MMAS	ρ, γ
(Amir et al., 2007)	Adaptive	ACS	β, q_0
(Abbas et al., 2007)	Adaptive	ACS	α, ρ and q0
(Huang et al., 2006)	Adaptive	ACS	β, ρ and q0
(Gaertner & Clark, 2005)	Adaptive	ACS	β, ρ, and q0
(Pilat & White, 2002)	Adaptive	ACS	β, q0, and ξ

2.7.4 Discussion on Parameter Control Strategies

Parameter setting in ACO algorithms is a complex task (Hao et al., 2006). Table 2.5 summarises the major drawbacks of the parameter setting strategies. '√' means that a drawback exists, and '–' means otherwise.

Table 2.5

Drawbacks of Parameter Setting Strategies

Drawbacks	Strategies			
	Tuning	**Self-adaptive**	**Deterministic parameter**	**Adaptive parameter**
Offline	√	-	-	-
Increased search space	-	√	-	-
Time-consuming	√	√	-	-
Human intervention	√	-	√	-
Do not consider feedback from the search	√	-	√	-
Do not consider different stages of the search	√	-	√	-
Do not consider different domains	√	-	√	-

The first strategy proposed by Robu et al. (2016), namely parameter tuning, has drawbacks. Parameter tuning is time consuming and needs human intervention; its parameters maintain constancy in different stages of the search, and the selection values are not necessarily optimal. These drawbacks worsen with different classification domains and the number of possible values. The deterministic parameter control proposed by (Merkle et al., 2002; Meyer, 2004; Salama & Abdelbar, 2010; Zhang & Sun, 2016) has a problem in determining the control schedule, a priori, and must be appropriately set. The deterministic parameter control does not consider the different stages of search spaces. In addition, the self-adaptive parameter control strategy proposed by Khichane et al. (2009) and Martens et al. (2007) have drawbacks. Firstly, the complexity of the search space in this strategy increases whilst the size of the environment is growing. Secondly, this mechanism needs time to reach the optimal value of the parameters. However, the parameter

adaptation used in other studies (Abbas, Juan & Mahdi, 2007; Amir et al., 2007; Gaertner & Clark, 2005; Huang et al., 2006; Pilat & White, 2002; Sagban et al., 2015; Wang et al., 2015) only utilises the feedback from the search itself to adapt parameter values. Therefore, this research proposes the parameter adaptation strategy to online adapt the pre-pruning threshold value, which is not used in any Ant-Miner variants. This strategy will not be a part of the search space but will be a different search space that externally supply the parameter values. Feedback from the search itself (e.g., the quality of solution) based on the problem at hand is also used to change the pre-pruning threshold value during the fluctuations of the learning process.

2.8 Summary

The Ant-Miner produces a set of rules that achieves performance similar to, and outperforms, other classification techniques in various application domains. This classifier can interpret and explain the relationships between features, which makes it one of the easiest decision making classifiers due to its simplicity. However, the Ant-Miner classifier faces a local optimisation problem because the Ant-Miner algorithm does not use any local search procedure. Additionally, the exiting rule pruning techniques in the literature are inefficient; analysis shows that original Ant-Miner pruning procedure has a drawback with no possibilities of adding elimination terms later. Therefore, this procedure produces poor quality rules. The proposed threshold pre-pruning methods in the literature for term selection are constant, and these values are critical for estimating the importance or strength of the term to be present in the rule. The drawback of the proposed hybrid pruning is that the user must determine

the number of terms that will be included in the rule. Therefore, the proposed threshold (i.e., pre-pruning) in the literature can be improved by using feedback from the learning process to vary this threshold value according to the dataset, and stage of the search. Additionally, new post-pruning techniques can be developed with flexibility to add/remove terms during post-pruning. The ILS has the ability to mature and exploit the neighbourhood structure and obtain a good classification model. The next chapter presents the methodology of this research.

CHAPTER THREE
RESEARCH METHODOLOGY

3.1 Introduction

This chapter describes the research methodology, the framework of the study, and provides the evaluation and validation methods to the domain of rule-based classification. The methodological aspect to address the research objectives as well as the evaluation method and the analysis of evaluation results are also provided. The framework and the process of developing the proposed adaptive ACO algorithm for rules-based classification to achieve the research objectives are described in Section 3.2. The dataset and the data discretisation method are explained in Section 3.3. The summary of this chapter is presented in Section 3.4.

3.2 Research Framework

The framework of the research is a roadmap of this study and the aim is to provide direction to achieve the research objectives. The research framework consists of three stages as shown in Figures 3.1 and 3.2. These figures reveal two different views (i.e., the macro and micro) to the proposed classifier. The first stage includes the development of an adaptive pre-pruning selection technique by using an online adapting criterion and the ACO principles. The second stage focuses on the development of a genetic-based post-pruning technique by using GA for term selection. The third stage includes the proposed hybridisation of the Ant-Miner classification algorithm with the ILS algorithm to improve the best rule discovered

by each ant. The performance of the proposed mechanisms has been evaluated separately as stand-alone algorithms before they are integrated into the proposed adaptive ACO algorithm for rules-based classification (AGI-AntMiner).

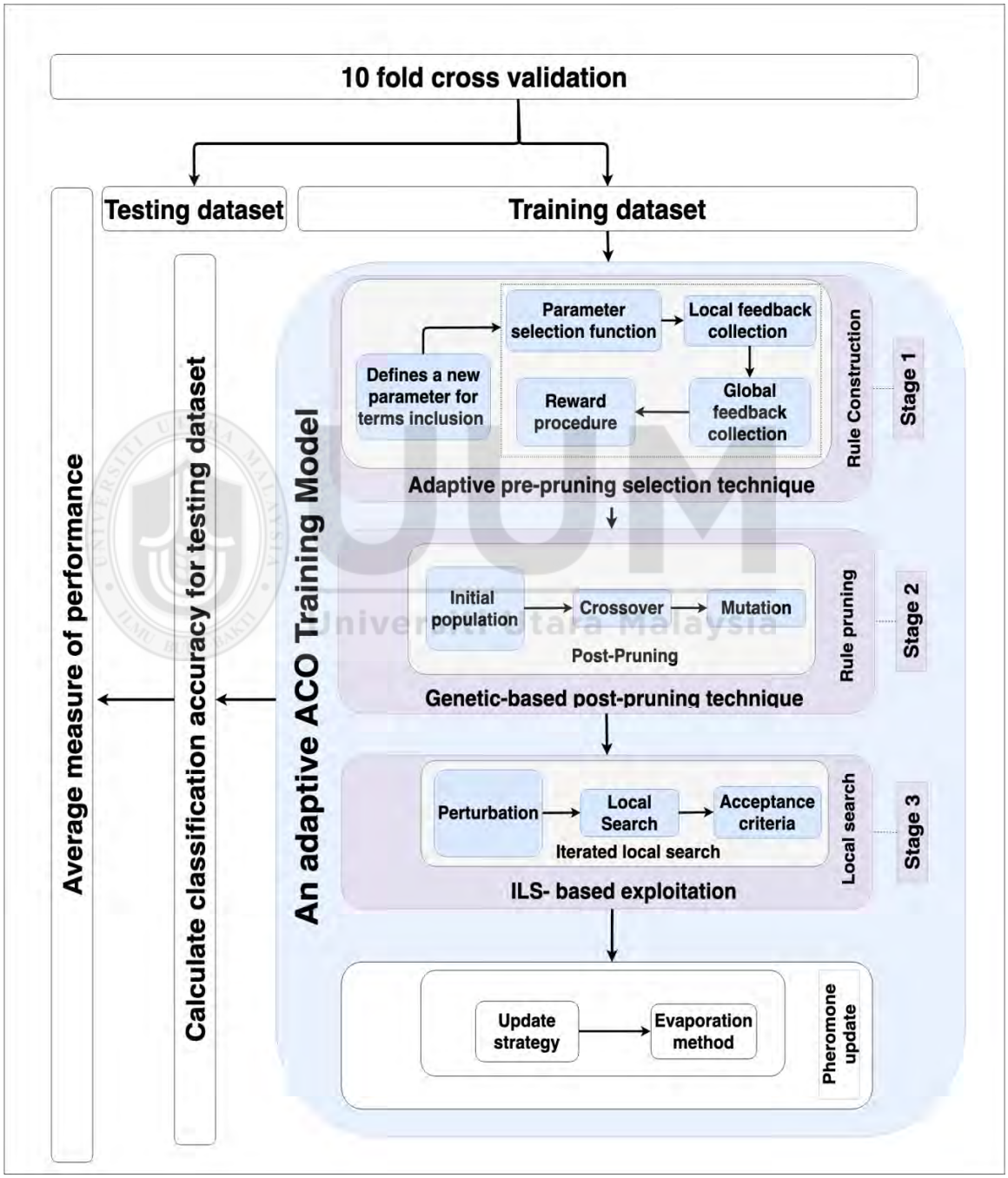

Figure 3.1. Macro view to the proposed AGI-AntMiner classifier

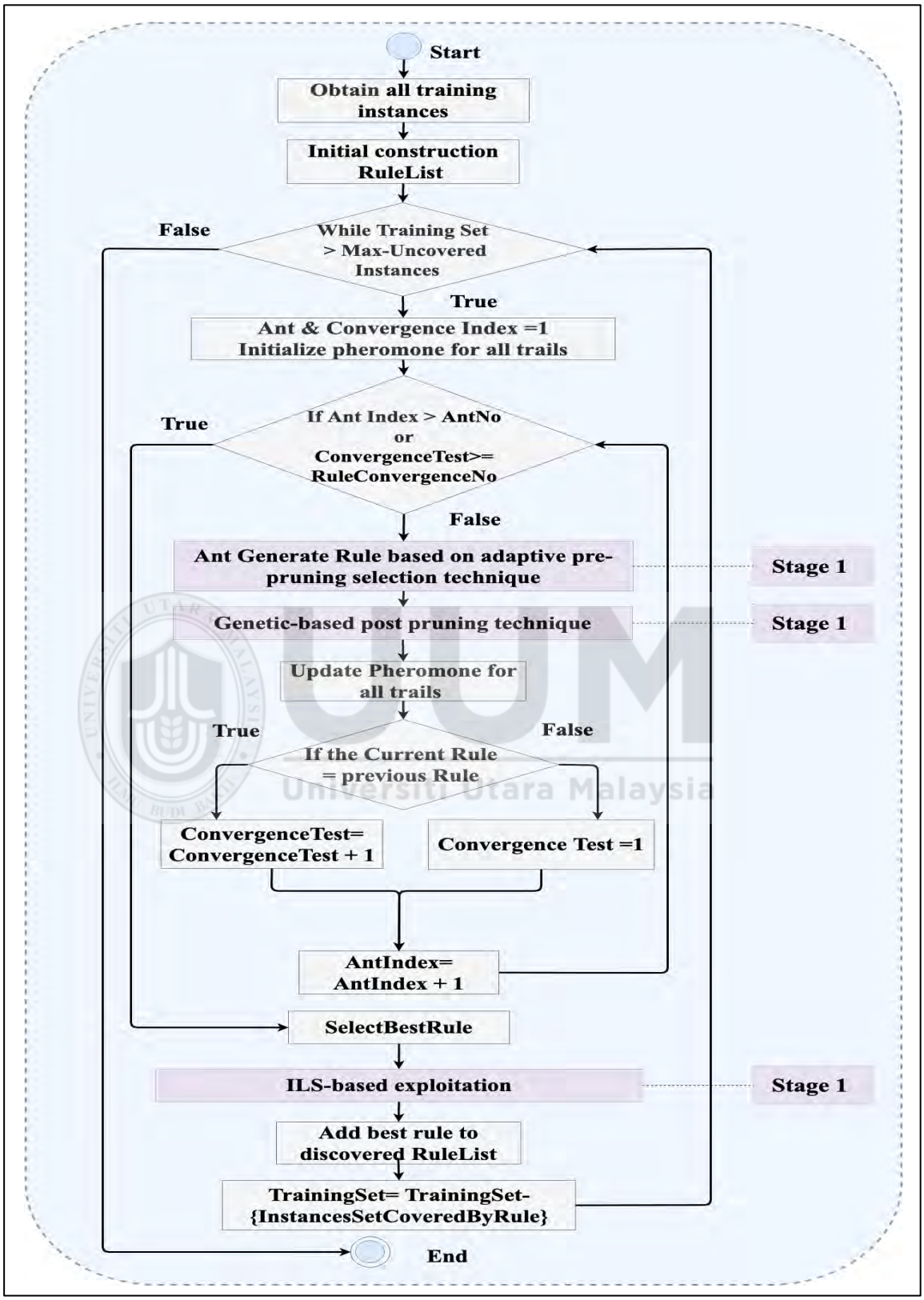

Figure 3.2. Micro view to the proposed AGI-AntMiner classifier

3.2.1 Adaptive Pre-pruning Selection Technique

A pre-pruning technique is proposed to select the appropriate threshold value based on the data. The proposed technique is the result of a modification to the work of Thangavel and Jaganathan (2007) and Tripathy et al. (2013) on TACO and MTACO classification algorithms that use the Ant-Miner classification algorithm with a fixed threshold value (i.e., 0.6 or 0.7). Figure 3.3 shows the architecture of the proposed A-AntMiner classification algorithm used in this research.

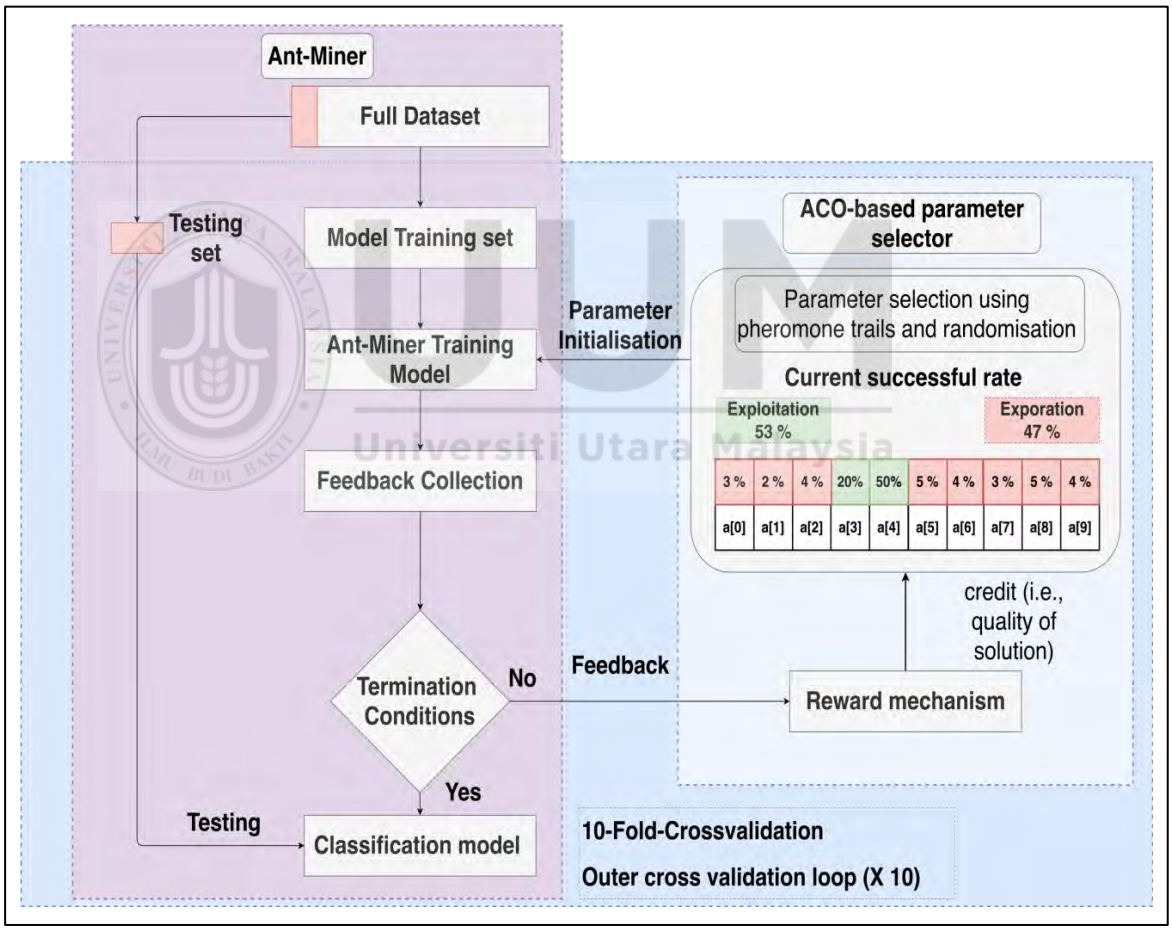

Figure 3.3. Architecture of the proposed A-AntMiner classification algorithm

In this research, adaptive pre-pruning selection technique is develop based on a new parameter called ζ to control inclusion/removal of terms in rule construction process.

In this technique the probability of each term will lie between 0 and 1 ($0.0 \leq \zeta \leq 1.0$) and uses the concept and principle of ACO algorithm to optimise the right ζ value. Subsequently, the Adaptive Ant-Miner (A-AntMiner) classification algorithm has been implemented based on the proposed strategy.

The ACO algorithm controls the selection of ζ value through four procedures: i) state transition rule, ii) parameter selection function, iii) reward procedure, iv) local and global feedback collections. A state transition probability function is used to simultaneously select the ζ parameter of the Ant-Miner based on either the amount of pheromone or randomisation. In this manner, the ACO can stochastically select a single threshold value. Then, the quality of each discovered rule (local feedback) and the quality of the elite rule (global feedback) collected during the learning process of the A-AntMiner classifier are determined. These feedback indicators will be transformed into reward assignments to guide the search process. The transition probability of ACO is used to determine whether the current ζ value must be dominated or not. The process will then be repeated in the learning process of the A-AntMiner classifier.

3.2.2 Genetic-based Post-pruning Technique

A new post-pruning technique is proposed on the basis of the search behaviour of the GA to find the optimal pruning rule and introduce a new classification algorithm named Genetic-based Ant-Miner (GA-AntMiner). The operational steps of the proposed technique are depicted in Figure 3.4 which consists of population initialisation, crossover, mutation, updating of the instance list, rule consequent

determination, rule quality calculation and ending with acceptance criteria rule. The technique has been developed to overcome the 'nesting effect' problem. The term elimination processes entail exchanges (crossover and mutation) between the same element in the current rule and in another temporary array.

Crossover is a critical genetic operator; in particular, a two-point crossover is used to exchange the terms between two arrays (rule array and another temporary array). With this approach, the pruned term can be added later to the pruning process. The main objective of using the mutation operator is to ensure diversity and avoid the search process from being trapped in local optima by preventing the generated rule pruning results from becoming similar to one another. The number of instances covered by the current rule is checked by using the 'update instance list' procedure because the majority of the cases covered by the pruned rule may change, as opposed to those cases covered by the original rules. Once the instance list becomes fully updated, the classifier selects the consequent ('Then' part) of the rule to specify the predicted class. This step is used to determine the rule consequent by assigning a majority class to the different instances covered by the rule. Then, the quality of the pruned rule is measured and compared with the quality of the original rule. If the quality of the pruned rule is better than the quality of the original rule, then the former replaces the latter. This procedure is repeated until the termination condition is met.

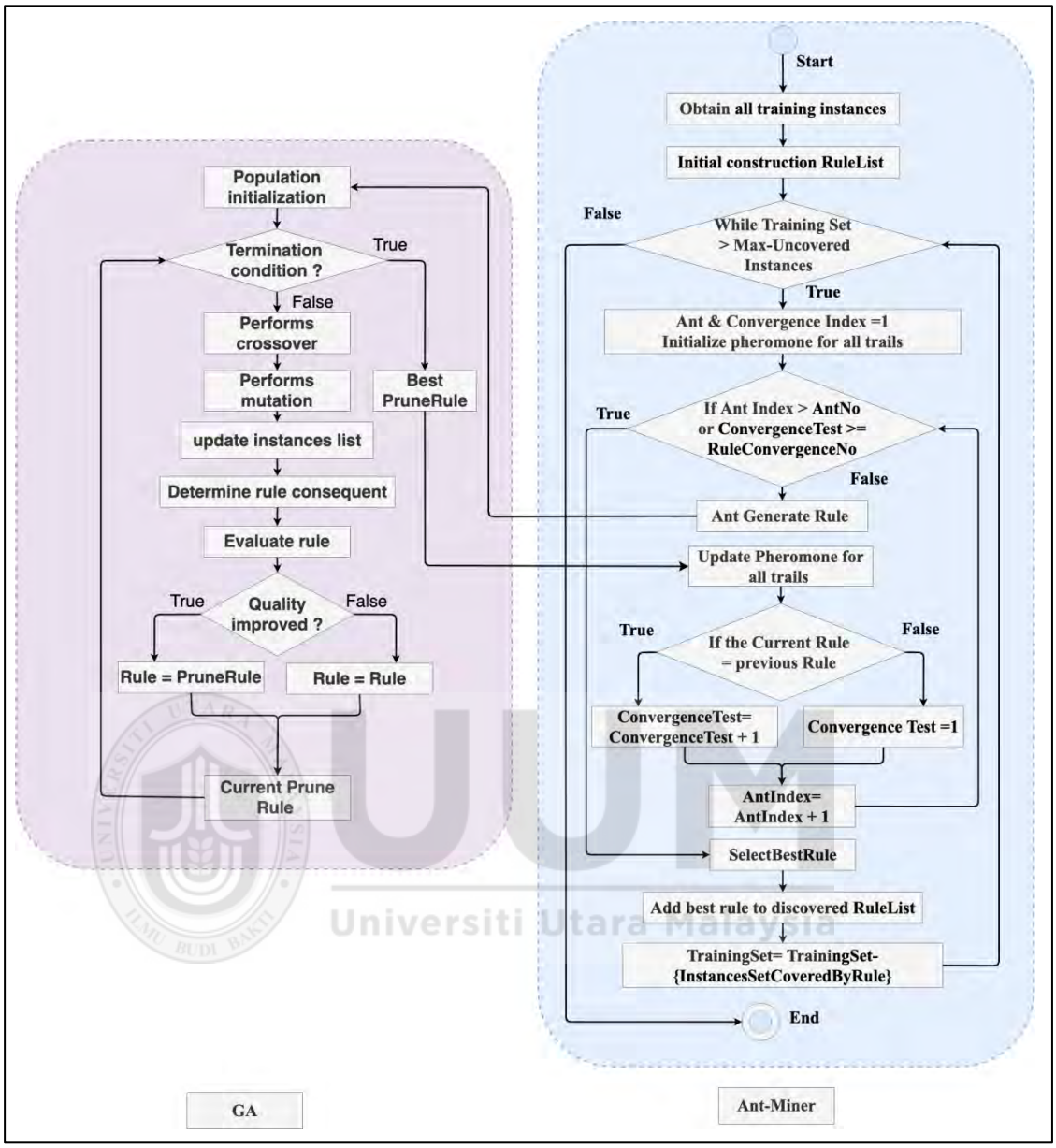

Figure 3.4. GA-AntMiner Classifier

3.2.3 Hybridisation with Iterated Local Search

Local search has not been explicitly mentioned as an Ant-Miner component. Questions about where and how components should be used properly to improve the search strategy needs to be answered initially before successful hybridisation can be

achieved, indicating that the hybridisation of the Ant-Miner with the ILS search is not a trivial task. The intensification of ILS will be used to improve the local exploitation of Ant-Miner search. Figure 3.5 shows the hybridization between the ILS algorithm and Ant-Miner algorithm to improve the list of best discovered rules.

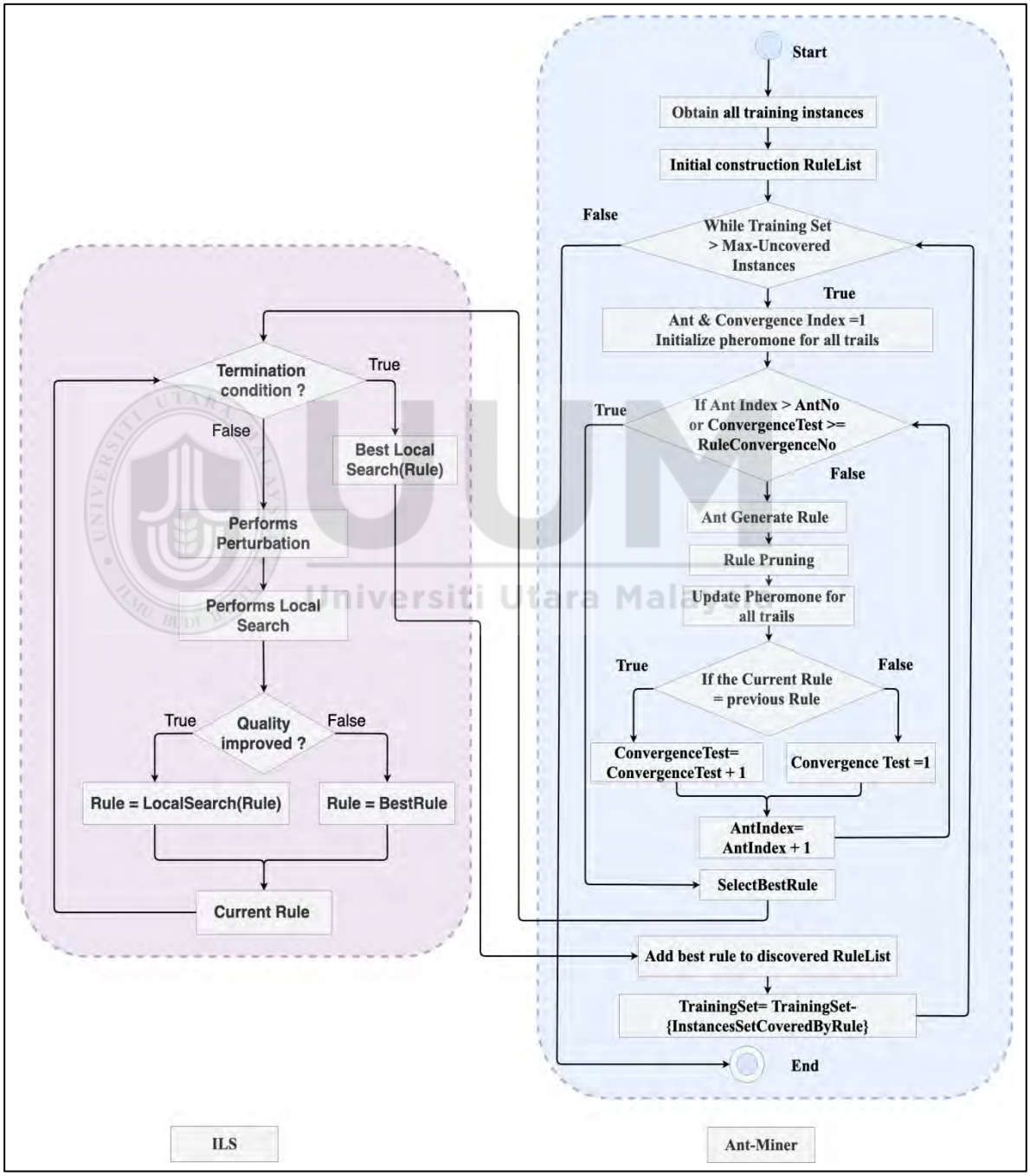

Figure 3.5. Hybridisation of Ant-Miner algorithm with ILS algorithm

ILS uses three components: i) rule perturbation to create a new starting rule from the best-found rules; ii) local search to find the best improvement in neighbourhood structures; and, iii) acceptance criterion to accept or reject the current accuracy improvement of the generated rule. This kind of local search assists the Ant-Miner to enhance the exploitation by focusing on promising areas of the search space.

3.2.4 Performance Evaluation

This section includes information on how the proposed algorithms have been evaluated. The algorithm is divided into two parts: one for comparison and the other for performance measurement. Three stand-alone algorithms corresponding to the three above mentioned stages have been evaluated separately and compared with existing classification algorithms to address the same issues in the literature. The comparison is performed by using benchmark datasets from the UCI repository.

The first and second stages focus on the previous ACO-based classification techniques and the most related modification procedure on rule pruning described in the literature. The compared classifiers include the original Ant-Miner (Parpinelli et al., 2002a), CAnt-Miner (Otero et al., 2008), ACO/PSO2 (Holden & Freitas, 2008), TACO-Miner (Thangavel & Jaganathan, 2007; Tripathy et al., 2013) and Ant-Miner with a hybrid pruner (Chan & Freitas, 2006a).

The CAnt-Miner classifier is an Ant-Miner version that can handle continuous attributes during training model construction (Otero et al., 2008). ACO/PSO2 is a hybrid swarm intelligence metaheuristic algorithm for rule-based classification (Holden & Freitas, 2008). The pruning procedures of ACO/PSO2 are applied to

discover the best rule from each iteration. ACO/PSO2 uses two pruning procedures. The first procedure is the original Ant-Miner pruning procedure and applied to the best rule discovered whose number of terms is less than a fixed value (i.e. 20). If the constructed rule entails more than 20 terms for each rule, then the pruning iterates to remove the unimportant or detrimental terms from the classification rule until the number is decreased to 20 terms. Subsequently, the Ant-Miner pruning procedure is implemented.

TACO-Miner provides a threshold criterion value based on the information gain of each term. If the information gain value associated with the term is below the threshold value, then the term will be rejected in the inclusion process. The threshold is considered a pre-pruning criterion and used to accept or reject terms (Thangavel & Jaganathan, 2007; Tripathy et al., 2013).

Another comparison for the proposed algorithm was with another classifier introduced by Chan and Freitas (2006). This classifier is a new hybridisation of the original Ant-Miner's rule pruner with another rule pruner. The authors focused on information gain of terms and a new parameter, *r*, which is a fixed value use to determine the acceptable number of terms in a rule. The first procedure is applied to each rule that exceeds the number of acceptable terms allowable in a rule. The number of terms in the selected rule is then reduced until its value reaches the *r* value. This selection method is implemented on the basis of information gain and roulette wheel values. Thereafter, the second procedure of the original rule-pruner procedure of the Ant-Miner is applied (Chan & Freitas, 2006a).

Lastly, the third stage compared the performance of our proposed hybrid classifier with the previous hybrid Ant-Miner classification algorithms (ACO/PSO2 and ACO/SA) which have been described in the literature. The ACO/SA is the hybrid Ant-Miner and SA for rule induction (Saian, 2013; Saian & Ku-Mahamud, 2012). This hybridisation with SA aims to overcome the problem of local optima by optimising the term selection in the rule construction process. Meanwhile, ACO/PSO2 is a hybrid swarm intelligence metaheuristic algorithm for rule-based classification (Holden & Freitas, 2008).

In this study, the three stand-alone classification algorithms are integrated into the proposed adaptive ACO algorithm for rules-based classification. The final stage of which compares the performance of the proposed algorithms with several other rule induction algorithms as discussed in Chapter 2 (Section 2.3 & Section 2.4) and the state-of-the-art classification algorithms called PART, and FURIA which are an industrial standard classification algorithm. PART uses a sequential covering strategy and, together with the decision tree algorithm, generates a set of discovered rules. At each iteration of the sequential covering strategy, the algorithm generates a decision tree, selects the leaf with the largest coverage to obtain rules and then ignores the rest of the tree (Vijayarani & Divya, 2011). Meanwhile, FURIA is a modification variant of the RIPPER classifier (see Chapter 2, Section 2.3).

The Ant-Mining parameters of Robu et al. (2016) and López-Ibáñez et al. (2016) are used in the present work to ensure that the selected classification algorithms can handle the same parameter values and provide a fair evaluation of the results. The parameters used for the Ant-Miner classifiers are listed in Table 3.1.

Table 3.1

Ant-Mining Experimental Parameters

Parameter	Description	Value
NA	Ant number	10
MCR	Minimum number of instances must cover by the rule	5
MUC	Maximum of uncovered instances	10
RC	Convergence limit	10
NI	Iterations number	10

The 10-fold cross-validation method (Han & Kamber, 2006; Parpinelli et al., 2002; Saian & Ku-Mahamud, 2012; Salama & Otero, 2013; Zhang & Sun, 2016) has been used to evaluate the performance of the proposed algorithms. The method randomly partitions the dataset into ten approximately similar-size subsets (folds), as shown in Figure 3.6. Witten and Frank (2005) has reported that the well-known 10-folds cross-validation has become the standard method in machine learning and data mining classification task. For instance, in different classification techniques, results shown that ten is the right number of folds to get the best estimate of error leaning on theoretical evidence for support. Each proposed algorithm is then trained and tested ten times with different subsets (a single subset is used each time) during the testing and remaining stages of the training process. Furthermore, each fold in the cross-validation for the stochastic classifiers is run ten times. Meanwhile, the deterministic classifiers are run just once. In using this approach, all instances in the dataset will be part of the training and the testing. The 10-fold cross-validation is considered a good

method for the performance evaluation of classifiers due to its variance and relatively low bias (Han & Kamber, 2006a).

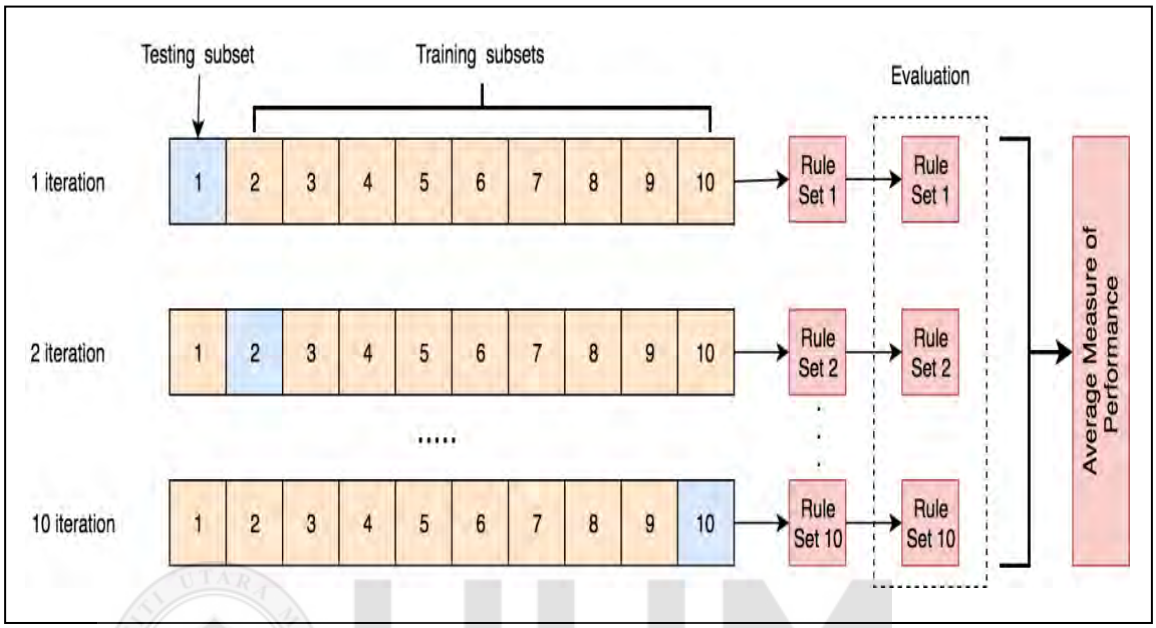

Figure 3.6. 10-fold cross-validation method

Performance evaluation has been carried out using the following three criteria: classification accuracy; simplicity of the rule list (rule number); and, model size (number of terms for each rule). The first criterion refers to the rule set provided by each proposed algorithm that will be applied to the test instances. The process is unobservable during training, and the class labels are unknown. The new instances covered by the rule are applied, and each instance is given a class label as predicted by the rule. Classification accuracy is determined on the basis of the number of test instances correctly and incorrectly classified by the rule set. The numbers will be tabulated in a matrix similar to that shown in Table 3.2.

Table 3.2

Classification Accuracy Evaluation Matrix (Confusion Matrix)

Actual class	**Predicted class**	
	Class 1	**Class 0**
Class 1	TP	FN
Class 0	FP	TN

The count number of all instances from class 1 and correctly predicted by the rule set as class 1 is represented as *TP* while *TN* is the count number of all instances from class 0 and correctly predicted by the rule set as class 0. The count number of all instances from class 1 but incorrectly predicted by the rule set as class 0 is represented by *FN* while *FP* is the count number of all instances from class 0 but incorrectly predicted by the rule set as class 1. The accuracy of the rule set is computed as the number of correct predictions divided by total number of all predictions, as shown in Equation 3.1.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP}+\text{FN}+\text{FP}+\text{TN}} \tag{3.1}$$

The average and standard deviation values of all testing results in the 10-run by using the 10-fold cross-validation method will be reported to determine the accuracy performance of the Ant-Miner classifier for each dataset.

The rule list to be discovered is measured, as implied in the literature according to the total number of rules in the discovered lists. The third criterion of the performance evaluation is the simplicity of classification model. This aspect is important because the goal is to discover knowledge that can be easily validated and

interpreted by experts. In the Ant-mining literature, the model size is measured according to the total number of terms in the discovered rules. Thus, the results are based on the average, standard deviation and rank that have been measured to determine the average number of rules, and the average number of terms in all rules for the 10-run by using the 10-fold cross-validation method.

Furthermore, in order to determine the appropriate balance between classification accuracy and compressibility of the classification model, the nonparametric Friedman test is conducted with the Holm post-hoc test (Demsar, 2006; Liang et al., 2016; Salama & Freitas, 2012; Yang et al., 2016; Yang et al. , 2015). In this manner, the performance of all classifiers is examined in accordance with classification accuracy and simplicity can be observed. Then, the result of the nonparametric Friedman test with Holm's post-hoc test is used to determine the average classification accuracy rank versus the average number of discovered rule rank, and the average model size rank for all classifiers. This test aims to find the optimal classifier that balances different objectives. The test is conducted to rank the algorithms' performance for each dataset in descending order. A low rank implies good algorithm performance (Yang et al., 2016). The test is used to rank the best classifier that balances between classification accuracy and model size. For example, classifier A dominates classifier B if and only if the following two conditions are true: the first condition, A, is not worse than B with respect to both objectives, i.e., classification accuracy and the model size. The second condition, A is strictly better than B at least in one objective (Otero et al., 2013a). Thus, classifier A will be optimal only if it is not dominated by any other classifiers (Ghosh & Nath, 2004).

3.3 Experimental Dataset

Benchmark datasets are used to compare the proposed algorithms in this research with the commonly related Ant-Mining classification algorithms in the literature. The benchmark datasets are selected in accordance with Ant-mining literature. These benchmarks are secondary datasets that have been chosen from UCI (Dua & Karra, 2017). The datasets vary in terms of number of instances (ranging from 150–8124), attributes (range of 4–60) and class labels. In addition, the attributes consist of categorical and continuous types. The selected datasets are as follows: Balance Scale, Breast Cancer (Ljubljana), Breast Cancer (Wisconsin), Credit-a, Credit-g, Diabetes, Heart (Cleveland), Heart (Statlog), Hepatitis, Ionosphere, Iris, Lymphography, Mushroom, Segment, Sonar and Tic-Tac-Toe. The main features of each dataset are summarised in Table 3.3. The features include the following: name of datasets, number of instances, number of attributes, number of values in each class attribute and type of attributes.

Table 3.3

Main Dataset Features to be Used in the Experiments

Datasets Name	Instances Number	Attributes Number	Classes Number	Type of Attributes
Balance Scale	625	4	3	Categorical
Breast Cancer (Ljubljana)	286	9	2	Categorical

Breast Cancer (Wisconsin)	699	9	2	Continuous
Credit-a	690	15	2	Categorical, Continuous
Credit-g	1000	20	2	Categorical, Continuous
Diabetes	768	8	2	Continuous
Heart (Cleveland)	303	13	5	Categorical, Continuous
Heart (Statlog)	270	13	2	Categorical, Continuous
Hepatitis	155	19	2	Categorical, Continuous
Ionosphere	351	34	2	Continuous
Iris	150	4	3	Continuous
Lymphography	148	18	4	Categorical, Continuous
Mushroom	8124	22	2	Categorical
Segment	2310	19	7	Continuous
Sonar	208	60	2	Categorical, Continuous
Tic-tac-toe	958	9	2	Categorical

The first dataset used in our experiment is the Balance Scale, developed by Siegler to model psychological experiments, consists of 625 instances with four categorical attributes. The class attribute of this dataset has right, left or balanced values.

The Breast Cancer (Ljubljana) dataset was created by Milan Soklic and Matjaz Zwitter from the Institute of Oncology, Ljubljana, Yugoslavia. The dataset consists of 286 instances and nine attributes. All attribute types are categorical. The class

attribute of this dataset has two values: ‘recurrence events’ and ‘no recurrence events’.

The Breast Cancer (Wisconsin) dataset, created by Dr. William H Wolberg from the University of Wisconsin Hospitals, consists of 699 instances and nine continuous attributes. The class attribute of this dataset has two values: benign and malignant.

The Credit-a dataset was created by Quinlan from credit card applications. This dataset consists of 690 instances with 15 categorical and continuous attributes. The class label of this dataset has two values: positive (307 instances) and negative (383 instances).

The Credit-g dataset about the German credit was created by Dr. Hans Hofmann and consists of 1000 instances with 20 categorical and continuous attributes. This dataset has two classes: good and bad.

The Diabetes dataset was based on the female-patient information of Pima Indian Heritage. In the dataset, the patients are at least 21 years old. This dataset has been used to investigate if the patients’ diabetes information accords with the World Health Organisation criteria. The dataset consists of 768 instances with eight continuous attributes and divided into two classes: positive and negative.

The Heart (Cleveland) dataset on heart diseases was developed by Cleveland Clinic in 1988, a non-profit medical academy. The dataset includes 303 instances and 13

attributes. It has eight categorical and five continuous attributes. The class attribute consists of five values.

The Heart (Statlog) dataset on heart diseases consists of 270 instances with 13 categorical and continuous attributes divided into two classes absent (150 instances) and present (120 instances).

The Hepatitis dataset was developed by Gong from the Bojan Cestnik, Jozef Stefan Institute, Carnegie-Mellon University, Ljubljana, Yugoslavia. The dataset comprises 155 instances with 19 attributes. Thirteen attributes are categorical and six attributes are continuous. The class attribute has two values: die and live.

The radar dataset called Ionosphere was collated by a certain system in Goose Bay, Labrador, consisting of 351 instances and 34 continuous attributes with two classes: good (225 instances) and bad (126 instances).

The Iris dataset was created by Fisher in 1988. This dataset has 150 instances with four attributes, all of which are continuous. The dataset has three values for the class attribute Iris Virginica, Iris Virginica and Iris Setosa.

The Lymphography dataset was developed at the University Medical Centre, Institute of Oncology, Yugoslavia at 1988. This dataset consists of 148 instances divided into four classes: normal, metastases, malign lymph and fibrosis. The dataset consists of 18 attributes that are categorical and continuous.

The Mushroom dataset, which includes descriptions of hypothetical samples corresponding to species of gilled mushrooms, was developed by Lincoff and Schlimmer, as reported in the Audubon Society Field Guide to North American Mushrooms. The dataset has 8124 instances and 22 categorical attributes. The class attribute consists of two values: edible and poisonous.

The Segment dataset was developed by the Vision Group of the University of Massachusetts in. It consists of 2310 instances with 19 continuous attributes. The classes in this dataset are equally divided into seven classes: window, sky, foliage, brick face, path, grass and cement.

The Sonar dataset, developed by Gorman and Sejnowski, includes 208 instances. This dataset was used to discriminate between sonar signals bounced off a metal cylinder and those bounced off a roughly cylindrical rock. This dataset consists of 60 categorical and continuous attributes. The attribute is divided into two classes: rock and mines.

The last dataset utilised in this research is the Tic-Tac-Toe dataset. This dataset, which was developed by David Aha, encodes the complete set of possible board configurations at the end of tic-tac-toe games. The dataset has 958 instances, and each instance corresponds to one tic-tac-toe square. The dataset consists of nine categorical attributes and a class attribute. The class attribute has two values: negative and positive.

The application domains of the abovementioned datasets are categorised into six types: life, financial, physical, social, image and gaming. The highest percentage (56.25%) belongs to the life domains, followed by the financial and physical application domains (12.5%). The remaining social, image and gaming domains equally have 6.25%.

In accordance with the number of attributes in the dataset, the size of the dataset can be categorised into four main types: small, medium, high and very high (Kudo & Sklansky, 2000). The majority (75%) of the datasets used in the experiment are classified as small with sizes in the range of 0–19 attributes. Meanwhile, 18.75% of the datasets have medium sizes in the 20–49 attribute range. The remaining datasets (6.25%) are classified as high, with attributes ranging from 50 to 100.

In terms of missing values, the majority (62.5%) of the datasets — Balance Scale, Credit-g, Diabetes, Heart (Statlog), Ionosphere, Iris, Lymphography, Segment, Sonar and Tic-Tac-Toe — used in our experiment have no missing values. By contrast, 37.5% of the datasets have missing values, and these are the Breast Cancer (Ljubljana), Breast Cancer (Wisconsin), Credit-a, Heart (Cleveland), Hepatitis and Mushroom datasets.

A dataset usually entails different formats, particularly categorical and continuous formats, and the difference lies in the number of values they can take (Fayyad & Irani, 1992; Lavanya & Rani, 2011). Categorical attributes have a finite number of particular values, whereas continuous attributes have infinite possibilities for the

number of values, i.e., temperature or weight (Witten et al., 2016). Discretisation plays an important role in knowledge discovery and DM which, in turn, increase the performance of classifiers in terms of accuracy and learning time. Most studies show that rules with discrete values are closer to the knowledge-level aspect, easier to use and more comprehensible than the rules with continuous values. The traditional method of rule induction algorithms, including Ant-Miner, cannot cope with continuous attributes in its rule discovery process. Thus, a discretisation of the continuous attributes is needed either before the rule construction process (Liu et al., 2002; Liu & Lin, 2008; Otero et al., 2008; Parpinelli et al., 2002) or during the rule learning process (Otero et al., 2008, 2009; Rajpiplawala & Singh, 2014).

In this study, the discretisation will be carried out as a pre-processing step by using the C4.5-Disc method, which uses the well-known C4.5 algorithm for discretizing continuous attributes (Zahiri, 2012). In essence, the C4.5 will be applied for each continuous attribute in the dataset. The algorithm first extracted the continuous attribute and the target (class) attribute from the dataset. Then, C4.5 algorithm build a decision tree by using an entropy-based metric for that extracted dataset (the continuous attribute and the target attribute), to determine the partitions for categorical intervals. Each path in the decision tree represented the definition of a categorical interval generated by C4.5. Witten and Frank (2005) and Quinlan (2014) have described further details on C4.5. discretization method on numeric attributes. The method has also been used in the experiment of the Ant-Mining algorithms (Parpinelli et al., 2002).

3.4 Summary

The research aims to develop an adaptive ACO algorithm for rules-based classification by following the same experiment methodology used to develop the Ant-Miner variants. The stages and methods in the research framework have been successfully executed to achieve the objectives of the study. The implementation of the proposed methods is detailed in Chapter 4.

CHAPTER FOUR
AN ADAPTIVE ANT COLONY OPTIMISATION ALGORITHM FOR RULES-BASED CLASSIFICATION

4.1 Introduction

Ant-Miner suffers from overfitting and easily falls into local optima. Such a drawback leads the search restricted to only global search which produce a vulnerable rule set. To ensure the accuracy of classifier, an adaptive pre-pruning technique is redesigned to dynamically select the appropriate threshold based on the quality of the rules; and post-pruning is improved to be able to add/remove the terms during the pruning process; and finally, improved the exploitation mechanism by using a multiple-neighbourhood structure.

The development steps of the proposed adaptive ACO classification algorithms are explained in detail in this chapter. Section 4.2 introduces the A–AntMiner as a classification algorithm with pre-pruning technique to select the appropriate terms during the learning process. Section 4.3 introduces the GA–AntMiner classification algorithm that finds feasible pruning rules and overcomes the limitation of the existing post-pruning procedure. Section 4.4 presents the implementation of the ILS-AntMiner algorithm according to the concepts and components of the ILS algorithm to improve classification rule quality. Section 4.5 focuses on the integration of the above-mentioned modifications to form the proposed AGI-AntMiner algorithm for rules-based classification. Finally, Section 4.6 summarises the chapter.

4.2 Adaptive–AntMiner Classifier

The adaptive pre-pruning selection technique describes the application of ACO concept which collects the feedback from the classification process to adjust the threshold of pre-pruning in the Ant-Miner classifier. This dynamic adjusted threshold is called importance rate (ζ) based on probability of each term. The probability is computed by the heuristic and the pheromone information. The adaptive pre-pruning selection technique controls the importance rate (ζ) to prevent the irrelative terms during the rule's construction phase of the classifier according to its probability. To understand the strategy of ζ selection, three vectors have been developed, a vector of threshold values is denoted as $ParametersValue(v_n) = \{ \upsilon_1, \upsilon_2, \upsilon_3,..., \upsilon_n\}$ which represents various ζ values, and another vector of probabilities for each parameter value υ_n, denoted as $ParametersProbabilities(v_n) = \{p(\upsilon_1), p(\upsilon_2), p(\upsilon_3) ,..., p(\upsilon_n)\}$, which determines the selection probability of each parameter value, and a vector to collect the local and global quality of the threshold values, denoted as $feedbackCollection\ (v_n) = \{q(\upsilon_1), q(\upsilon_2), ..., q(\upsilon_n)\}$. When each threshold value, υ_n, is used, then its current quality $q(\upsilon_n)$ is updated. An example of the contraction graph of A-AntMiner for a simplified Breast Cancer (Wisconsin) dataset is presented in Figure 4.1 to better understand the classification process,

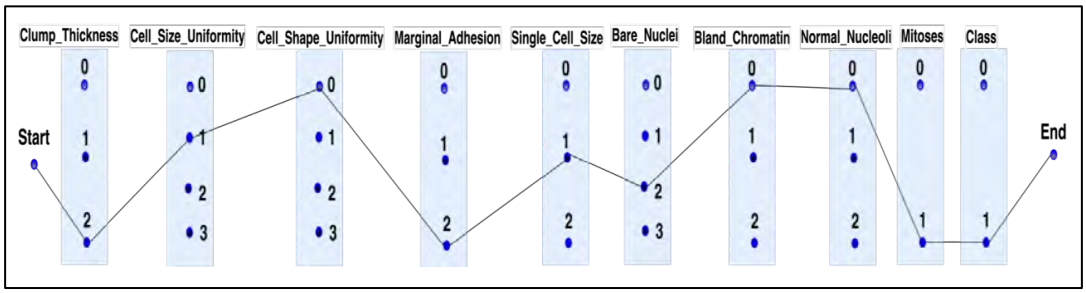

Figure 4.1. A–AntMiner Contraction Graph for Simplified Breast Cancel Dataset

In this graph, all terms compensate with integer numbers (e.g., Clump_Thickness = 0, 1, or 2 rather than '\'(-inf-4.5]\", '\'(4.5-6.5]\", or '\'(6.5-inf)\"). The constructed graph of A-AntMiner is fully connected, where the rule term is a conjunction of the form Attribute=Value. Then, the rule will be discovered while the ant select the path shown in boldface (refer to Figure 4.1.). Figure 4.2 provides a low-level description of the A–AntMiner algorithm.

```
    Input: arff dataset
    Output: classification rule
1   TrainingSet= {all Training instances};
2   InitialConstructionRuleList =[];
3   DiscoveredRuleList=[];
4   WHILE(TrainingSet>MaxUncoveredInstances)
5   AntIndex=1; ConvergenceTest=1;
6   InitializePheromone;
7   InitializeFeedbackCollection;
8   ParameterSelection; /*   select ζt value determine the importance or the strength of
    each term */
                // Rule construction start here
9   REPEAT
10              WHILE (TermsNotSelected==True)
11                      IF (TheProbabilityOfChosenTermij is >= ζt)
12                      THEN AddTerm;
13                      ELSE RejectTermFromInclusion;
14                      END IF;
15              END WHILE
                // Rule construction end here
16  LocalFeedbackCollection;   // Local Feedback Collection
17  RulePrune;    // Rule pruning
18  UpdatePheromone;    // Pheromone updating
19  IF(CurrentRule=PreviousRule)
20  THEN ConvergenceTest = ConvergenceTest + 1;
21  ELSE ConvergenceTest = 1;
22  END IF AntIndex = AntIndex + 1;
23  UNTIL(AntIndex>=AntNo) OR (ConvergenceTest>=RuleConvergenceNo)
24  SelectBestRule;
25  GlobalFeedbackCollection;     // Global Feedback Collection
26  AddBestRule to DiscoveredRuleList;
27  TrainingSet = TrainingSet-{InstancesSetCoveredByRule};
28  END-WHILE
```

Figure 4.2. A–AntMiner classifier pseudocode

The parameters used for the A-AntMiner classifier are adopted from Robu et al. (2016) that, introduced a research of Ant-Miner parameter values and López-Ibáñez et al. (2016) which, shows the default ACO parameter values and listed in Table 4.1.

Table 4.1

A-AntMiner Experimental Parameters

Parameter	Description	Value
NA	Ant number	10
MCR	Minimum number of instances must cover by the rule	5
MaxUncoveredInstances	Maximum of uncovered instances	10
ConvergenceTest	Convergence test	10
NI	Iterations number	10
ρ	Evaporation rate	0.32
α	Alpha	1
q0	Exploration/exploitation parameter	[0.1,...0.9]

In Figure 4.2, the first step of the A-AntMiner starts by randomly partition the dataset into ten approximately similar-size subsets (folds). The training step will use nine of them while the remaining one will be used in the testing step. A-AntMiner initialised the training set with all instances in the nine folds of the training dataset. The second and the third steps, initialise the *ConstructionRuleList,* and the *DiscoveredRuleList*; The fourth step is WHILE loop of A-AntMiner, corresponds to a number of executions of the REPEAT-UNTIL loop, which will discover one classification rule. This rule is added to the list of discovered rules *ConstructionRuleList* and the training instances that are correctly covered by this rule

(i.e., instances satisfying the rule antecedent and having the class predicted by the rule consequent) are deleted from the *TrainingSet*. This process is performed iteratively while the number of uncovered training instance is more than a pre-specified parameter, called *MaxUncoveredInstances*, which is ten.

The fifth step is initialise the *AntIndex* =1 and *ConvergenceTest* =1; The sixth step is initializing the pheromone matrix with a small amount of pheromone that inversely proportional to the number of terms in the dataset using the following Equation:

$$\tau_{\mathrm{n}}(t=0) = \frac{1}{\sum_{i=1}^{a} bi} \tag{4.1}$$

where *a* is the total number of attributes in the dataset, and *bi* is the total number of terms in each attribute *ai*. The seventh step is to initialise the feedback collection vector with a small value using Equation (4.2).

$$\tau_{\mathrm{n}}(t=0) = \frac{1}{\mathrm{n}} \tag{4.2}$$

where *n* is the size of feedback vector.

Parameter Selection Function

Step eight is the selection probability. This is to select ζ value from vector of threshold values. The values of ζ lies between [0,...,1]; Each $P(v_n)$ is calculated using a modified version of state transition rule given in Equation (4.3).

$$P(v_n)=\begin{cases} arcmax_{vn} \in P \quad if\ q < q_0 \text{ (exploitation)} \\ R, \qquad\qquad otherwise \text{ (exploration)} \end{cases} \tag{4.3}$$

where q_0 is a parameter of the controlling state transition rule which will gradually increase between [0.1,...,0.9]. The maximum (*max*) and minimum (*min*) values of the range are identified experimentally in this work by using the following Equation.

$$q_0 = (max\ )\ e^{-\mathrm{t}\frac{\ln(a)}{a}} \tag{4.4}$$

where a is the number of ants and t is the index of the current ant in the ACO. This qquation is set as q_0 with *min* value to increase the exploration of the ACO algorithm at the beginning of the construction process. During the construction process, each ant is assigned its own q_0 value. Over time, the value of q_0 will gradually increase and will stop when it reaches the value of *max*. *q,* which is a random value uniformly distributed in between [0,...,1]. *R* is a randomly chosen threshold value which is a pure exploration instead of the biased exploration calculated by the original formula of the state transition rule proposed by Dorigo and Gambardella (1997). Figure 4.3 shows the process of controlling the state transition rule by the q_0 parameter.

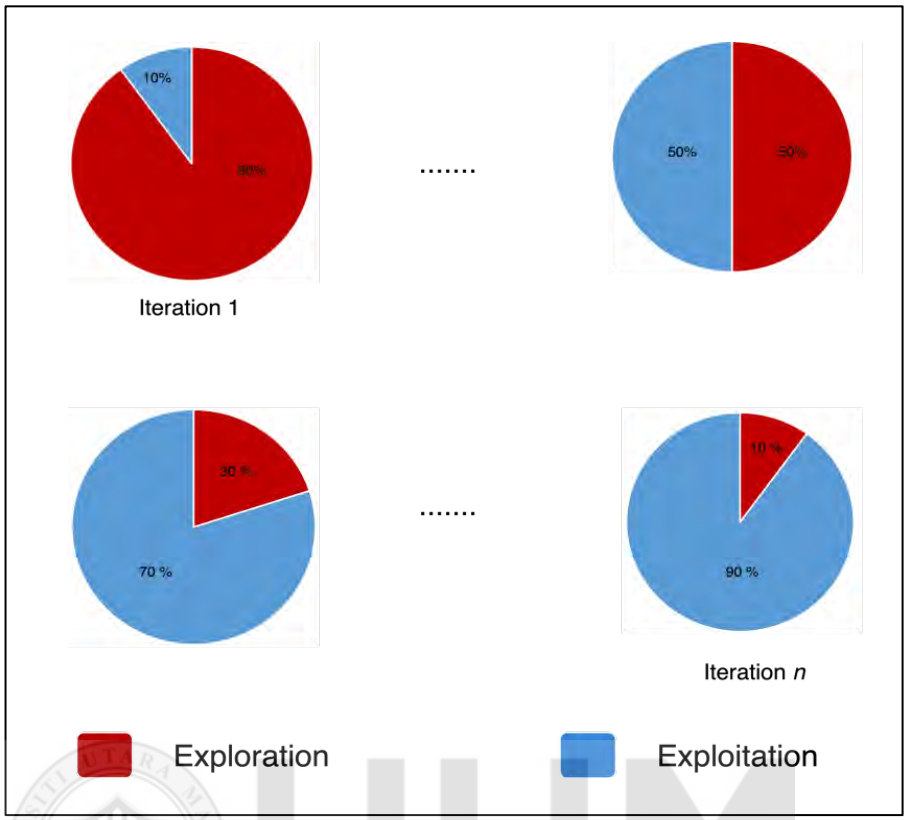

Figure 4.3. Process of controlling the state transition rule

P_n is the probability of selecting a specific value from available values and is simplified in Figure 4.4, which is calculated by using Equation 4.5.

$$P_n = \frac{[\tau_{n(t)}]}{\sum_{n=1}^{l}[\tau_{n(t)}]} \tag{4.5}$$

where $[\tau_{n(t)}]$ is the amount of the quality associated to each threshold value to probabilistically select one of them.

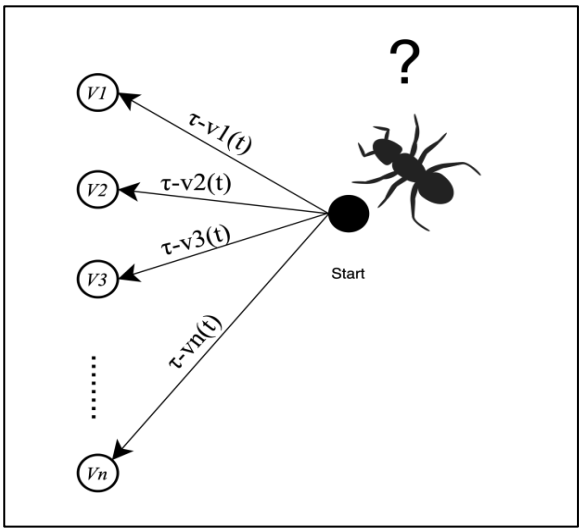

Figure 4.4. Construction graph with probabilities employed by the adaptive pre-pruning selection technique

An example of the result for probabilities is shown in Table 4.2. The first column represents the available threshold values while the second column represents the quality of the threshold values based on the feedback collected locally and globally. The last column represents the probability of obtaining a specific threshold value. Thus, the use of the probabilities was able to help in deriving conclusions about the causes of the successful or unsuccessful performance in different threshold values. For example, the success rate of the threshold value is υ7, which is higher than the other. This means that threshold value υ7 has a higher probability of being selected.

Table 4.2

Conditional probabilities of threshold values

Threshold	Quality	Probabilities
v1	0.01	0.0034
v2	0.5	0.1695
v3	0.22	0.0746
v4	0.3	0.1017

v5	0.4	0.1356
v6	0.2	0.0678
v7	0.9	0.3051
v8	0.21	0.0712
v9	0.1	0.0339
v10	0.11	0.0373

In step nine, the A-AntMiner classifier will start with training the data instances to discover one classification rule. The rule is then added to the discovered rule list, *ConstructionRuleList*. This process consists of three main procedures, namely rule construction, pruning and pheromone updating.

Rule Construction

Step number ten to step number fifteen represent the rule construction phase, where each ant starts to select terms to be added to the rule antecedent. The ant will add one term at a time to improve the classification accuracy according to its ζ value. If the probability of the selected term is greater than or equal to the ζ, then the term will be added. Otherwise, the term will be excluded. The probability of term selection to be added to the current rule is given by Equation (4.6) (Parpinelli et al., 2002).

$$\text{Probability} = \frac{[\tau_{ij(t)}]\ [\eta_{ij}]}{\sum_{i=1}^{a} xi \cdot \sum_{j=1}^{bi} [\tau_{ij(t)}]\ [\eta_{ij}]} \tag{4.6}$$

where $[\tau_{ij(t)}]$ is the amount of pheromone concentration for each term at iteration (t), $[\eta_{ij}]$ is the problem depending upon heuristic function, *a* is the attribute number in the dataset, *bi* is the number of different values for each attribute, and *xi* is set to 1 if attribute is not visited yet by the ant, and 0 otherwise.

In addition, the heuristic function value is used together with the pheromone value to decide on selection of term. In the Ant-Miner, the heuristic function is inspired by information theory. The Ant-Miner computes the amount of information contained in each term (entropy). The heuristic function is given by Equations (4.7) and (4.8) (Parpinelli et al., 2002).

$$\eta_{ij} = \frac{log_2\, k - H(W|A_i = V_{ij})}{\sum_{i=1}^{a} xi \cdot \sum_{j=1}^{bi} (log_2\, k - H(W|A_i = V_{ij}))} \tag{4.7}$$

$$H(W|A_i = V_{ij}) = -\sum_{w=1}^{k} \left[\frac{P(W|A_i = V_{ij})}{T_{ij}}\right] * log_2 \left[\frac{P(W|A_i = V_{ij})}{T_{ij}}\right] \tag{4.8}$$

where w is the class attribute, k is the number of classes, $P(W|A_i = V_{ij})$ is the partition containing the instances where attribute Ai has value Vij with class w, $|Tij|$ is the total number of instances in partition Tij (instances where attribute Ai has value Vij), a represents the total number of attributes, and bi is the number of values in the particular of attribute i.

This process is repeated until all attributes have been used, or the minimum number of instances is covered by the current rule. Once the rule antecedent task is completed, the A-AntMiner selects the consequence class label of the rule by assigning the majority class among the instances covered by the rule and calculates its quality.

Local Feedback Collection

In step sixteen, the adaptive parameter selection strategy starts to collect feedback. The A-AntMiner construction rule applies the local feedback collection according to Equation (4.9) (Dorigo & Gambardella, 1997).

$$\tau_{n(t+1)} = (1 - \rho).\tau_{n(t)} + \rho.Q(t) \quad (4.9)$$

where the value of ρ indicates the quality evaporation rate to avoid unlimited accumulation on specific parameter value, and *Q (t)* is the quality of the discovered rule.

Rule Pruning

In step seventeen, the discovered rule will then undergo pruning procedure which aims to avoid the overfitting problem by reducing the size of the discovered rules to increase comprehensibility. Figure 4.5 shows an example of discovered rule constructed from Breast Cancer (Wisconsin) dataset by A-AntMiner classifier.

IF Clump_Thickness = '\'(6.5-inf)\" ***AND*** Cell_Size_Uniformity = '\'(-inf-1.5]\" ***AND*** Cell_Shape_Uniformity = '\'(2.5-4.5]\" ***AND*** Bare_Nuclei = '\'(2.5-5.5]\" ***AND*** Normal_Nucleoli ={'\'(-inf-2.5]\ ***AND*** Normal_Nucleoli {'\'(-inf-2.5]\" ***THEN*** 'malignant'

Rule quality = 0.1892

TP =9; *FP* =16; *FN* = 20; *TN* = 25 ;

Figure 4.5. Example of Complete Rule

The pruning procedure prunes one term at a time to improve the rule quality. The procedure loops until no more improvement occurs or only one term is left under the rule. The class value of the dataset can potentially change during this procedure because the majority of the classes in the cases covered by rule pruning may change compared with cases covered by the original rules. Figures 4.6 shows an example of rule pruning effect on a same rule obtained in Figure 4.5.

IF Clump_Thickness = '\'(6.5-inf)\" ***THEN*** 'malignant'

Rule quality = 0.678261

TP =20; FP =7; *FN* = 5; *TN* = 39;

Figure 4.6. Example of Pruned Rule

Pheromone Updating

In step eighteen, the pheromone will be updated after rule construction and prune procedures. The approach of pheromone updates has two basic steps. First, increase the amount of the pheromone in all terms appearing in the rule according to rule quality by Equations (4.10) and (4.11) (Parpinelli et al., 2002).

$$\tau_{ij(t+1)} = \tau_{ij(t)} + \tau_{ij(t)}.Q \quad (4.10)$$

$$Q = +\frac{TP}{TP+FN} * \frac{TN}{FP+TN} \quad (4.11)$$

where *TP* is the number of instances covered by the discovered rule and has the class predicted by the rule. *FN* is the total number of instances covered by the discovered rule and has a class different from the class predicted by the rule. *TN* is the total

number of instances not covered by the discovered rule and does not have the class predicted by the rule. *FP* is the total number of instances not covered by the discovered rule but has class predicted by the rule. Second, evaporating every term does not occur in the rule, which is achieved by normalising unused terms. The rule quality by Equation (4.11) has also used in rule pruning, the local feedback collection and global feedback collection stages as reward procedure to keep track of the best rule. Therefore, another ant will build its rule derived from the updated amount of pheromone. The process is complete until one of the following stopping criteria is met. In the first criterion, the number of ants should be equal to or greater than the number of discovered rules. The second criterion depends on the number of the rule convergence threshold, where the ant starts to converge by constructing a rule similar to that one constructed before. The best among all construction rules will be added to the list of discovered rules.

Steps nineteen, twenty and twenty one will check if the current Ant has discovered a rule that is similar to rule that constructed by the previous Ant *CurrentRule* = *PreviousRule*. The number of convergence test parameter *ConvergenceTest* =10.

In steps twenty two, and twenty three, the current iteration of the REPEAT loop of A-AntMiner is therefore stopped and the next iteration will start. This iteration is repeated until one of two conditions occur. The first condition if the *ConvergenceTest* become 10, then A-AntMiner concludes that the ants have converged to a single rule; or the *AntIndex* is reached to the maximum number which is greater than the user-specified Ant number (i.e., *AN*).

In step twenty four, the A-AntMiner classifier will select the best discovered rule among all discovered rules.

Global Feedback Collection

In step twenty five, the adaptive parameter selection strategy collects feedback while the Ant-Miner selects the best rule among all discovered rules in the iteration by applying the global feedback collection using Equation (4.12).

$$\tau_{n(t_{best})} = (1 - \rho).\ \tau_{n(t_{best})} + \rho.Q(t_{best}) \quad (4.12)$$

where ρ is the quality decay parameter and $Q(t_{best})$ is the quality of the best discovered rule. This mechanism, together with the transition rule, aims to select the best ζ value.

In step twenty six, the best rule among all discovered rules will be added to the list of discovered rules, *DiscoveredRuleList* as mentioned earlier. Subsequently, the classifier will start a new loop by initially setting all terms with the same amount of pheromone. An example of the final discovered rules from Breast Cancer (Wisconsin) in the *DiscoveredRuleList* introduced by A-AntMiner is shown in the following Figure 4.7.

IF Single_Epi_Cell Size = '\(-inf-2.5]\' ***AND*** Normal_Nucleoli = '\(-inf-2.5]\' ***THEN*** 'benign'

IF Cell_Size_Uniformity = '\(4.5-inf)\' ***THEN*** 'malignant'

IF Bare_Nuclei = '\(5.5-inf)\' ***THEN*** 'malignant'

IF Bland_Chromatin = '\(-inf-2.5]\' ***THEN*** 'benign'

IF Clump_Thickness = '\(6.5-inf)\' ***THEN*** 'malignant'

IF Cell_Shape_Uniformity = '\(-inf-1.5]\' ***THEN*** 'benign'

IF Marginal_Adhesion = '\(3.5-inf)\' ***THEN*** 'malignant'

Default rule: 'malignant'

Correctly Classified Instances:	667	95.42 %
Incorrectly Classified Instances:	32	4.58 %

Rules Number : 8

Model Size: 8

Figure 4.7. Classification model obtained from A-AntMiner Classifier

4.3 GA–AntMiner Classifier

The overall goal of this section is to describe a new post-pruning procedure for the Ant-Miner classifier using the GA concept. Three algorithmic components have been added (initial population, crossover and mutation) to the proposed technique. Figure 4.8 presents a low-level description of the GA–AntMiner algorithm. The modification aims to minimise the number of terms in the discovered rule as well as maximise classification accuracy. In the GA-AntMiner classifier, all the components

remain the same (have been explained in previous section) except for the post-pruning procedures which include: population initialisation, crossover, mutation, updating of instance list, determination of consequent rule, calculation of rule quality and stopping criteria.

```
    Input: arff dataset
    Output: classification rule
1   TrainingSet= {all Training instances};
2   InitialConstructionRuleList =[];
3   DiscoveredRuleList=[];
4   WHILE(TrainingSet>MaxUncoveredInstances)
5   AntIndex=1; ConvergenceTest=1;
6   InitializePheromone;
7   REPEAT
8   RuleConstruction;
          // Genetic-based post-pruning technique start here
9   PopulationInitialization;
10       While (termination condition not met);
11            Crossover;
12            Mutation;
13            UpdateInstancesList;
14            DetermineRuleConsequent;
15            EvaluateRule;
16            IF Quality (PruneRule ) > Quality (Rule);
17            Rule= PruneRule;
18            End IF
19        END-WHILE
           // Genetic-based post-pruning technique end here
20  UpdatePheromone;
21  IF(CurrentRule=PreviousRule)
22  THEN ConvergenceTest = ConvergenceTest + 1;
23  ELSE ConvergenceTest = 1;
24  END IF AntIndex = AntIndex + 1;
25  UNTIL(AntIndex>=AntNo) OR
    ConvergenceTest>=RuleConvergenceNo)
26  SelectBestRule;
27  AddBestRule to DiscoveredRuleList;
28  TrainingSet = TrainingSet-{InstancesSetCoveredByRule};
29  END-WHILE
```

Figure 4.8. GA–AntMiner classifier pseudocode

The parameters and values used in evaluating the GA-AntMiner classifier are listed in Table 4.3. The parameters and values are commonly used in many studies are published in Robu et al. (2016) and Raymer et al. (2000).

Table 4.3

GA-AntMiner Experimental Parameters

Parameter	Description	Value
NA	Ant number	10
MCR	Minimum number of instances must cover by the rule	5
MaxUncoveredInstances	Maximum of uncovered instances	10
ConvergenceTest	Convergence test	10
NI	Iterations number	10
CR	Crossover Rate	0.8
MR	Mutation Rate	0.1

The Ant-Miner generates the classification rule as an integer, one-dimensional array that has a size equal to the number of features in the dataset and consists of two parts. The first (antecedent) is where each bit is associated with a dataset feature. If the bit of this array equals a positive integer number, then one term of that feature is allowed to participate in the classification rule. Otherwise, if the bit of this array equals a negative value, the terms of that feature will not be included. Meanwhile, the second part represents the classification label.

In the original post-pruning procedure, the discovered rule will undergo pruning procedure, which aims to avoid the overfitting problem by reducing the size of the discovered rules to increase comprehensibility. The pruning procedure prunes one

term at a time to improve the rule quality. In this case, any removed term cannot be added back again. This will deprive the opportunity of obtaining a good pruned rule by adding/removing the terms during the pruning process. Figure 4.9 shows an example of the original post-pruning procedure on Breast Cancer (Wisconsin) dataset. This dataset consists of nine features (Clump_Thickness, Cell_Size_Uniformity, Cell_Shape_Uniformity, Marginal_Adhesion, Single_Epi_Cell_Size, Bare_Nuclei, Bland_Chromatin, Normal_Nucleoli, and Mitoses). Each feature has number of values called terms, all terms compensate with integers numbers (e.g., Clump_Thickness = 0, 1, or 2 rather than '\'(-inf-4.5]\', '\'(4.5-6.5]\', or '\'(6.5-inf)\').

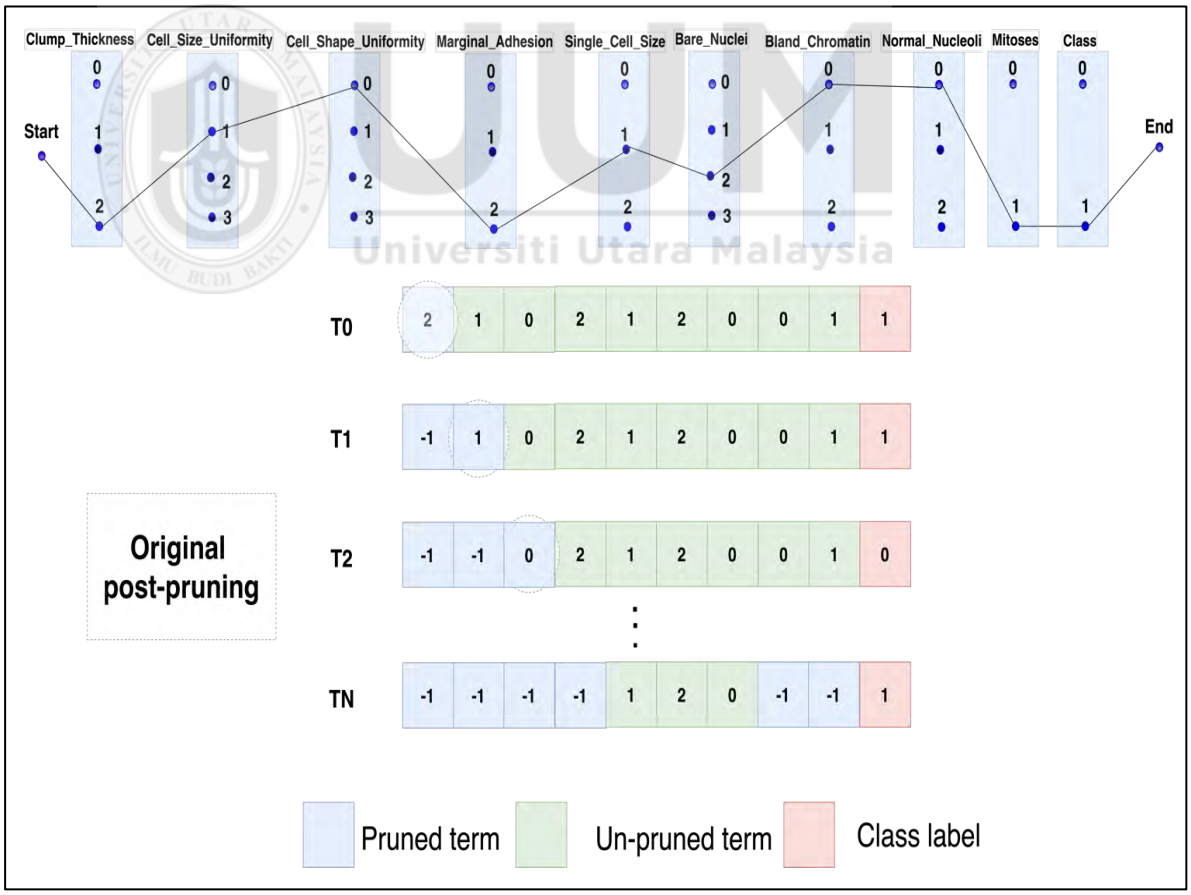

Figure 4.9. An example of original post-pruning technique

Initial population

Step number nine to step number nineteen represent the proposed genetic-based post-pruning technique. Step nine in this research display initial population procedure, which is to create an additional one-dimensional array (rule) of negative values in all of its elements with the same size of the original rule as described in Figure 4.10 (e.g., Breast Cancer (Wisconsin) dataset consist of ten features. Thus, the additional array size=10). The population size is constant and equal to two arrays, and pruning procedure trims the irrelative terms in these two arrays to improve their performance. The second array is created to perform the crossover and mutation operations. In this way, the initial population of genetic-based pruning technique is initialized. Two arrays (i.e., two parents chromosomes) are combined together to form offspring chromosomes.

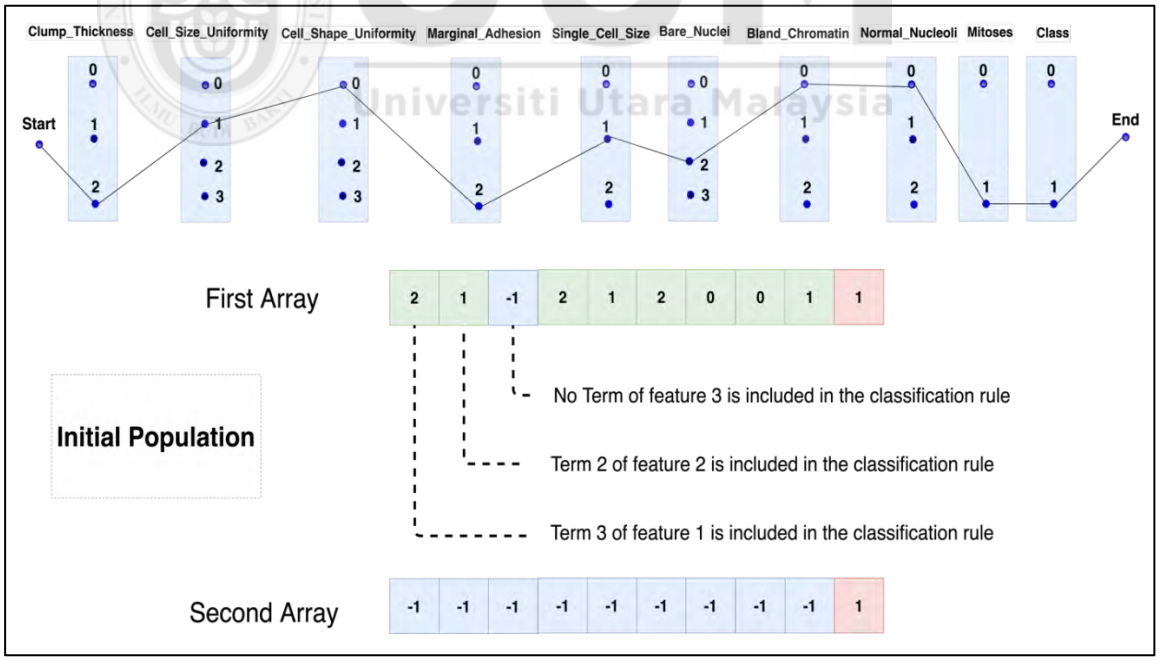

Figure 4.10. Chromosomes of genetic-based post-pruning technique

Step ten of the pruning process is repeated until termination condition is met (number of iterations equal to the feature numbers in the dataset). In this loop the term elimination processes will occur implicitly through crossover and mutation between the two chromosomes while each eliminated term can be re-added. The crossover operator is the process in which parent chromosomes (rules) exchange genetic information to create the best pruning rule.

Crossover

Step eleven is the crossover operation where Figure 4.11 shows the crossover pseudocode. The crossover rate parameter is performed to decide whether the rules should have a crossover or not. In this study, the crossover rate parameter is set to 0.8. The parameter of crossover rate is compared with a random number distributed in between [0,...,1] to perform the crossover operation. In addition, different methods are used to trade genetic information between two individuals. The crossover operation used in this study is two single-point crossover operations. The first point is the first term in the rule, whereas the second point corresponds to the high correlation term that will improve the pruning rule quality.

```
1  FOR each term in the rule
2          IF CrossoverRate > Random();
3                  FirstTerm = SelectFirstTerm();
4                  SecondTerm= SelectSecondTerm();
5                  Offspring = Crossover (FirstTerm, SecondTerm, FirstParent, SecondParent);
6          ELSE: Offspring= (FirstParent, SecondParent);
7  END IF
8  END Loop
```

Figure 4.11. Crossover operator pseudocode

Mutation

Step twelve is the mutation operator used to maintain genetic diversity from one generation of a rule pruning to the next. Figure 4.12 shows the mutation pseudocode. The mutation rate parameter of 0.1 is used to perform mutation in a similar way as the crossover operator. If the mutation rate is greater than the random number distributed in between [0,...,1], then each gene has an equal chance of being mutated during the mutation stage.

The mutation operator selects a random bit in the parent chromosomes, then flipping the value of this bit. An example of two single point crossovers and one single point mutation operator used in the post-pruning technique on Breast Cancer dataset is shown in Figure 4.13.


```
1        IF MutationRate > Random();
2            MutatedTerm = SelectMutatedTerm();
3            Offspring = Mutation (MutatedTerm , FirstParent, SecondParent);
4        ELSE: Offspring= (FirstParent, SecondParent);
5   END IF
```

Figure 4.12. Mutation operator pseudocode

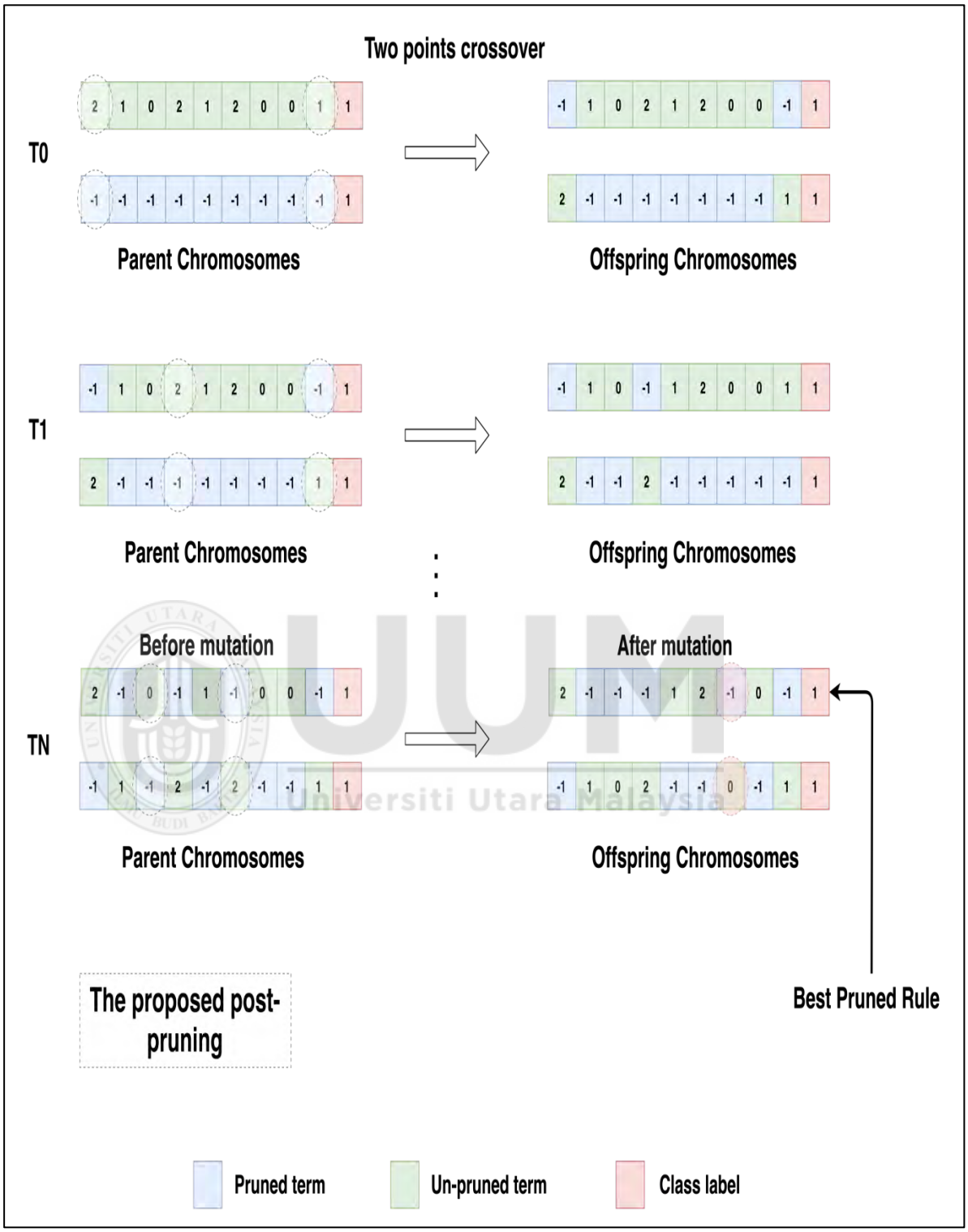

Figure 4.13. Crossover operation with two single points and mutation operation with one single point

In steps fourteen and fifteen, the number of instances covered by the pruning rule is checked using the update instance list procedure. If the number of instances changes, the classifier selects the consequence ('then' part) of the rule by assigning the majority class appearing in the instances covered by the rule.

In steps sixteen and seventeen, the quality of the pruning rule is compared with the original rule using Equation (4.11). If the pruning rule has a higher quality than the original rule, then it replaces the original rule. In step number nineteen, these procedures are repeated until the termination condition is met, which is a fixed amount of elimination using crossover and mutation operators.

4.4 Hybridising with Iterated Local Search

The overall goal of hybridisation of the Ant-Miner with ILS (ILS-AntMiner) is to benefit from the good characteristics of both algorithms to form the neighbourhood structures. In ILS, there is a strong inter-correlation between exploration components (e.g., perturbation) and exploitation components (e.g., local search). Furthermore, ILS has succeeded in employing the perturbation in exploiting multiple neighbourhood structures. Figure 4.14 shows how ILS utilises perturbation to improve the search in two of its neighbourhood structures in the proposed hybridisation. The change that may be applied to the classification rule is defined by a neighbourhood structure. A neighbourhood structure is a function $N: S \rightarrow 2^S$ that assigns to every $s \in S$ a set of neighbourhood $N(s) \subseteq S$. $N(s)$ is also called the neighbourhood of s.

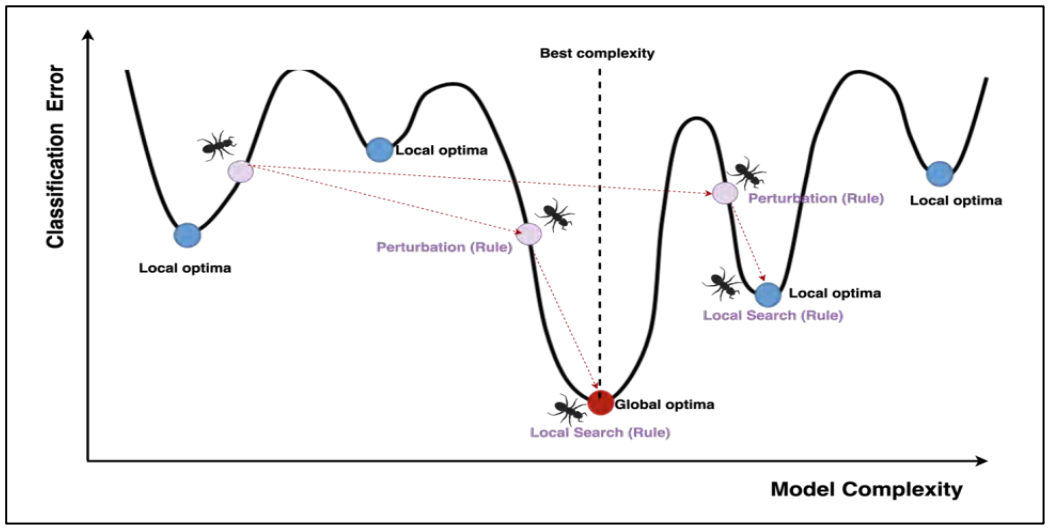

Figure 4.14. Proposed hybrid search strategy

In the ILS-AntMiner classifier, all the steps remain the same (have been explained in previous Section 4.2) except for the ILS (i.e., perturbation, local search and acceptance criterion) will be applied for the best discovered rule in each iteration to accelerate the search. Thus, ILS is only applied for the iteration-best rule instead of applying it for all constructed rules in order to ensure the algorithm remains lightweight. The iteration-best rule is determined according to Equation (4.11). The ILS–Ant-Miner pseudocode (refer Figure 4.15) shows the adaptation of the above mentioned components of the ILS-based algorithm in the AntMiner framework. The combined feature makes the proposed classifier substantially different from the previous Ant-Mining classification algorithm. In addition, the work proposed by Toksari (2016) that hybrid between ACO and ILS algorithms is different from our hybridisation study in three aspects. The first aspect the hybridization was between the AS variant and ILS, while the current research combined the Ant-Miner variant and ILS. The second aspect, the evaluation performance was conducted using different economic indicators, such as population, export, GDP and import, on

Turkey's dataset. ILS-AntMiner used different evaluation performance that explained in Equations (4.13) and (4.14) as discussed later in the Acceptance Criterion section. Finally, the third aspect, Toksari (2016) used modified perturbation mechanism known as ''random sign" that add positive or negative sign for each variable in the forecasting of Turkey's dataset. This work has proposed a new perturbation mechanism to be adaptive with data classification (refer perturbation section and Figure 4.16).

```
    Input: arff dataset
    Output: classification rule
1   TrainingSet= {all Training instances};
2   InitialConstructionRuleList =[];
3   DiscoveredRuleList=[];
4   WHILE(TrainingSet>MaxUncoveredInstances)
5   AntIndex=1; ConvergenceTest=1;
6   InitializePheromone;
7   REPEAT
8   RuleConstructs;
9   RulePrune;
10  UpdatePheromone;
11  IF(CurrentRule=PreviousRule)
12  THEN ConvergenceTest = ConvergenceTest + 1;
13  ELSE ConvergenceTest = 1;
14  END IF AntIndex = AntIndex + 1;
15  UNTIL(AntIndex>=AntNo) OR
    (ConvergenceTest>=RuleConvergenceNo)
                                        //ILS stage starts here
16  SelectBestRule;
17          REPEAT
18             BestRule′ =Perturbation (BestRule);
19             BestRule* =LocalSearch(BestRule');
20             BestRule = AcceptanceCriterion (BestRule, BestRule*);
21           UNTIL termination condition met
                                        //ILS stage ends here
22  AddBestRule to DiscoveredRuleList;
23  TrainingSet = TrainingSet-{InstancesSetCoveredByRule};
24  END-WHILE
```

Figure 4.15. ILS–AntMiner pseudocode

The parameters used for the ILS-AntMiner classifier are adopted from Robu et al. (2016) and Stützle and Ruiz (2017) and are listed in Table 4.4.

Table 4.4
ILS-AntMiner Experimental Parameters

Parameter	Description	Value
NA	Ant number	10
MCR	Minimum number of instances must cover by the rule	5
MaxUncoveredInstances	Maximum of uncovered instances	10
ConvergenceTest	Convergence test	10
NI	Iterations number	10
Perturbation	Perturbation size	4
LocalSearch	Local search size	2

The procedure starts with training data instances to discover one classification rule. The rule will be added to the *discovered-rule-list*. This process will stop when all instances in the dataset are less than pre-specified parameter values, which are known as *MaxUncoveredInstances*. The best rule in the *discovered-rule-list* will be selected according to its quality by using Equation (4.11). Therefore, the best discovered rule in each iteration as shown in step number sixteen will be the kernel of ILS to form the neighbourhood structure.

Perturbation

The ILS-AntMiner is a multiple neighbourhood structure algorithm. Over each iteration in step number seventeen, the perturbation generates a new starting rule where the local search can be applied until termination condition met (i.e., equal to number of attributes to each dataset).

Step eighteen is the perturbation operation, According to Stützle and Ruiz (2017) that use ILS in TSP problem, the perturbation size has been larger than the local search to ensure that the perturbation is enough to escape from local optima. A two-edges-exchange has been used for the local search procedure, and perturbation is implemented by using four-edges-exchange. For this reason, perturbation strength depends on the number of classification rule components that need to be modified. A strong perturbation leads to modify several terms and the structure of the current classification rule may be lost. Thus, the size of perturbation is twice the size of local search movement which is *4-terms exchange.*

The rule solution is represented as an integer, one-dimensional array that has a size equal to the number feature of the dataset. Each bit in the array is associated with a feature, and each one has a different integer number of terms (data dependence). These constraints will not allow the array bits to change easily, as the random movement will not be able to explore other terms that do not appear in the current rule.The constraint by the features and their possible integer number of terms can lead to non-existent solutions. Figure 4.16 shows how the original perturbation that is used in TSP problem, and swap between the values of bit 1 and bit 3 will lead to non-existent solutions in the classification rule (note that the value of bit 3 (Cell_Shape_ Uniformity attribute) = 3 is not identified in bit 1 (Clump_Thickness attribute)). Thus, the perturbation procedure has been modified to select four features from the rule. The perturbing moves were subsequently chosen within the neighbourhood structure of the selected attributes for the current classification rule. For example, the selected Cell_Shape_ Uniformity attribute = 3 is replaced within the available

Cell_Shape_Uniformity attribute values (e.g., 0,1,2,3) only. The differences between ILS that used in TSP problem and rule classification are clarified with Breast Cancer (Wisconsin) dataset example in Figure 4.16.

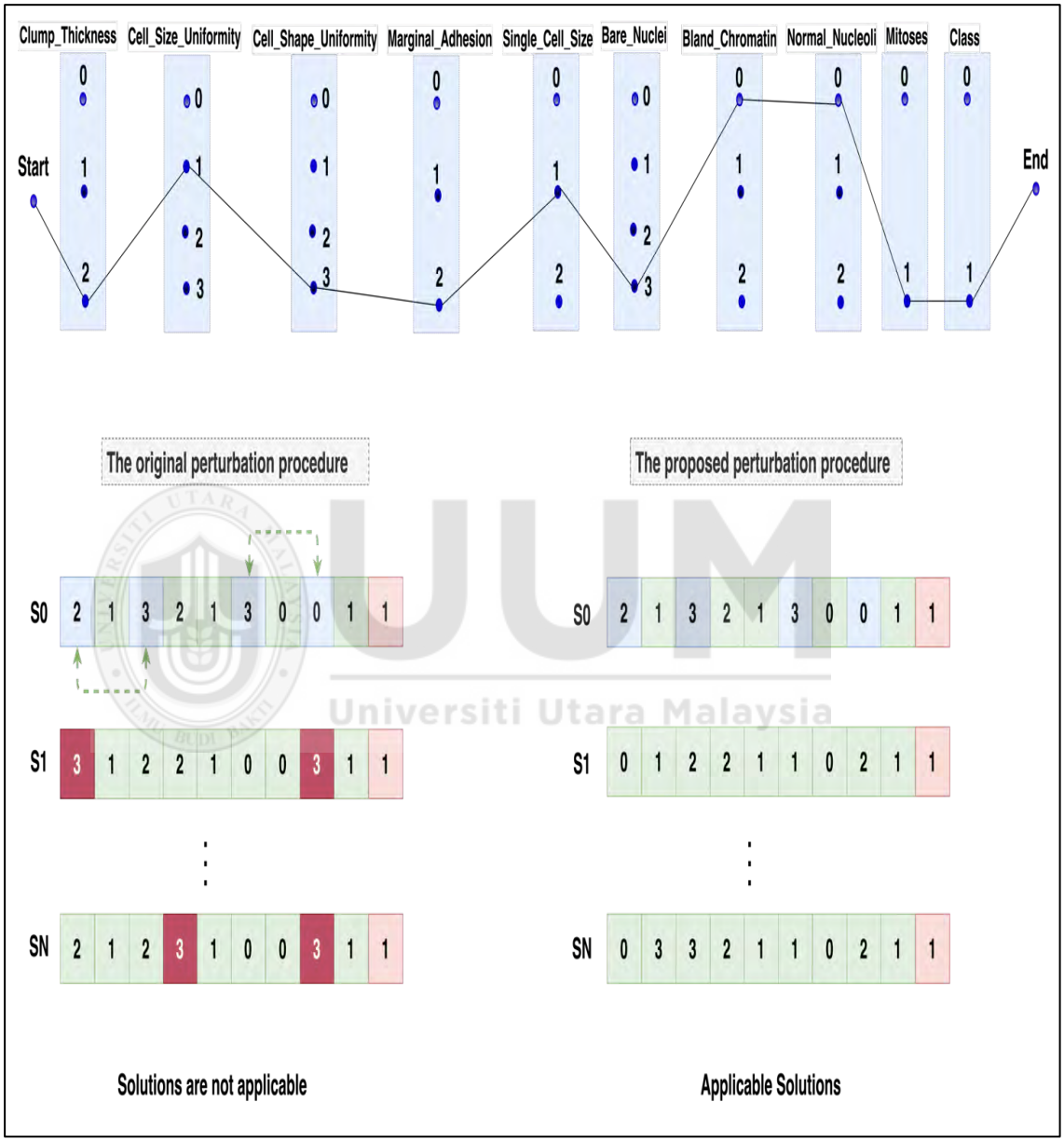

Figure 4.16. The 4-terms exchange perturbation

Local Search

Step number nineteen represent the local search operation, in ordered to exploit the new promising search regions, the size of local search movement in ILS-AntMiner is set to *2-terms exchange* which also use in TSP problem (Stützle & Ruiz, 2017). The *2-terms exchange* replaces up to two terms from the current rule and then adds other possible terms in the same manner by using perturbation procedure (see Figure 4. 16). The procedure then repeatedly performs operations from the given rule until no further improvement can be achieved.

Acceptance Criterion

In steps twenty, the acceptance criterion is used to accept the better-quality classification rule during the local search stages by using our developed Equations (4.13) and (4.14).

$$\text{AcceptanceCriterion} = \begin{cases} \text{BestRule}^*, \ if \ \text{Quality (BestRule}^*) > \text{Quality (BestRule)} \\ \text{BestRule}, \ Otherwise \end{cases} \quad (4.13)$$

$$\text{Quality (BestRule}^*) = \frac{TP}{TP+FN+FP} + \frac{TN}{FP+TN} \quad (4.14)$$

where *TP* is the number of instances covered by the discovered rule and class predicted by the rule. *FN* is the total number of instances covered by the discovered rule and class different from the class predicted by the rule. *TN* is the total number of instances not covered by the discovered rule and does not have the class predicted by

the rule. *FP* is the total number of instances not covered by the discovered rule but has the class predicted by the rule.

The acceptance criteria in ILS will guide the search toward the bottom of the neighbourhood structure shape. Furthermore, it determines the size of movement in the current search region.

4.5 AGI-AntMiner classifier

This section explains the AGI-AntMiner for rules-based classification based on the integration of the three modifications as explained in Sections 4.2 to 4.4. Figure 4.17 shows the integration of these modifications in the AGI-AntMiner algorithm for classification pseudocode. The main objective of this integration is to find the classification model that balances between rule accuracy and comprehensibility.

The first modification proposes the ζ parameter as an importance rate to select the term. Then, the ACO algorithm is used as an adaptive strategy to determine, dynamically, the ζ parameter value with respect to the dataset, learning process stages and quality of the obtained solution. This control strategy uses machine learning and ACO principles and consists of four procedures (initialise feedback collection memory, ζ parameter selection, local and global feedback collection). Section 4.3 explains these procedures in detail.

```
    Input: arff dataset
    Output: classification rule
1   TrainingSet= {all Training instances};
2   InitialConstructionRuleList =[];
3   DiscoveredRuleList=[];
4   WHILE(TrainingSet>MaxUncoveredInstances)
5   AntIndex=1; ConvergenceTest=1;
6   InitializePheromone;
                              //A-AntMiner procedures starts here
7   InitializeFeedbackCollection;
8   ParameterSelection; /*   select ζ t value determine the importance or the strength of each
    term */
9   REPEAT
10              WHILE (TermsNotSelected==True)
11                      IF (TheProbabilityOfChosenTermij is >= ζt)
12                      THEN AddTerm;
13                      ELSE RejectTermFromInclusion;
14                      END IF;
15              END WHILE
16  LocalFeedbackCollection;
                              //GA-AntMiner procedures starts here
17  PopulationInitialization;
18              WHILE (termination condition not met);
19                      Crossover;
20                      Mutation;
21                      UpdateInstancesList;
22                      DetermineRuleConsequent;
23                      EvaluateRule;
24                      IF Quality ( PruneRule ) > Quality (Rule);
25                      Rule= PruneRule;
26                      End IF
27              END-WHILE
                              //GA-AntMiner procedures ends here
28  UpdatePheromone;
29  IF(CurrentRule=PreviousRule)
30  THEN ConvergenceTest = ConvergenceTest + 1;
31  ELSE ConvergenceTest = 1;
32  END IF AntIndex = AntIndex + 1;
33  UNTIL(AntIndex>=AntNo) OR (ConvergenceTest>=RuleConvergenceNo)
34  SelectBestRule;
                              //ILS-AntMiner procedures starts here
35              REPEAT
36                  BestRule′ =Perturbation (BestRule);
37                  BestRule* =LocalSearch(BestRule');
38                  BestRule = AcceptanceCriterion (BestRule, BestRule*);
39              UNTIL termination condition met
                              //ILS-AntMiner procedures ends here
40  GlobalFeedbackCollection;
                              //A-AntMiner procedures ends here
41  AddBestRule to DiscoveredRuleList;
42  TrainingSet = TrainingSet-{InstancesSetCoveredByRule};
43  END-WHILE
```

Figure 4.17. AGI-AntMiner algorithm for classification pseudocode

Thereafter, the second modification of the post-pruning procedure aims to avoid the overfitting problem by reducing the size of the discovered rules to increase comprehensibility. This modification is based on a new post-pruning technique that incorporates the concept of the GA. The operational steps of the proposed technique are population initialisation, crossover, mutation, update of instance list, determination of the consequent rule, calculation of rule quality and stopping criteria, which were discussed in the GA–AntMiner section.

The last modification is done by applying the local search stage. The best rule among all construction rules will undergo an adapted ILS procedure. The ILS algorithm uses an iterative behaviour by successively modifying the current rule with a better rule in its neighbourhood. The ILS has four algorithmic procedures (initial solution, perturbation, local search and acceptance criterion).

Section 4.5 explains the adaptation of the related procedure. The main objective for the local search is to escape from local optima and to enhance classification accuracy. Therefore, the proposed AGI-AntMiner algorithm for rules-based classification aims to generate the balance between classification accuracy and simplicity through the combination of these modifications. Additionally,

4.6 Summary

One of the challenging issues in the ACO-based classification algorithm is to increase classification accuracy and simplicity of discovery rules. For this purpose, the characteristics of the proposed pre- and post-pruning and local search are

combined to produce the proposed AGI-AntMiner-based rule classification. Therefore, the proposed classification consists of three major modifications on the original Ant-Miner algorithm. Three classification algorithms based on the modifications were implemented as a stand-alone classifier with the equations, figures and pseudocode. The first classification algorithm, A–AntMiner, introduced a pre-pruning strategy based on an online adaptive threshold value to determine the strength and importance of each term to be included in the rule. The second classifier, GA–AntMiner, developed an enhancement for the post-pruning technique by using the concept and principles of GA to remove irrelative terms and overcome the limitation of the existing post-pruning. Lastly, the third classifier, ILS–AntMiner, adapted the ILS to enable the classifier to escape from local optima and to improve classification accuracy. The next chapter presents the experimental result of the proposed classifiers.

CHAPTER FIVE
EXPERIMENTAL RESULTS

5.1 Introduction

This chapter presents an empirical evaluation of the proposed ACO-based rules classification algorithm. The proposed classifiers are tested on 16 different datasets from UCI and compared with the most-related baseline ant-mining classification algorithms. Section 5.2 presents the experimental methods used to develop the experiments. Section 5.3 shows the results and analysis of the A-AntMiner classifier. Section 5.4 presents the experiment results of the proposed GA-AntMiner classifier. Section 5.5 discusses the results of the hybridising Ant-Miner with ILS. Section 5.6 presents the last evaluation stage, wherein the experimental results of the AGI-AntMiner for classification are compared with related ant-mining and state-of-the-art rules-based classification algorithms. Finally, Section 5.7 summarizes the chapter.

5.2 Experimental Design

The experimental results of this research use a set of 16 real-world problems from the UCI datasets repository. These benchmark datasets are widely used in the literature of Ant-mining classification algorithms. The discretisation has been carried out as a pre-processing step for transferring continuous attributes into discrete attributes, the well-known C4.5 algorithm for discretizing continuous attributes has been used. In addition, the evaluation carried out with previous Ant-mining classification algorithms are considered the most related to our study as well as the

state of art rule-based classification algorithms discussed in Chapter 2 (Section 2.3 & Section 2.4) and Chapter 3 Section (3.2.4).

The well-known 10-fold cross-validation method is used in our empirical evaluation stage. The dataset in this method is divided into 10 subsets. All subsets have the same size. Nine subsets are used for the learning process, whereas the remaining subset is utilised for the testing stage. This process is repeated 10 times in the same way but with a different subset for learning and testing to guarantee that all subsets are used in both stages. Thereafter, the performances of the 10 testing subsets are averaged, and the standard deviations are computed. The 10-fold cross-validation method is commonly used in studies involving ant-mining algorithms. The evaluation metrics have been analysed using conventional benchmark scenarios from the ant-mining literature. These benchmarks include the classification accuracy, number of discovered rules and number of terms in the discovered rules.

In order to evaluate the proposed classification algorithms three comparing methods are used. The first step of evaluation method involves an individual comparison with other classifiers. In this case, the proposed classifiers undertake a comparison with other classifiers, individually, for each evaluation metric. In the second method of evaluation, the overall performances for all classifiers are conducted to all classification metrics. Finally, the results of the nonparametric Friedman test with Holm's post-hoc test are used to determine the average classification accuracy rank versus the average number of discovered rule rank and the average model size rank for all classifiers. This test aims to find the best classifier that balances between different objectives (e.g., classification accuracy and model size). The experimental

methods used to develop the experiments in this study are presented in Figure 5.1 below.

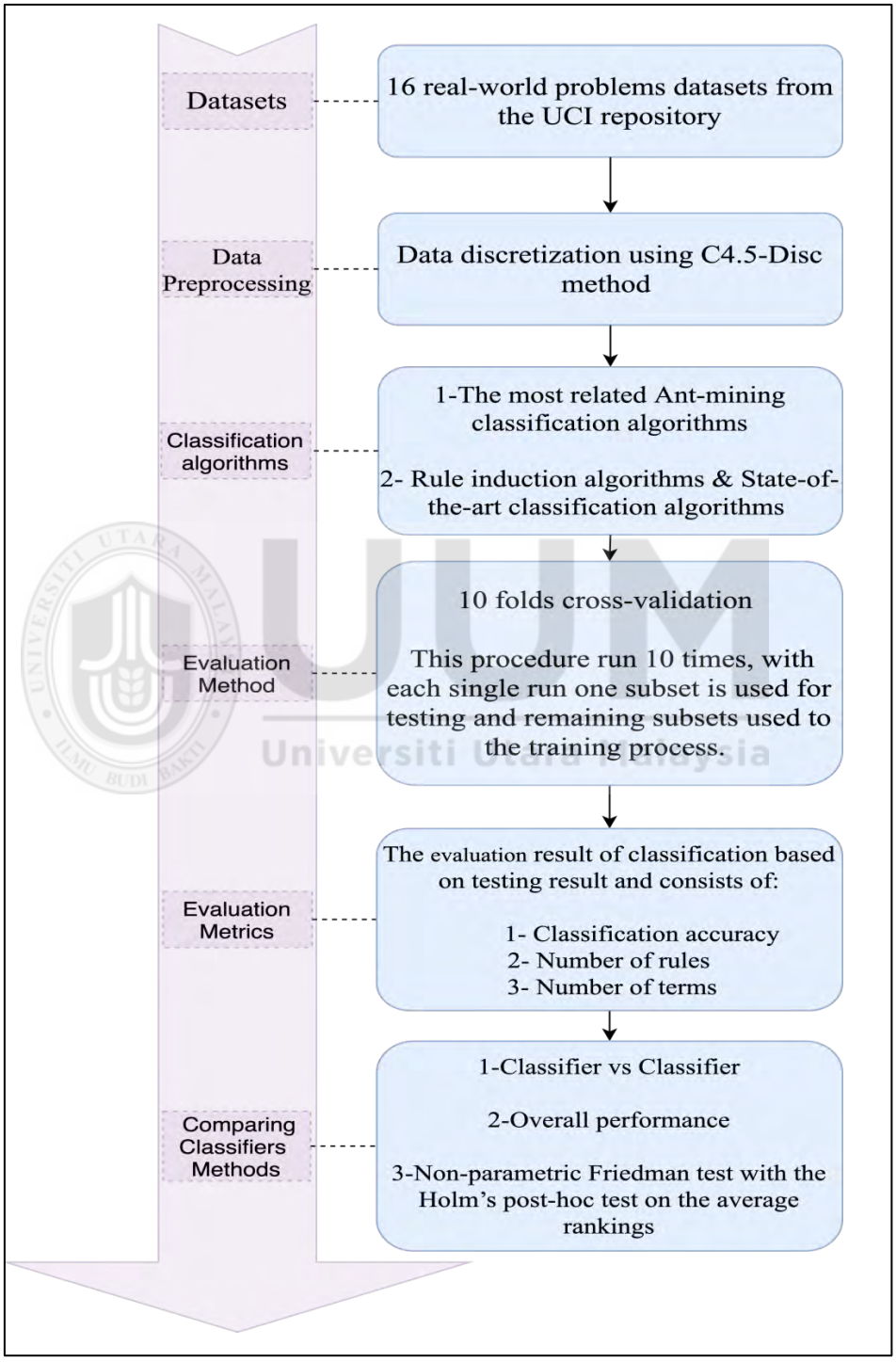

Figure 5.1. Experimental Design

5.3 Results and Analysis of the A-AntMiner Classifier

The results of the implemented A-AntMiner classifier are compared with those of five different classification algorithms, namely original Ant-Miner (Parpinelli et al., 2002a), CAnt-Miner (Otero et al., 2008), ACO/PSO2 (Holden & Freitas, 2008), TACO-Miner (Thangavel & Jaganathan, 2007; Tripathy et al., 2013), and Ant-Miner with a Hybrid Pruner called, Hybrid Pruner in our research (Chan & Freitas, 2006a). These classification algorithms are considered to be the most-related classifiers with different rule pruning procedures in the literature. The classification performance of the six algorithms is analysed in detail by using 16 datasets from UCI. Several benchmark scenarios are used for analysing the A-AntMiner performance. In the first scenario, Tables 5.1, 5.2, and 5.3 show the experimental results of the average classification accuracy, average number of discovered rules, and model size, respectively, by using the 10-fold cross-validation method. In each table, the first row presents the average classification accuracy, and the numbers after the symbol "+/–" are standard deviations. The best result for each dataset is written in bold. The second row displays the classifier performance rank for each dataset. The experimental results in Tables 5.1, 5.2, and 5.3 are used to determine the best classifiers.

Table 5.1 shows that the A-AntMiner outperforms the Ant-Miner classifier in all datasets in terms of classification accuracy, which is the relation between the numbers of instances classified correctly with the dataset size. The A-AntMiner also achieves 13 and 12 out of 16 datasets compared with CAnt-Miner and ACO/PSO2,

respectively. Furthermore, the A-AntMiner is better than TACO-Miner and Hybrid Pruner in 15 datasets.

The experimental results show that the A-AntMiner obtains the best results in 10 datasets compared with the other classifiers. Furthermore, the A-AntMiner obtains the second best results in four datasets, namely Credit-g, Diabetes, Segment, and Tic-tac-toe. However, the second best classifier is ACO/PSO2, which obtains the best results in three datasets. The CAnt-Miner, and TACO-Miner obtain the best results in two, and one datasets respectively. Table 5.1 displays the average, standard deviation, and performance rank for all datasets in classification accuracy.

Table 5.1

Average classification accuracy (average +/− standard deviation, performance rank) obtained using 10-fold cross-validation method for all classifiers and A-AntMiner

Dataset		**Ant-Miner**	**CAnt-Miner**	**ACO/ PSO2**	**TACO-Miner**	**Hybrid Pruner**	**A-AntMiner**
Balance Scale	Accuracy	69.73% +/- 1.58%	69.29 % +/- 1.112	68.66 % +/- 4.97	66.65% +/- 2.1%	68.62% +/- 1.21%	**72.13% +/- 2.7%**
	Rank	2	3	4	6	5	**1**
Breast Cancer Ljubljana	Accuracy	72.32% +/- 1.73%	74.87% +/- 1.846	70.94 % +/- 5.37	74.66% +/- 2.52%	72.67% +/- 2.52%	**75.23% +/- 2.59%**
	Rank	5	2	6	3	4	**1**
Breast Cancer Wisconsin	Accuracy	94.43% +/- 1.17%	94.42% +/- 0.889	93.86 % +/- 4.56	94.56% +/- 0.85%	94% +/- 1.06%	**95.42% +/- 0.67%**
	Rank	3	4	6	2	5	**1**
Credit-a	Accuracy		84.92%	84.69	78.99%	84.64%	**85.22%**

		84.49% +/- 1.04%	+/- 1.063	% +/- 4.39	+/- 2.59%	+/- 1.06%	**+/- 1.31%**
	Rank	5	2	3	6	4	**1**
Credit-g	Accuracy	70.7% +/- 1%	**71.80% +/- 0.841**	71.0 % +/- 4.52	69.4% +/- 2.16%	70.4% +/- 0.81%	71.3% +/- 1.83%
	Rank	4	**1**	3	6	5	2
Diabetes	Accuracy	71.12% +/- 2.01%	74.61% +/- 2.197	**76.31 % +/- 4.32**	71.99% +/- 1.49%	73.3% +/- 1.68%	75.39% +/- 1.84%
	Rank	6	3	**1**	5	4	2
Heart Cleveland	Accuracy	76.17% +/- 2.85%	77.23% +/- 1.652	78.51 % +/- 6.16	76.13% +/- 2.32%	76.63% +/- 1.49%	**80.46% +/- 2.67%**
	Rank	5	3	2	6	4	**1**
Heart Statlog	Accuracy	77.78% +/- 2.41%	77.77% +/- 2.869	78.89 % +/- 7.78	77.78% +/- 2.14%	77.78% +/- 2.59%	**79.63% +/- 2.6%**
	Rank	3	6	2	3	3	**1**
Hepatitis	Accuracy	80.03% +/- 3.68%	76.20% +/- 2.034	76.13 % +/- 8.34	78.98% +/- 3.65%	75.71% +/- 2.89%	**80.36% +/- 4.04%**
	Rank	2	4	5	3	6	**1**
Ionosphere	Accuracy	86.03% +/- 1.77%	84.60% +/- 1.074	65.51 % +/- 7.46	79.54% +/- 1.89%	86.51% +/- 1.77%	**88.85% +/- 1.32%**
	Rank	3	4	6	5	2	**1**
Iris	Accuracy	94% +/- 1.85%	94.66% +/- 1.663	94.0 % +/- 8.14	94.67% +/- 1.94%	94.67% +/- 1.66%	**96% +/- 1.78%**
	Rank	5	4	5	2	2	**1**
Lymphography	Accuracy	71.37% +/- 1.87%	74.85% +/- 3.475	77.19 % +/- 12.59	**78.56% +/- 2.89%**	68.26% +/- 2.59%	75.57% +/- 2.67%
	Rank	5	4	2	**1**	6	3
Mushroom	Accuracy	97.14% +/- 0.42%	97.93% +/- 0.561	**100.0 % +/- 0.0**	96.27% +/- 0.75%	97.91% +/- 0.45%	97.78% +/- 0.63%
	Rank	5	2	**1**	6	3	4
Segment	Accuracy	80.04% +/- 1.4%	**84.76% +/- 0.846**	82.08 % +/- 4.64	76.88% +/- 0.85%	82.99% +/- 1.24%	83.98% +/- 0.62%
	Rank	5	**1**	4	6	3	2
Sonar	Accuracy	75.61%	77.88% +/-	54.86 % +/-	72.1% +/-	75.09% +/-	**78.9% +/- 2.73%**

		+/- 2.64%	2.482	3.87	4.01%	3.63%	
Tic-tac-toe	Rank	3	2	6	5	4	**1**
	Accuracy	73.58% +/- 1.72%	72.23% +/- 1.361	**100.0 % +/- 0.0**	71.59% +/- 1.57%	72.33% +/- 1.4%	74.4% +/- 2.61%
	Rank	3	5	**1**	6	4	2

Table 5.2 shows that the A-AntMiner is better than Ant-Miner in eight datasets in terms of discovered rules, which refer to the average number of rules for the 10-fold cross-validation. In addition, the A-AntMiner is better than ACO/PSO2 in 14 datasets. The A-AntMiner earns better results in seven datasets than the Hybrid Pruner classifier. Conversely, the A-AntMiner obtains worse results in 11 and 13 datasets than CAnt-Miner and TACO-Miner classifiers, respectively.

In comparison with the other classifiers, TACO-Miner obtains the best results in nine datasets. The CAnt-Miner obtains the best results in four datasets. The A-AntMiner, ACO/PSO2, and Hybrid Pruner classifiers obtain the best results in two datasets. Furthermore, the Ant-Miner obtains the best result in one dataset.

Table 5.2

Average number of rules (average +/− standard deviation, performance rank) obtained using 10-fold cross-validation method for all classifiers and A-AntMiner

Dataset		**Ant-Miner**	**CAnt-Miner**	**ACO/ PSO2**	**TACO-Miner**	**Hybrid Pruner**	**A-AntMiner**
Balance Scale	Accuracy	8 +/- 0	7.89 +/- 0.33	22 +/- 0	**6.7 +/- 0.26**	8 +/- 0	7 +/- 0
	Rank	4	3	6	**1**	4	2

Breast Cancer Ljubljana	Accuracy	6.1 +/- 0.1	**5.90 +/- 0.32**	11.3 +/- 2.05	6.4 +/- 0.45	6.5 +/- 0.22	6.1 +/- 0.23
	Rank	2	**1**	6	4	5	2
Breast Cancer Wisconsin	Accuracy	7.7 +/- 0.21	7.0 +/- 0.0	9.9 +/- 1.37	**6.8 +/- 0.25**	7.4 +/- 0.22	7.5 +/- 0.22
	Rank	5	2	6	**1**	3	4
Credit-a	Accuracy	7.6 +/- 0.16	6.90 +/- 0.32	20.1 +/- 1.37	7.2 +/- 0.33	7.4 +/- 0.4	**6.7 +/- 0.45**
	Rank	5	2	6	3	4	**1**
Credit-g	Accuracy	9.4 +/- 0.22	8.9 +/- 0.87	12.9 +/- 4.84	**8.5 +/- 0.17**	9.1 +/- 0.31	8.6 +/- 0.34
	Rank	5	3	6	**1**	4	2
Diabetes	Accuracy	9.7 +/- 0.21	9.5 +/- 1.08	37.1 +/- 2.23	**9.3 +/- 0.3**	**9.3 +/- 0.26**	9.5 +/- 0.17
	Rank	5	3	6	**1**	**1**	3
Heart Cleveland	Accuracy	6.3 +/- 0.21	6.3 +/- 0.82	10.6 +/- 1.42	**5.6 +/- 0.16**	6.4 +/- 0.27	5.9 +/- 0.23
	Rank	3	3	6	**1**	5	2
Heart Statlog	Accuracy	6.3 +/- 0.15	5.8 +/- 0.78	9.7 +/- 1.70	**5.7 +/- 0.15**	6 +/- 0.21	6.6 +/- 0.27
	Rank	4	2	6	**1**	3	5
Hepatitis	Accuracy	5.6 +/- 0.22	**5 +/- 0.47**	5.5 +/- 0.84	5.2 +/- 0.2	5.3 +/- 0.3	5.1 +/- 0.18
	Rank	6	**1**	5	3	4	2
Ionosphere	Accuracy	6.2 +/- 0.2	6.2 +/- 0.63	**1.0 +/- 0**	4.8 +/- 0.33	6 +/- 0.21	7 +/- 0.21
	Rank	4	4	**1**	2	3	6
Iris	Accuracy	4.4 +/- 0.27	**4.3 +/- 0.67**	4.7 +/- 0.48	**4.3 +/- 0.21**	**4.3 +/- 0.21**	**4.3 +/- 0.21**
	Rank	5	**1**	6	**1**	**1**	**1**
Lymphography	Accuracy	6 +/- 0.21	6.9 +/- 0.87	15.4 +/- 1.34	**5.1 +/- 0.23**	5.5 +/- 0.27	6.5 +/- 0.22
	Rank	3	5	6	**1**	2	4
Mushroom	Accuracy	8.1 +/- 0.53	**5.7 +/- 0.9**	17.6 +/- 1.42	7.5 +/- 0.34	8.5 +/- 0.22	9.2 +/- 0.51
	Rank	3	**1**	6	2	4	5

Segment	Accuracy	**16.5 +/- 0.5**	18.1 +/- 2.60	33.2 +/- 4.36	18.5 +/- 0.75	18.9 +/- 0.55	19.9 +/- 0.48
	Rank	**1**	2	6	3	4	5
Sonar	Accuracy	5.7 +/- 0.15	5.4 +/- 0.51	**1.4 +/- 0.51**	5.8 +/- 0.2	5.9 +/- 0.18	6.3 +/- 0.15
	Rank	3	2	**1**	4	5	6
Tic-tac-toe	Accuracy	8.4 +/- 0.54	8.1 +/- 2.46	18.9 +/- 2.13	**7.1 +/- 0.41**	9.1 +/- 0.48	8.7 +/- 0.67
	Rank	3	2	6	**1**	5	4

Table 5.3 shows that the A-AntMiner is better than Ant-Miner, CAnt-miner, ACO/PSO2, and Hybrid Pruner in 14, 12, 13, and 13 datasets, respectively, in terms of the average size of the classification model, which refers to the average number of terms for the 10-fold cross-validation. Moreover, the A-AntMiner is better than TACO-Miner in five datasets.

In comparison with other classifiers, TACO-Miner obtains the best results in eight datasets. The A-AntMiner achieves the second best performance in six datasets. ACO/PSO2 obtains the best results in three datasets. The CAnt-Miner achieves the best result in one dataset.

Although, TACO-Miner has been shown as a good classifier to finding good results of the number of discovered rules and number of terms in the discovered rules (see Table 5.2 and 5.3), there are inevitably problems in classification accuracy for the same datasets. These arise because TACO-Miner used a fixed threshold value (0.6) to eliminate the irrelevant terms, while A-AntMiner used a dynamic adjusted

threshold value based on the given dataset. For example, in Balance Scale dataset the TACO-Miner discovered a rule list with 6.7 rules and 6.6 terms per rule with average classification performance of 66. 65%. A-AntMiner discovered rule list on the ordered of very little differences, having 7 rules, and 8 conditions per rules with 72.13% classification accuracy. The outcomes of these experiments show that A-AntMiner can notably increase the classification accuracy with an optimal number of rules and number of terms per rule.

Table 5.3.

Average model size (average +/− standard deviation, performance rank) obtained using 10-fold cross-validation method for all classifiers and A-AntMiner

Dataset		Ant-Miner	CAnt-Miner	ACO/PSO2	TACO-Miner	Hybrid Pruner	A-AntMiner
Balance Scale	Accuracy	11 +/- 0	11 +/- 1	52 +/- 0	**6.6 +/- 0.48**	11 +/- 0	8 +/- 0
	Rank	3	3	6	**1**	3	2
Breast Cancer Ljubljana	Accuracy	7.8 +/- 0.29	7.80 +/- 1.14	26.8 +/- 6.196	7.7 +/- 0.5	8.7 +/- 0.62	**6.8 +/- 0.47**
	Rank	3	3	6	2	5	**1**
Breast Cancer Wisconsin	Accuracy	8.4 +/- 0.22	8.0 +/- 0.94	17.1 +/- 2.42	**7.1 +/- 0.38**	7.4 +/- 0.31	8.1 +/- 0.28
	Rank	5	3	6	**1**	2	4
Credit-a	Accuracy	10.6 +/- 0.4	10.0 +/- 1.83	70.6 +/- 7.6	8.6 +/- 0.5	10.3 +/- 0.58	**8 +/- 0.61**
	Rank	5	3	6	2	4	**1**
Credit-g	Accuracy	14.7 +/- 0.58	16.4 +/- 4.24	30.5 +/- 16.33	13 +/- 0.56	13.2 +/- 0.77	**10.1 +/- 0.85**
	Rank	4	5	6	2	3	**1**
Diabetes	Accuracy	10.5 +/- 0.4	11 +/- 1.94	112.5 +/- 9.312	9.5 +/- 0.4	11.3 +/- 0.75	**8.7 +/- 0.26**
	Rank	3	4	6	2	5	**1**

Heart Cleveland	Accuracy	9.6 +/- 0.69	10 +/- 2.49	28.3 +/- 4.347	**7.7 +/- 0.54**	9.2 +/- 0.8	8.1 +/- 0.5
	Rank	4	5	6	**1**	3	2
Heart Statlog	Accuracy	9.6 +/- 0.58	7.8 +/- 1.39	25.9 +/- 4.30	**5.7 +/- 0.26**	8.7 +/- 0.62	8 +/- 0.39
	Rank	5	2	6	**1**	4	3
Hepatitis	Accuracy	8.1 +/- 0.48	7.9 +/- 1.44	11.6 +/- 2.31	7.8 +/- 0.65	8.1 +/- 0.71	**6.7 +/- 0.45**
	Rank	4	3	6	2	4	**1**
Ionosphere	Accuracy	7.4 +/- 0.64	6.7 +/- 0.94	**2.2 +/- 0.42**	5.9 +/- 0.92	7.1 +/- 0.46	6.9 +/- 0.28
	Rank	6	3	**1**	2	5	4
Iris	Accuracy	3.4 +/- 0.27	3.4 +/- 0.84	**3.3 +/- 0.94**	**3.3 +/- 0.21**	3.4 +/- 0.27	**3.3 +/- 0.21**
	Rank	4	4	**1**	**1**	4	**1**
Lymphography	Accuracy	9.1 +/- 0.5	11.2 +/- 2.39	42.8 +/- 6.48	**5.2 +/- 0.39**	8.8 +/- 0.65	9.2 +/- 0.47
	Rank	3	5	6	**1**	2	4
Mushroom	Accuracy	9.3 +/- 1.25	**4.7 +/- 0.94**	33.4 +/- 2.87	7 +/- 0.26	8.2 +/- 0.47	8.2 +/- 0.51
	Rank	5	**1**	6	2	3	3
Segment	Accuracy	21.9 +/- 0.69	22.5 +/- 4.30	59.3 +/- 7.9	**20.9 +/- 1.39**	24.8 +/- 1.2	22.2 +/- 0.65
	Rank	2	4	6	**1**	5	3
Sonar	Accuracy	10 +/- 0.49	9.7 +/- 1.15	**0.9 +/- 1.97**	7.5 +/- 0.37	10.4 +/- 0.62	8.9 +/- 0.55
	Rank	5	4	**1**	2	6	3
Tic-tac-toe	Accuracy	10.7 +/- 1.69	10.7+/- 4.34	53.6 ± 7.306	**6.8 +/- 0.81**	12.6 +/- 1.31	9.9 +/- 1.39
	Rank	3	3	6	**1**	5	2

An example of the final classification model from the Credit-a dataset introduced by A-AntMiner and other compared classifiers are shown in the following Figure 5.2.

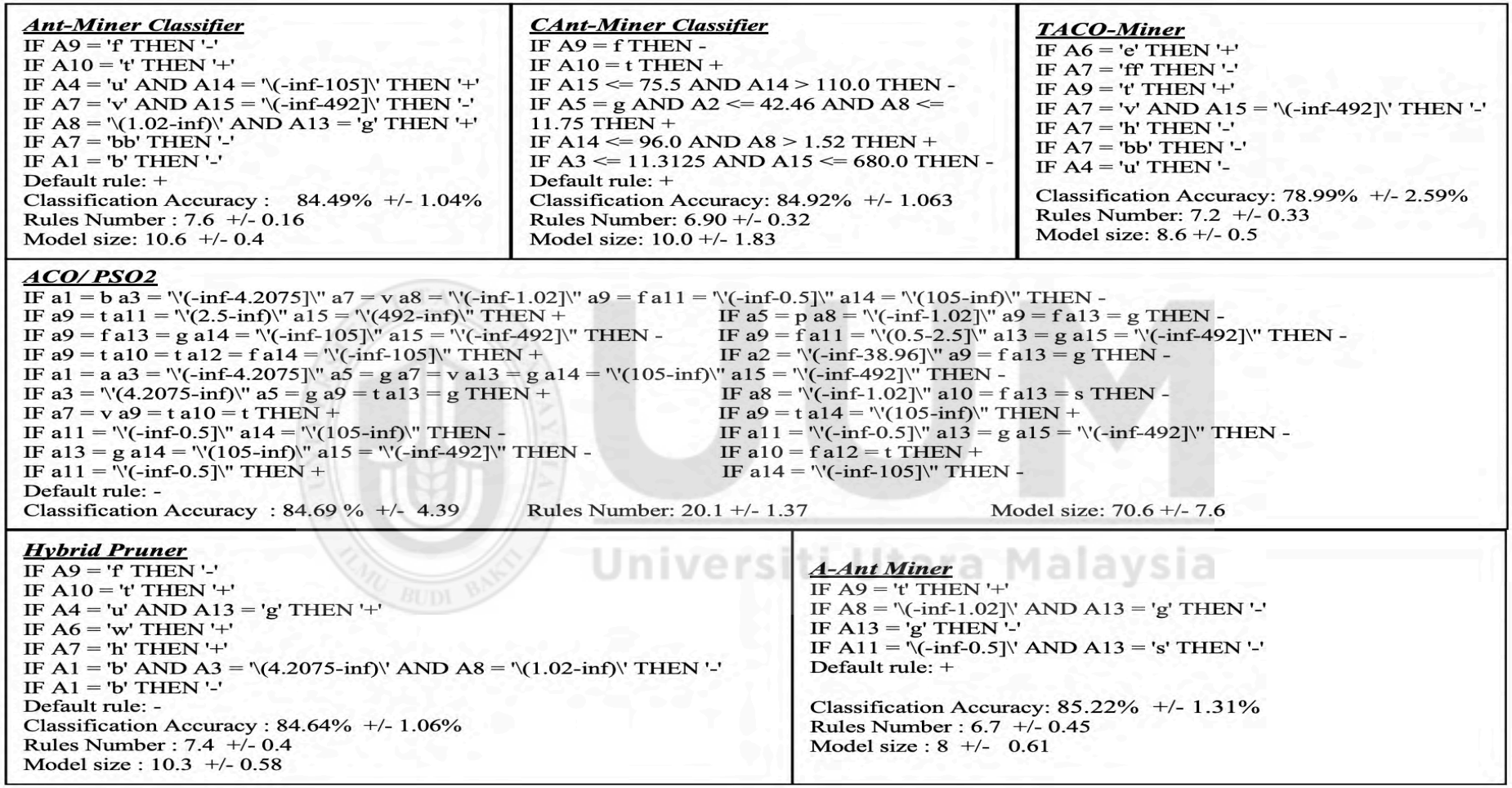

Figure 5.2. An example of the classification model obtained by A-AntMiner and the other classification algorithms

Table 5.4 and Figures 5.3 and 5.4 show the results of the nonparametric Friedman test with Holm's post-hoc test in the second scenario. This scenario determines the average classification accuracy rank, average number of discovered rule rank, and average model size rank of the statistical results, which are reported in Table 5.4, across the 16 datasets. Figure 5.3 displays the results of the average classification accuracy rank versus the average number of rule rank. Figure 5.4 presents the results of the average classification accuracy rank versus the average model size rank. In all cases, the lowest rank indicates a good algorithm performance.

Table 5.4

Test results of A-AntMiner and other classifiers based on average performance rank on all datasets

	Ant-Miner	**CAnt-Miner**	**ACO/PSO 2**	**TACO-Miner**	**Hybrid Pruner**	**A-AntMiner**
Accuracy	4.09375	3.125	3.59375	4.53125	4.09375	**1.5625**
Rule	3.9375	2.5	5.3125	**2**	3.71875	3.53125
Terms	4.21875	3.625	5.125	**1.5625**	4.125	2.34375

Figure 5.3 shows that the results obtained by the A-AntMiner and CAnt-Miner classifiers dominate the other four classifiers. The A-AntMiner only performs slightly worse than CAnt-Miner in the average number of discovered rules and slightly better in the classification accuracy performance. Therefore, A-AntMiner outperforms the other classifiers except CAnt-Miner. Figure 5.4 proves that the results obtained by the A-AntMiner dominate other classifiers when considering the classification accuracy and model size ranks. The A-AntMiner only performs slightly worse than TACO-Miner in terms of model size, but it is better than TACO-Miner

and the other classifiers in terms of classification accuracy. Therefore, the A-AntMiner is the best classifier that balances between the classification accuracy and model size. This balance aims to overcome the problems of overfitting and under-fitting, considering that a less accurate rule in the training process that covers numerous training instances is better than an accurate rule that covers only one instance. This case helps to avoid the overfitting problem and increases the generalisation of the classification model. Our classifier aims to find the appropriate pruning criteria for each dataset by considering the feedback (local and global) in the learning process on the basis of the classification accuracy and avoiding the learning process from constructing a classification model that is too simple to describe a given set of data. Understanding these two phenomena allows the development of a classification algorithm that balances two extremes. Therefore, the A-AntMiner outperforms other classifiers when the classification accuracy and comprehensibility (i.e., number of discovered rules, and number of terms per rule) are balanced.

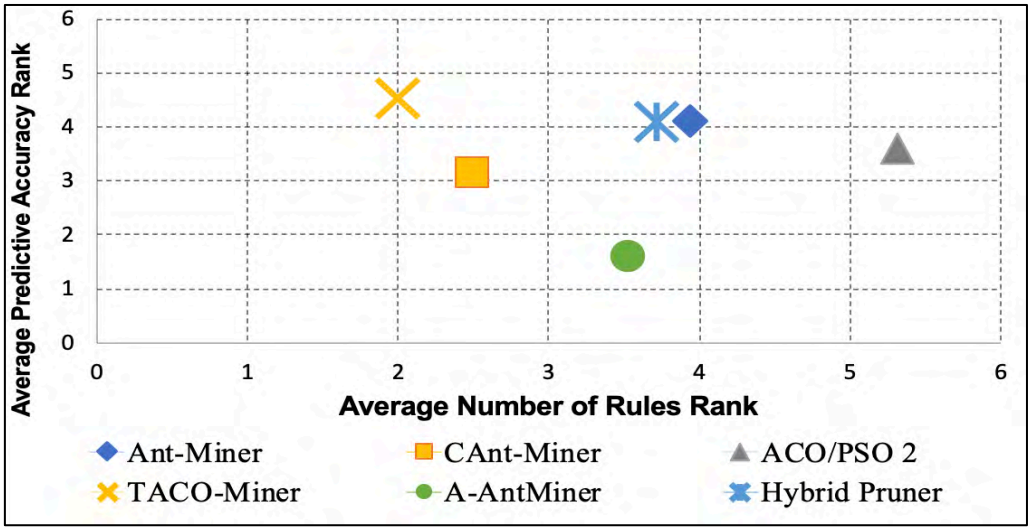

Figure 5.3. Results of A-AntMiner on the average classification accuracy rank versus the average number of discovered rule rank

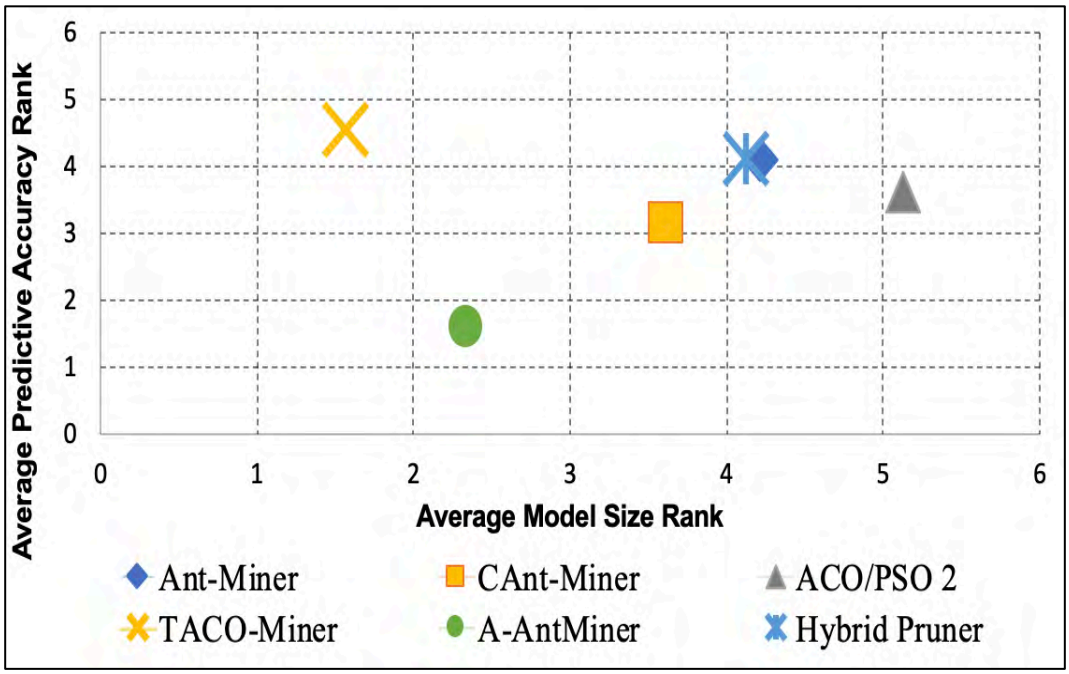

Figure 5.4. Results of A-AntMiner on the average classification accuracy rank versus the average model size rank

5.4 Results and Analysis of the GA-AntMiner Classifier

This section compares the results of the GA-AntMiner classifier with those of related classifiers with different rule pruning procedures. These classifiers include the original Ant-Miner (Parpinelli et al., 2002a), CAnt-Miner (Otero et al., 2008), ACO/PSO2 (Holden & Freitas, 2008), TACO-Miner (Thangavel & Jaganathan, 2007; Tripathy et al., 2013), and Ant-Miner with a Hybrid Pruner called, Hybrid Pruner in our research (Chan & Freitas, 2006a). Experiments on 16 datasets from the UCI repository are conducted for all classification algorithms. The experiments use the 10-fold cross-validation technique based on the benchmark scenarios explained in Chapter 3. In the first method, Tables 5.5, 5.6, and 5.7 show the experimental

results of the average classification accuracy, average number of discovered rules, and model size. In each table, the first row presents the average classification accuracy, and the numbers after the symbol "+/−" are standard deviations. For each dataset, the best result is written in bold. The second row displays the performance rank for each dataset. The experimental results in Tables 5.5, 5.6, and 5.7 are used to determine the best classifiers.

Table 5.5 shows that the GA-AntMiner is better than Ant-Miner in all datasets. The GA-AntMiner is better than TACO-Miner and Hybrid Pruner in 15 datasets. Furthermore, the GA-AntMiner outperforms CAnt-Miner and ACO/PSO2 in 13 and 12 datasets, respectively. In comparison with the other classifiers, the GA-AntMiner achieves the highest results in 10 datasets. The GA-AntMiner obtains the second best performance in four datasets (Credit-g, Diabetes, Segment, and Tic-tac-toe). Meanwhile, the second best classifier is ACO/PSO2 with three datasets. The CAnt-Miner achieves the best results in two datasets and the TACO-Miner classifier obtains the best result in one dataset. Ant-Miner and Hybrid Pruner acquire the lowest results in all datasets.

Table 5.5

Average classification accuracy (average +/− standard deviation, performance rank) obtained using 10-fold cross-validation methods for all classifiers and GA-AntMiner

Dataset		Ant-Miner	CAnt-Miner	ACO/ PSO2	TACO-Miner	Hybrid Pruner	GA-AntMiner
Balance Scale	Accuracy	69.73% +/- 1.58%	69.29 % +/- 1.112	68.66 % +/- 4.97	66.65% +/- 2.1%	68.62% +/- 1.21%	**71.53% +/- 1.46%**

Breast Cancer (Ljubljana)	Rank	2	3	4	6	5	**1**
	Accuracy	72.32% +/- 1.73%	74.87% +/- 1.846	70.94 % +/- 5.37	74.66% +/- 2.52%	72.67% +/- 2.52%	**75.53% +/- 2.59%**
Breast Cancer (Wisconsin)	Rank	5	2	6	3	4	**1**
	Accuracy	94.43% +/- 1.17%	94.42% +/- 0.889	93.86 % +/- 4.56	94.56% +/- 0.85%	94% +/- 1.06%	**94.71% +/- 1.4%**
Credit-a	Rank	3	4	6	2	5	**1**
	Accuracy	84.49% +/- 1.04%	84.92% +/- 1.063	84.69 % +/- 4.39	78.99% +/- 2.59%	84.64% +/- 1.06%	**85.8% +/- 0.68%**
Credit-g	Rank	5	2	3	6	4	**1**
	Accuracy	70.7% +/- 1%	**71.80% +/- 0.841**	71.0 % +/- 4.52	69.4% +/- 2.16%	70.4% +/- 0.81%	71.5% +/- 1.51%
Diabetes	Rank	4	**1**	3	6	5	2
	Accuracy	71.12% +/- 2.01%	74.61% +/- 2.197	**76.31 % +/- 4.32**	71.99% +/- 1.49%	73.3% +/- 1.68%	75% +/- 1.12%
Heart (Cleveland)	Rank	6	3	**1**	5	4	2
	Accuracy	76.17% +/- 2.85%	77.23% +/- 1.652	78.51 % +/- 6.16	76.13% +/- 2.32%	76.63% +/- 1.49%	**79.27% +/- 1.81%**
Heart (Statlog)	Rank	5	3	2	6	4	**1**
	Accuracy	77.78% +/- 2.41%	77.77% +/- 2.869	78.89 % +/- 7.78	77.78% +/- 2.14%	77.78% +/- 2.59%	**80% +/- 1.67%**
Hepatitis	Rank	3	6	2	3	3	**1**
	Accuracy	80.03% +/- 3.68%	76.20% +/- 2.034	76.13 % +/- 8.34	78.98% +/- 3.65%	75.71% +/- 2.89%	**81.93% +/- 2.71%**
Ionosphere	Rank	2	4	5	3	6	**1**
	Accuracy	86.03% +/- 1.77%	84.60% +/- 1.074	65.51 % +/- 7.46	79.54% +/- 1.89%	86.51% +/- 1.77%	**87.22% +/- 1.35%**
Iris	Rank	3	4	6	5	2	**1**
	Accuracy	94% +/- 1.85%	94.66% +/- 1.663	94.0 % +/- 8.14	94.67% +/- 1.94%	94.67% +/- 1.66%	**96% +/- 1.47%**

	Rank	5	4	5	2	2	**1**
Lymphography	Accuracy	71.37% +/- 1.87%	74.85% +/- 3.475	77.19 % +/- 12.59	**78.56% +/- 2.89%**	68.26% +/- 2.59%	75.49% +/- 3.52%
	Rank	5	4	2	**1**	6	3
Mushroom	Accuracy	97.14% +/- 0.42%	97.93% +/- 0.561	**100.0 % +/- 0.0**	96.27% +/- 0.75%	97.91% +/- 0.45%	97.85% +/- 0.31%
	Rank	5	2	**1**	6	3	4
Segment	Accuracy	80.04% +/- 1.4%	**84.76% +/- 0.846**	82.08 % +/- 4.64	76.88% +/- 0.85%	82.99% +/- 1.24%	83.33% +/- 1.07%
	Rank	5	**1**	4	6	3	2
Sonar	Accuracy	75.61% +/- 2.64%	77.88% +/- 2.482	54.86 % +/- 3.87	72.1% +/- 4.01%	75.09% +/- 3.63%	**78.42% +/- 2.73%**
	Rank	3	2	6	5	4	**1**
Tic-tac-toe	Accuracy	73.58% +/- 1.72%	72.23% +/- 1.361	**100.0 % +/- 0.0**	71.59% +/- 1.57%	72.33% +/- 1.4%	75.45% +/- 2.1%
	Rank	3	5	**1**	6	4	2

Table 5.6 shows that the GA-AntMiner has the lowest number of discovered rules in all datasets for 10-fold cross-validation compared with Ant-Miner. The GA-AntMiner obtains the lowest results in 15 datasets compared with CAnt-Miner and Hybrid Pruner classifiers. The GA-AntMiner is better than ACO/PSO2 and TACO-Miner in 14 and 13 datasets, respectively.

In comparison with the other classifiers, the GA-AntMiner succeeds in 12 datasets. In addition, the GA-AntMiner obtains the second best results in three datasets (Lymphography, Mushroom, and Sonar). The second best performance is obtained by ACO/PSO2 and TACO-Miner classifiers, which achieve the lowest numbers of

rules in two datasets. The CAnt-Miner acquires the best result in one dataset (Mushroom).

Table 5.6

Average number of rules (average +/− standard deviation, performance rank) obtained using 10-fold cross-validation methods for all the classifiers and GA-AntMiner

Dataset		Ant-Miner	CAnt-Miner	ACO/ PSO2	TACO-Miner	Hybrid Pruner	GA-AntMiner
Balance Scale	Accuracy	8 +/- 0	7.89 +/- 0.33	22 +/- 0	6.7 +/- 0.26	8 +/- 0	**6.6 +/- 0.22**
	Rank	4	3	6	2	4	**1**
Breast Cancer (Ljubljana)	Accuracy	6.1 +/- 0.1	5.90 +/- 0.32	11.3 +/- 2.05	6.4 +/- 0.45	6.5 +/- 0.22	**5 +/- 0.15**
	Rank	3	2	6	4	5	**1**
Breast Cancer (Wisconsin)	Accuracy	7.7 +/- 0.21	7.0 +/- 0.0	9.9 +/- 1.37	6.8 +/- 0.25	7.4 +/- 0.22	**6.4 +/- 0.16**
	Rank	5	3	6	2	4	**1**
Credit-a	Accuracy	7.6 +/- 0.16	6.90 +/- 0.32	20.1 +/- 1.37	7.2 +/- 0.33	7.4 +/- 0.4	**5.9 +/- 0.28**
	Rank	5	2	6	3	4	**1**
Credit-g	Accuracy	9.4 +/- 0.22	8.9 +/- 0.87	12.9 +/- 4.84	8.5 +/- 0.17	9.1 +/- 0.31	**7.8 +/- 0.13**
	Rank	5	3	6	2	4	**1**
Diabetes	Accuracy	9.7 +/- 0.21	9.5 +/- 1.08	37.1 +/- 2.23	9.3 +/- 0.3	9.3 +/- 0.26	**8.5 +/- 0.22**
	Rank	5	4	6	2	2	**1**
Heart (Cleveland)	Accuracy	6.3 +/- 0.21	6.3 +/- 0.82	10.6 +/- 1.42	**5.6 +/- 0.16**	6.4 +/- 0.27	**5.6 +/- 0.22**
	Rank	3	3	6	**1**	5	**1**
Heart (Statlog)	Accuracy	6.3 +/- 0.15	5.8 +/- 0.78	9.7 +/- 1.70	5.7 +/- 0.15	6 +/- 0.21	**5.3 +/- 0.15**
	Rank	5	3	6	2	4	**1**
Hepatitis	Accuracy	5.6 +/- 0.22	5 +/- 0.47	5.5 +/- 0.84	5.2 +/- 0.2	5.3 +/- 0.3	**4.9 +/- 0.23**

Ionosphere	Rank	6	2	5	3	4	**1**
	Accuracy	6.2 +/- 0.2	6.2 +/- 0.63	**1.0 +/- 0**	4.8 +/- 0.33	6 +/- 0.21	5.8 +/- 0.13
Iris	Rank	5	5	**1**	2	4	3
	Accuracy	4.4 +/- 0.27	4.3 +/- 0.67	4.7 +/- 0.48	4.3 +/- 0.21	4.3 +/- 0.21	**4 +/- 0**
Lymphography	Rank	5	2	6	2	2	**1**
	Accuracy	6 +/- 0.21	6.9 +/- 0.87	15.4 +/- 1.34	**5.1 +/- 0.23**	5.5 +/- 0.27	5.5 +/- 0.22
Mushroom	Rank	4	5	6	**1**	2	2
	Accuracy	8.1 +/- 0.53	**5.7 +/- 0.9**	17.6 +/- 1.42	7.5 +/- 0.34	8.5 +/- 0.22	6.9 +/- 0.18
Segment	Rank	4	**1**	6	3	5	2
	Accuracy	16.5 +/- 0.5	18.1 +/- 2.60	33.2 +/- 4.36	18.5 +/- 0.75	18.9 +/- 0.55	**15.3 +/- 0.3**
Sonar	Rank	2	3	6	4	5	**1**
	Accuracy	5.7 +/- 0.15	5.4 +/- 0.51	**1.4 +/- 0.51**	5.8 +/- 0.2	5.9 +/- 0.18	5.1 +/- 0.18
Tic-tac-toe	Rank	4	3	**1**	5	6	2
	Accuracy	8.4 +/- 0.54	8.1 +/- 2.46	18.9 +/- 2.13	7.1 +/- 0.41	9.1 +/- 0.48	**5.7 +/- 0.33**
	Rank	4	3	6	2	5	**1**

Table 5.7 shows that the GA-AntMiner achieves the best results for model size in all datasets compared with the Ant-Miner classifier. By using the same token, the GA-AntMiner achieves the best results in 15 datasets compared with CAnt-Miner and Hybrid Pruner classifiers. The GA-AntMiner gains over 14 datasets compared with ACO/PSO2. However, the GA-AntMiner and TACO-Miner classifiers are in the same fashion with the highest result in eight datasets.

In comparison with other classifiers, the GA-AntMiner achieves the best results in nine datasets. The GA-AntMiner obtains the second best results in four datasets

(Balance Scale, Heart-Cleveland, Heart-Statlog and Mushroom). Meanwhile, the second best classifier is TACO-Miner with five datasets. ACO/PSO2 and CAnt-Miner achieve the best results in two datasets and one dataset, respectively. Furthermore, the Ant-Miner and Hybrid Pruner classifiers obtain no highest results compared with other classifiers.

Table 5.7

Average model size (average +/− standard deviation, performance rank) obtained using 10-fold cross-validation method for all classifiers and GA-AntMiner

Dataset		Ant-Miner	CAnt-Miner	ACO/ PSO2	TACO-Miner	Hybrid Pruner	GA-AntMiner
Balance Scale	Accuracy	11 +/- 0	11 +/- 1	52 +/- 0	**6.6 +/- 0.48**	11 +/- 0	9.2 +/- 0.61
	Rank	3	3	6	**1**	3	2
Breast Cancer (Ljubljana)	Accuracy	7.8 +/- 0.29	7.80 +/- 1.14	26.8 +/- 6.196	7.7 +/- 0.5	8.7 +/- 0.62	**6.7 +/- 0.42**
	Rank	3	3	6	2	5	**1**
Breast Cancer (Wisconsin)	Accuracy	8.4 +/- 0.22	8.0 +/- 0.94	17.1 +/- 2.42	7.1 +/- 0.38	7.4 +/- 0.31	**6.5 +/- 0.22**
	Rank	5	4	6	2	3	**1**
Credit-a	Accuracy	10.6 +/- 0.4	10.0 +/- 1.83	70.6 +/- 7.6	8.6 +/- 0.5	10.3 +/- 0.58	**8.3 +/- 0.8**
	Rank	5	3	6	2	4	**1**
Credit-g	Accuracy	14.7 +/- 0.58	16.4 +/- 4.24	30.5 +/- 16.33	13 +/- 0.56	13.2 +/- 0.77	**12.2 +/- 0.51**
	Rank	4	5	6	2	3	**1**
Diabetes	Accuracy	10.5 +/- 0.4	11 +/- 1.94	112.5 +/- 9.312	**9.5 +/- 0.4**	11.3 +/- 0.75	**9.5 +/- 0.37**
	Rank	3	4	6	**1**	5	**1**
Heart (Cleveland)	Accuracy	9.6 +/- 0.69	10 +/- 2.49	28.3 +/- 4.347	**7.7 +/- 0.54**	9.2 +/- 0.8	8.4 +/- 0.64
	Rank	4	5	6	**1**	3	2

Heart (Statlog)	Accuracy	9.6 +/- 0.58	7.8 +/- 1.39	25.9 +/- 4.30	**5.7 +/- 0.26**	8.7 +/- 0.62	7.5 +/- 0.37
	Rank	5	3	6	**1**	4	2
Hepatitis	Accuracy	8.1 +/- 0.48	7.9 +/- 1.44	11.6 +/- 2.31	7.8 +/- 0.65	8.1 +/- 0.71	**7.5 +/- 0.52**
	Rank	4	3	6	2	4	**1**
Ionosphere	Accuracy	7.4 +/- 0.64	6.7 +/- 0.94	**2.2 +/- 0.42**	5.9 +/- 0.92	7.1 +/- 0.46	6.2 +/- 0.42
	Rank	6	4	**1**	2	5	3
Iris	Accuracy	3.4 +/- 0.27	3.4 +/- 0.84	3.3 +/- 0.94	3.3 +/- 0.21	3.4 +/- 0.27	**3 +/- 0.15**
	Rank	4	4	2	2	4	**1**
Lymphography	Accuracy	9.1 +/- 0.5	11.2 +/- 2.39	42.8 +/- 6.48	**5.2 +/- 0.39**	8.8 +/- 0.65	8.9 +/- 0.77
	Rank	4	5	6	**1**	2	3
Mushroom	Accuracy	9.3 +/- 1.25	**4.7 +/- 0.94**	33.4 +/- 2.87	7 +/- 0.26	8.2 +/- 0.47	7 +/- 0.45
	Rank	5	**1**	6	2	4	2
Segment	Accuracy	21.9 +/- 0.69	22.5 +/- 4.30	59.3 +/- 7.9	20.9 +/- 1.39	24.8 +/- 1.2	**20.1 +/- 0.82**
	Rank	3	4	6	2	5	**1**
Sonar	Accuracy	10 +/- 0.49	9.7 +/- 1.15	**0.9 +/- 1.97**	7.5 +/- 0.37	10.4 +/- 0.62	9 +/- 0.56
	Rank	5	4	**1**	2	6	3
Tic-tac-toe	Accuracy	10.7 +/- 1.69	10.7+/- 4.34	53.6 ± 7.306	6.8 +/- 0.81	12.6 +/- 1.31	**4.3 +/- 0.7**
	Rank	3	3	6	2	5	**1**

An example of the final classification model from the Credit-a dataset introduced by GA-AntMiner and other compared classifiers are shown in the following Figure 5.5.

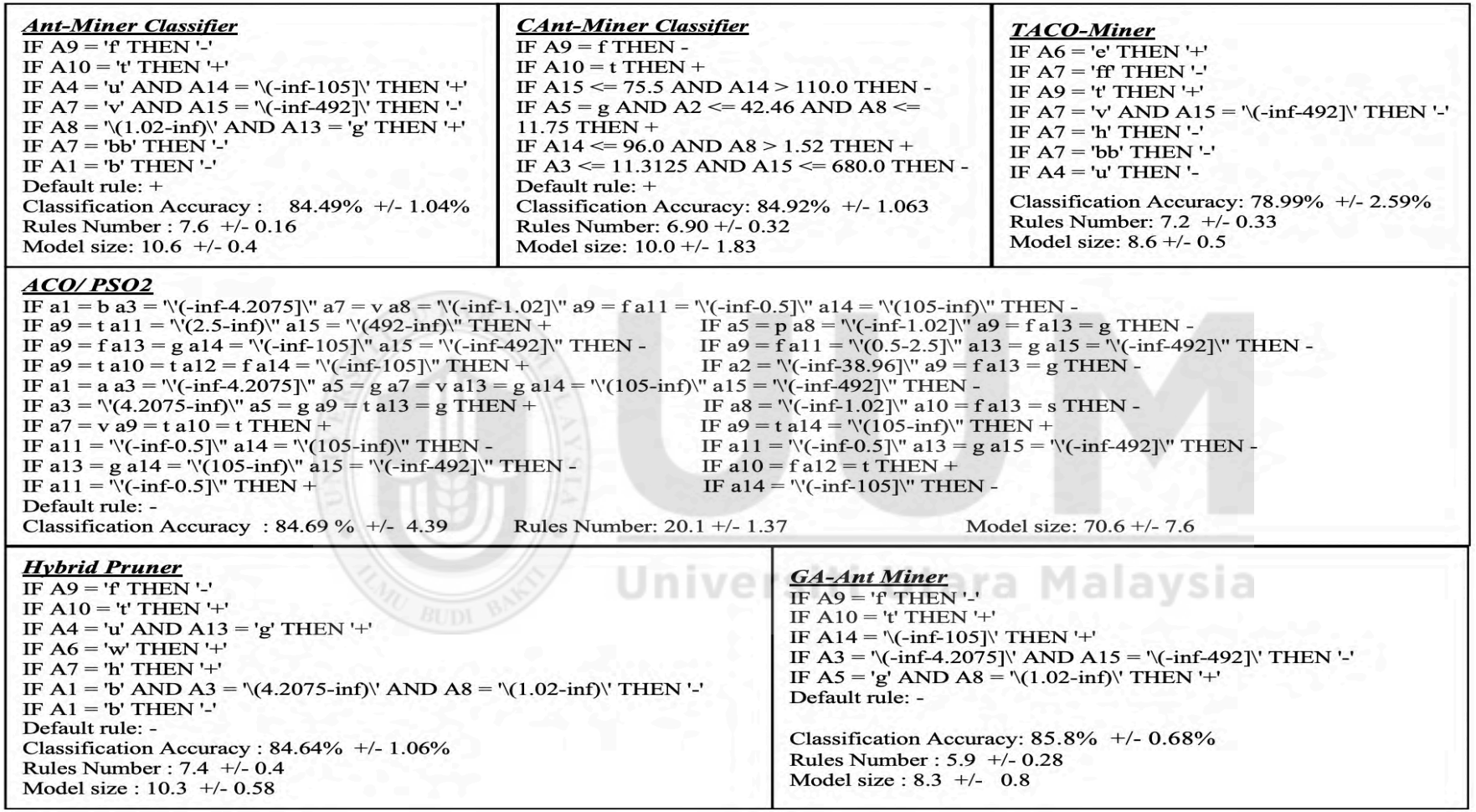

Figure 5.5. An example of the classification model obtained by GA-AntMiner and the other classification algorithms

Table 5.8 and Figures 5.6 and 5.7 show the results of the nonparametric Friedman test with Holm's post-hoc test to illustrate the second benchmark scenario. For the purpose of this evaluation test, the average classification accuracy rank, average number of discovered rule rank, and average model size rank of the statistical results across the 16 datasets are computed and listed in Table 5.8.

Figure 5.6 displays the results of the average classification accuracy rank versus the average number of rule rank. Figure 5.7 lists the results of the average classification accuracy rank versus the model size rank. In all cases, the lowest rank indicates a good algorithm performance.

Table 5.8

Test results of GA-AntMiner and other classifiers based on average performance rank on all datasets

	Ant-Miner	**CAnt-Miner**	**ACO/PSO 2**	**TACO-Miner**	**Hybrid Pruner**	**GA-AntMiner**
Accuracy	4.09375	3.125	3.59375	4.53125	4.09375	**1.5625**
Rule	4.40625	3.0625	5.3125	2.625	4.21875	**1.375**
Terms	4.34375	3.8125	5.15625	1.78125	4.21875	**1.6875**

Figure 5.6 shows that the results obtained by the GA-AntMiner classifier outperform those by the other five classifiers in terms of classification accuracy and the number of discovered rules. Therefore, the GA-AntMiner has a dominant result compared with the other classifiers.

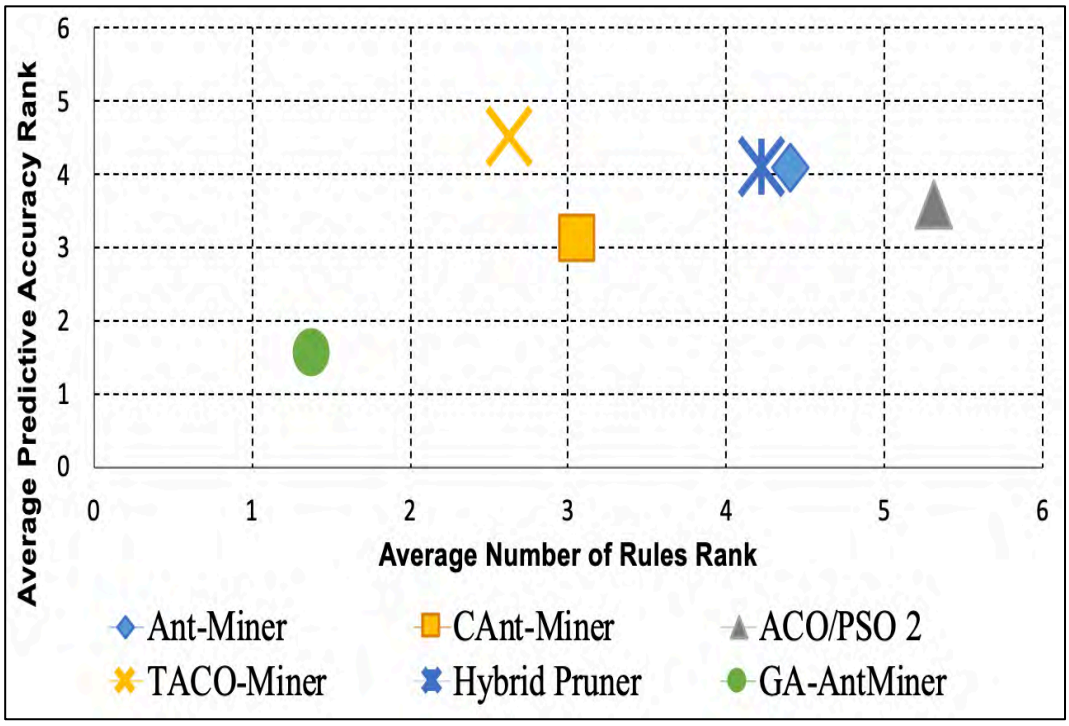

Figure 5.6. Results of GA-AntMiner on the average classification accuracy rank versus the average number of discovered rule rank

Figure 5.7 proves that the results obtained by the GA-AntMiner outperform those by the other classifiers when considering the classification accuracy and model size ranks. GA-AntMiner performs only slightly better than the TACO-Miner classifier in terms of model size, but is better than TACO-Miner and the other classifiers in terms of classification accuracy. Therefore, the GA-AntMiner is the dominant classifier that balances the classification accuracy and model size. This result is due to the enhancement on the post-pruning technique by using the concepts of the GA algorithm (i.e., crossover and mutation) to overcome the nesting effect problem and find the best fitting rule by minimising the number of terms based on the classification accuracy.

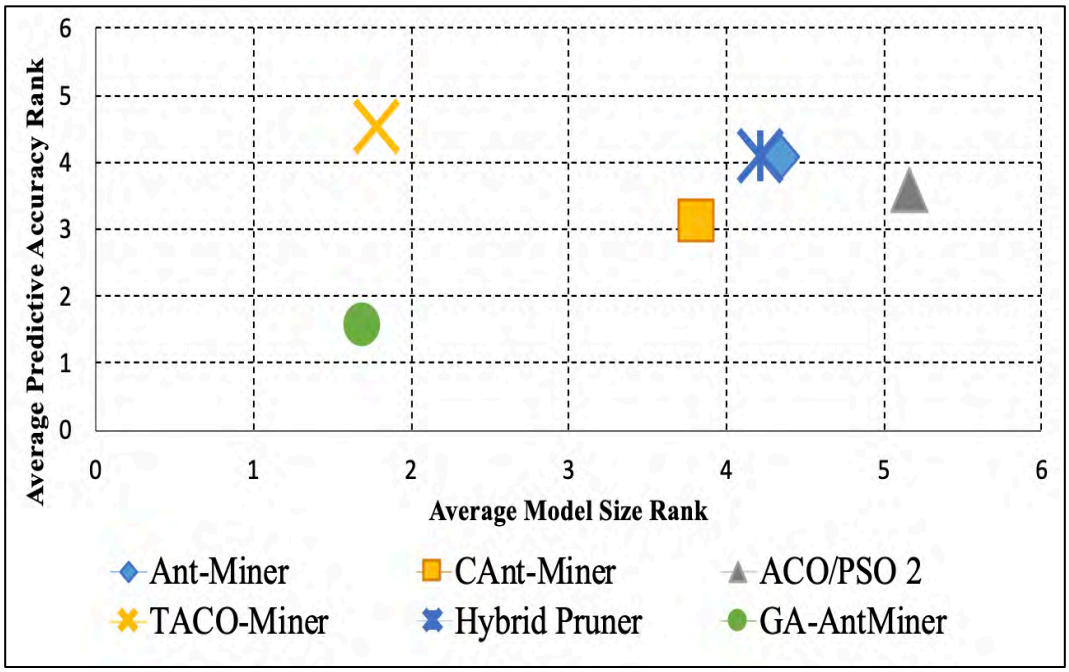

Figure 5.7. Results of GA-AntMiner on the average classification accuracy rank versus the average model size rank

5.5 Results and Analysis of the ILS-AntMiner Classifier

The implementation of hybridising the Ant-Miner with ILS or ILS-AntMiner, is evaluated with two other hybrid classifiers, namely ACO/PSO2 (Holden & Freitas, 2008) and ACO/SA (Saian, 2013; Saian & Ku-Mahamud, 2012). These classifiers are considered to be the most-related classifiers in the ant-mining literature. The classification performance is then analysed in detail for these algorithms by using 16 datasets from UCI under different benchmark scenarios. In the first stage, Tables 5.9, 5.10, and 5.11 show the experimental results of the average classification accuracy, average number of discovered rules, and model size by using the 10-fold cross-validation method. In each table the first row presents the average classification accuracy, and the numbers after the symbol "+/−" are standard deviations. For each dataset, the best result is written in bold. The second row displays the performance

rank for each dataset. The experimental results in Tables 5.9, 5.10, and 5.11 are used to determine the best classifiers.

As shown in Table 5.9, the ILS-AntMiner generates the highest classification accuracy compared with the other two classifiers. The ILS-AntMiner achieves the best results in 14 and eight datasets compared with ACO/PSO2 and ACO/SA, respectively.

The ILS-AntMiner achieves the first best results in eight datasets compared with the other two classifiers. The ILS-AntMiner obtains the second best results in six datasets. The ACO/SA achieves the overall best classification performance in six datasets. Finally, ACO/PSO2 obtains the third place in two datasets.

Table 5.9

Average classification accuracy (average +/− standard deviation, performance rank) obtained using 10-fold cross-validation method for all classifiers and ILS-AntMiner

Dataset		ACO/PSO 2	ACO/SA	ILS-AntMiner
Balance Scale	Accuracy	68.66% +/- 4.97	71.04 % +/- 3.91	**71.19% +/- 1.19%**
	Rank	3	2	**1**
Breast Cancer (Ljubljana)	Accuracy	70.94% +/-5.37	72.39 % +/- 9.09	**73.57% +/- 2.91%**
	Rank	3	2	**1**
Breast Cancer (Wisconsin)	Accuracy	93.86% +/-4.56	**96.14 % +/- 2.93**	95.14% +/- 0.53%
	Rank	3	**1**	2

Credit-a	Accuracy	84.69% +/-4.39	85.80 % +/- 2.58	**86.67% +/- 1.58%**
	Rank	3	2	**1**
Credit-g	Accuracy	71.0% +/-4.52	**75.50 % +/- 3.29**	72.4% +/- 1.43%
	Rank	3	**1**	2
Diabetes	Accuracy	76.31% +/-4.32	76.70 % +/- 4.11	**76.93% +/- 2.18%**
	Rank	3	2	**1**
Heart (Cleveland)	Accuracy	78.51% +/-6.16	81.78 % +/- 7.29	**82.28% +/- 2.4%**
	Rank	3	2	**1**
Heart (Statlog)	Accuracy	78.89% +/-7.78	81.11 % +/- 9.14	**82.06% +/- 2.22%**
	Rank	3	2	**1**
Hepatitis	Accuracy	76.13% +/-8.34	**83.25 % +/- 5.79**	78.17% +/- 2.74%
	Rank	3	**1**	2
Ionosphere	Accuracy	65.51% +/-7.46	**90.89 % +/- 3.98**	89.86% +/- 2.09%
	Rank	3	**1**	2
Iris	Accuracy	94.0% +/-8.14	93.33 % +/- 8.43	**96% +/- 1.47%**
	Rank	2	3	**1**
Lymphography	Accuracy	77.19% +/-12.59	78.29 % +/- 6.88	**80.36% +/- 4.62%**
	Rank	3	2	**1**
Mushroom	Accuracy	**100.0% +/-0.0**	99.01 % +/- 2.55	98.02% +/- 0.13%
	Rank	**1**	2	3
Segment	Accuracy	82.08% +/-4.64	**92.42 % +/- 1.60**	91.56% +/- 0.76%
	Rank	3	**1**	2
Sonar	Accuracy	54.86% +/-3.87	**80.36 % +/- 7.40**	76.47% +/- 2.37%
	Rank	3	**1**	2
Tic-tac-toe	Accuracy	**100.0% +/-0.0**	97.18 % +/- 1.88	81.93% +/- 1.52%
	Rank	**1**	2	3

Table 5.10 shows that the ILS-AntMiner has the lowest number of discovered rules in 12 datasets compared with ACO/PSO2. In addition, ILS-AntMiner outperforms the ACO/SA classifier in 14 datasets.

As shown in Table 5.10, the ILS-AntMiner outperforms the other classifiers and obtains the best results in 12 datasets. In addition, ILS-AntMiner achieves the second best results in four datasets. However, the second best performance is obtained by the ACO/PSO2 classifier with four datasets. Meanwhile, ACO/SA obtains the best result in one dataset.

Table 5.10

Average number of rules (average +/− standard deviation, performance rank) obtained using 10-fold cross-validation method for all classifiers and ILS-AntMiner

Dataset		**ACO/PSO 2**	**ACO/SA**	**ILS-AntMiner**
Balance Scale	Accuracy	22 +/- 0	19.20 +/- 1.72	**7.1 +/- 0.14**
	Rank	3	2	**1**
Breast Cancer (Ljubljana)	Accuracy	11.3 +/- 2.05	16.40 +/- 1.02	**8.2 +/- 0.3**
	Rank	2	3	**1**
Breast Cancer (Wisconsin)	Accuracy	9.9 +/- 1.37	11.90 +/-0.83	**8.6 +/- 0.22**
	Rank	2	3	**1**
Credit-a	Accuracy	20.1 +/- 1.37	20.40 +/- 2.33	**13.4 +/- 0.64**
	Rank	2	3	**1**
Credit-g	Accuracy	**12.9 +/- 4.84**	48.00 +/- 3.97	19.9 +/- 0.33
	Rank	**1**	3	2
Diabetes	Accuracy	37.1 +/- 2.23	29.30 +/- 1.10	**17.4 +/- 0.87**
	Rank	3	2	**1**
Heart (Cleveland)	Accuracy	10.6 +/- 1.42	12.80 +/- 0.87	**10.3 +/- 0.37**
	Rank	2	3	**1**
Heart (Statlog)	Accuracy	9.7 +/- 1.70	12.50 +/- 1.12	**9 +/- 0.35**
	Rank	2	3	**1**
Hepatitis	Accuracy	**5.5 +/- 0.84**	7.80 +/- 0.98	7.8 +/- 0.32
	Rank	**1**	2	2
Ionosphere	Accuracy	**1.0 +/- 0**	12.60 +/- 1.36	7.1 +/- 0.33
	Rank	**1**	3	2
Iris	Accuracy	4.7 +/- 0.48	4.70 +/- 0.46	**4.5 +/- 0.27**
	Rank	2	2	**1**
Lymphography	Accuracy	15.4 +/- 1.34	**7.90 +/- 0.83**	**7.9 +/- 0.15**
	Rank	3	**1**	**1**

Mushroom	Accuracy	17.6 +/- 1.42	24.90 +/- 1.76	**7.7 +/- 0.17**
	Rank	2	3	**1**
Segment	Accuracy	33.2 +/- 4.36	57.60 +/- 2.42	**27.4 +/- 0.4**
	Rank	2	3	**1**
Sonar	Accuracy	**1.4 +/- 0.51**	11.90 +/- 1.04	9.2 +/- 0.52
	Rank	**1**	3	2
Tic-tac-toe	Accuracy	18.9 +/- 2.13	33.30 +/- 3.32	**12.2 +/- 1.09**
	Rank	2	3	**1**

Table 5.11 shows the simplicity (the least number of terms per rule) of the constructed rules. The ILS-AntMiner dominates on 11 datasets compared with the ACO/PSO2 classifier. In addition, the ILS-AntMiner obtains the best results in 14 datasets compared with ACO/SA.

In comparison with the other classifiers, the ILS-AntMiner achieves the best results with 11 datasets and obtains the second best results in four datasets. Meanwhile, the second best performance is achieved by the ACO/PSO2 classifier with five datasets. ACO/SA obtains no highest results compared with the other classifiers.

Table 5.11

Average model size (average +/− standard deviation, performance rank) obtained using 10-fold cross-validation method for all classifiers and ILS-AntMiner

Dataset		**ACO/PSO 2**	**ACO/SA**	**ILS-AntMiner**
Balance Scale	Accuracy	52 +/- 0	42.90 +/- 5.15	**8.9 +/- 0.38**
	Rank	3	2	**1**
Breast Cancer (Ljubljana)	Accuracy	26.8 +/- 6.196	33.20 +/- 3.74	**13.1 +/- 0.81**
	Rank	2	3	**1**
Breast Cancer (Wisconsin)	Accuracy	17.1 +/- 2.42	18.90 +/- 2.02	**9.8 +/- 0.49**
	Rank	2	3	**1**

Credit-a	Accuracy	70.6 +/- 7.6	53.50 +/- 8.88	**25.2 +/- 1.98**
	Rank	3	2	**1**
Credit-g	Accuracy	**30.5+/- 16.33**	127.90 +/- 14.82	45.9 +/- 1.28
	Rank	**1**	3	2
Diabetes	Accuracy	112.5 +/- 9.312	65.70 +/- 3.90	**29.6 +/- 2.09**
	Rank	3	2	**1**
Heart (Cleveland)	Accuracy	28.3 +/- 4.347	29.10 +/- 3.53	**20.9 +/- 1.14**
	Rank	2	3	**1**
Heart (Statlog)	Accuracy	25.9 +/- 4.30	27.60 +/- 3.98	**17.5 +/- 1.08**
	Rank	2	3	**1**
	Accuracy	**11.6 +/- 2.31**	15.70 +/- 3.47	16.2 +/- 1.08
Hepatitis	Rank	**1**	2	3
Ionosphere	Accuracy	**2.2 +/- 0.42**	24.50 +/- 3.75	12.7 +/- 0.87
	Rank	**1**	3	2
Iris	Accuracy	**3.3 +/- 0.94**	4.80 +/- 1.08	4.1 +/- 0.48
	Rank	**1**	3	2
Lymphography	Accuracy	42.8 +/- 6.48	16.50 +/- 2.97	**16.2 +/- 0.63**
	Rank	3	2	**1**
Mushroom	Accuracy	33.4 +/- 2.87	37.00 +/- 2.90	**7.7 +/- 0.15**
	Rank	2	3	**1**
Segment	Accuracy	59.3 +/- 7.9	121.60 +/- 5.97	**43 +/- 1.26**
	Rank	2	3	**1**
Sonar	Accuracy	**0.9 +/- 1.97**	25.70 +/- 3.61	21 +/- 1.37
	Rank	**1**	3	2
Tic-tac-toe	Accuracy	53.6 +/- 7.306	96.50 +/- 11.14	**20.6 +/- 2.13**
	Rank	2	3	**1**

An example of the final classification model from the Credit-a dataset introduced by ILS-AntMiner and other compared classifiers are shown in Figure 5.8.

ACO/ PSO2

IF a1 = b a3 = '\'(-inf-4.2075]\" a7 = v a8 = '\'(-inf-1.02]\" a9 = f a11 = '\'(-inf-0.5]\" a14 = '\'(105-inf)\" THEN -
IF a9 = t a11 = '\'(2.5-inf)\" a15 = '\'(492-inf)\" THEN +
IF a9 = f a13 = g a14 = '\'(-inf-105]\" a15 = '\'(-inf-492]\" THEN -
IF a9 = t a10 = t a12 = f a14 = '\'(-inf-105]\" THEN +
IF a1 = a a3 = '\'(-inf-4.2075]\" a5 = g a7 = v a13 = g a14 = '\'(105-inf)\" a15 = '\'(-inf-492]\" THEN -
IF a3 = '\'(4.2075-inf)\" a5 = g a9 = t a13 = g THEN +
IF a7 = v a9 = t a10 = t THEN +
IF a11 = '\'(-inf-0.5]\" a14 = '\'(105-inf)\" THEN -
IF a13 = g a14 = '\'(105-inf)\" a15 = '\'(-inf-492]\" THEN -
IF a11 = '\'(-inf-0.5]\" THEN +
IF a5 = p a8 = '\'(-inf-1.02]\" a9 = f a13 = g THEN -
IF a9 = f a11 = '\'(0.5-2.5]\" a13 = g a15 = '\'(-inf-492]\" THEN -
IF a2 = '\'(-inf-38.96]\" a9 = f a13 = g THEN -
IF a8 = '\'(-inf-1.02]\" a10 = f a13 = s THEN -
IF a9 = t a14 = '\'(105-inf)\" THEN +
IF a11 = '\'(-inf-0.5]\" a13 = g a15 = '\'(-inf-492]\" THEN -
IF a10 = f a12 = t THEN +
IF a14 = '\'(-inf-105]\" THEN -
Default rule: -
Classification Accuracy : 84.69 % +/- 4.39 Rules Number: 20.1 +/- 1.37 Model size: 70.6 +/- 7.6

ACO/ SA

IF A4 = 'u' AND A8 = '\(1.02-inf)\' AND A13 = 'g' AND A14 = '\(105-inf)\' THEN '+'
IF A4 = 'u' AND A8 = '\(1.02-inf)\' AND A10 = 't' THEN '+'
IF A1 = 'b' AND A4 = 'u' AND A7 = 'v' AND A13 = 'g' THEN '+'
IF A1 = 'b' AND A7 = 'v' AND A8 = '\(1.02-inf)\' AND A13 = 's' THEN '+'
IF A7 = 'v' AND A8 = '\(-inf-1.02]\' AND A13 = 'g' THEN '+'
IF A4 = 'y' AND A10 = 't' AND A12 = 't' THEN '+'
IF A3 = '\(-inf-4.2075]\' AND A6 = 'm' AND A13 = 's' THEN '-'
IF A15 = '\(492-inf)\' THEN '+'
IF A9 = 'f' THEN '-'
IF A7 = 'bb' THEN '-'
IF A4 = 'y' AND A7 = 'h' THEN '+'
IF A4 = 'u' AND A7 = 'h' AND A11 = '\(-inf-0.5]\' THEN '+'
IF A1 = 'b' AND A6 = 'c' AND A13 = 'g' THEN '-'
IF A2 = '\(38.96-inf)\' AND A7 = 'v' THEN '+'
IF A6 = 'aa' AND A13 = 'g' THEN '-'
IF A14 = '\(-inf-105]\' THEN '+'
IF A6 = 'w' THEN '+'
IF A12 = 't' THEN '-'
Default rule: -
Classification Accuracy: 85.80 % +/-2.58 Rules Number: 20.40 +/- 2.33 Model size: 53.50 +/- 8.88

ILS-AntMiner

IF A10 = 't' THEN '+'
IF A6 = 'w' THEN '+'
IF A4 = 'y' AND A13 = 'g' AND A15 = '\(-inf-492]\' THEN '-'
IF A8 = '\(-inf-1.02]\' AND A13 = 's' THEN '-'
IF A2 = '\(38.96-inf)\' AND A3 = '\(-inf-4.2075]\' AND A15 = '\(-inf-492]\' THEN '-'
IF A1 = 'a' AND A8 = '\(1.02-inf)\' THEN '-'
IF A4 = 'u' THEN '+'
IF A4 = 'u' AND A14 = '\(-inf-105]\' THEN '+'
IF A7 = 'h' THEN '+'
IF A7 = 'bb' AND A12 = 't' THEN '-'
IF A3 = '\(4.2075-inf)\' AND A4 = 'u' AND A8 = '\(-inf-1.02]\' THEN '-'
Default rule: +
Default rule: -
Classification Accuracy : 86.67% +/- 1.58% Rules Number: 13.4 +/- 0.64 Model size: 25.2 +/- 1.98

Figure 5.8. An example of the classification model obtained by ILS-AntMiner and the other classification algorithms

Table 5.12 and Figures 5.9 and 5.10 show the evaluation of the performance based on the nonparametric Friedman test with Holm's post-hoc test. This test aims to find the dominant classifier among the 16 datasets. As shown in Table 5.12, in all cases, the lowest rank indicates a good algorithm performance. Thus, ILS-AntMiner obtains the best average rank in classification accuracy, number of discovered rules, and model size.

Figure 5.9 presents the results of the average classification accuracy rank versus the average number of rules rank. The ILS-AntMiner achieves the best classification accuracy and best number of discovered rules. Figure 5.10 displays the results of the average classification accuracy rank versus the model size rank. The ILS-AntMiner obtains the best classification accuracy and best model size. Under these circumstances, the ILS-AntMiner dominates the hybridisation with Ant-Miner classifier in all evaluation criteria. This result is due to the enhancement process achieved by the ILS algorithm to the rule produced by the Ant-Miner classifier. ILS uses the power of the multiple neighbourhood structures (i.e., perturbation and local search) procedures to escape from the local optima.

Table 5.12

Results of ILS-AntMiner and other classifiers based on average performance rank on all datasets

	ACO/PSO 2	**ACO/SA**	**ILS-AntMiner**
Accuracy	2.6875	1.6875	**1.625**
Rule	1.96875	2.71875	**1.3125**
Terms	1.9375	2.6875	**1.375**

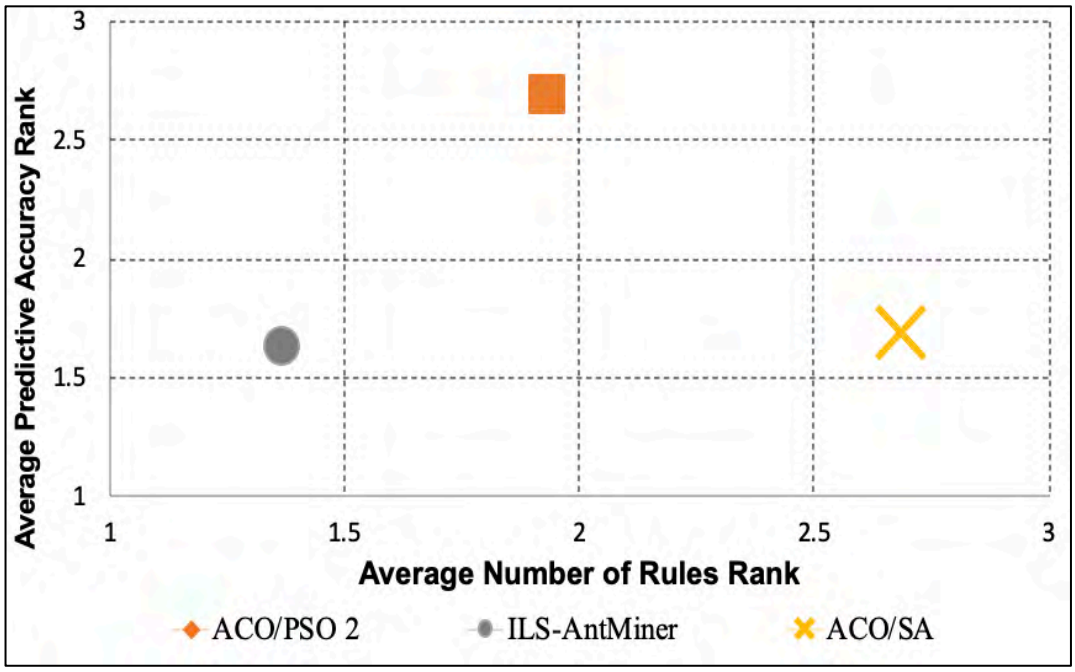

Figure 5.9. Results of ILS-AntMiner on the average classification accuracy rank versus the average number of discovered rules rank

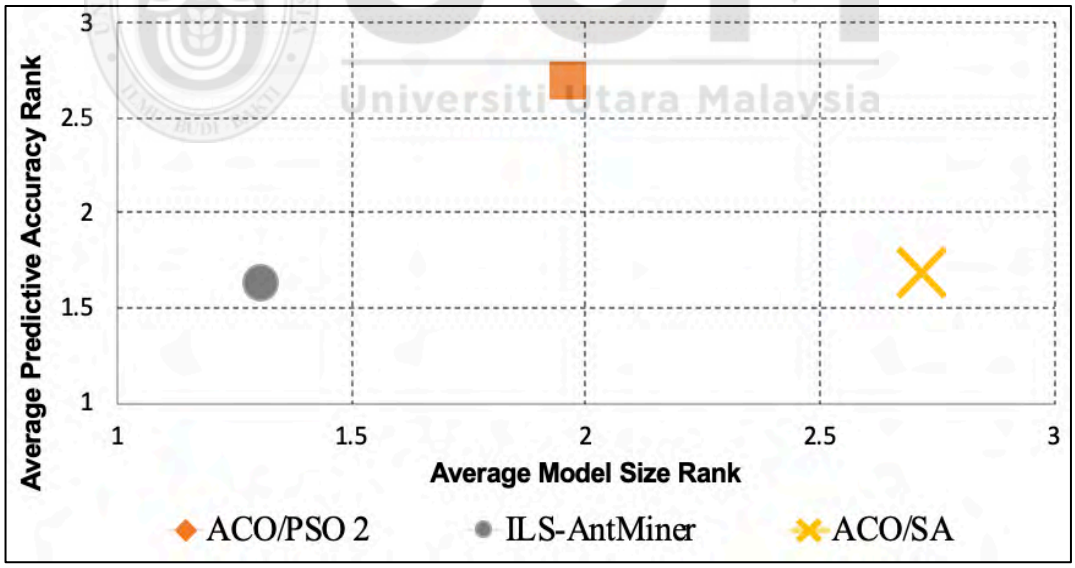

Figure 5.10. Results of ILS-AntMiner on the average classification accuracy rank versus the average model size rank

5.6 Results and Analysis of AGI-AntMiner for Classification

The current section presents the evaluation results of the AGI-AntMiner algorithm for rules-based classification. This classifier consists of three modifications, tested in the previous sections, to overcome the weakness and limitations of the Ant-Miner classifier. The proposed classifier was tested on the same datasets from the UCI repository by using the 10-fold cross-validation technique. The computational experiments compared the proposed classifier and the state-of-the art rule induction algorithms as well as ant-mining classification algorithms. The experiment of the cross-validation method was repeated 10 times and the program was run 10 times to enable each fold of data to take a turn as the testing dataset, producing 10 separate sets of performance statistics, such as classification accuracy, number of discovered rules, and model size. Finally, the experiment averaged these performance statistics and calculated the standard deviations for each of the performance statistics.

Tables 5.13, 5.14, and 5.15 show the adaptive ACO performance using 10-fold cross-validation. The performance of the proposed algorithm are evaluated using average classification accuracy, average number of discovered rules, and model size. In each table, the first row presents the average classification accuracy, and the numbers after the symbol "+/–" are standard deviations. For each dataset, the best result is written in bold. The second row displays the performance rank for each dataset. The experimental results in Tables 5.13, 5.14, and 5.15 are used to determine the best classifiers. The AGI-AntMiner outperforms the other classifiers in terms of classification accuracy. The classification results show that the AGI-AntMiner is better than the Ant-Miner and Conjunctive Rule classifiers in all datasets. The

proposed classifier can produce high classification accuracy in 14 datasets compared with ACO/PSO2, CAnt-MinerPB, and OneR classifiers. In addition, the AGI-AntMiner classifier achieves better results than Decision Table, FURIA, and PART classifiers over 13, 12, and 11 datasets, respectively. However, the performance of our classifier is competitive with that of ACO/SA with eight datasets for each classifier.

Table 5.13 shows the essence of the classification accuracy of the AGI-AntMiner. Considering the classification accuracy over all datasets, the proposed classifier performs best over seven datasets (Balance Scale, Breast Cancer [Ljubljana], Credit-a, Diabetes, Heart [Cleveland], Iris, and Lymphography). The second best classifiers are ACO/SA and FURIA with the best accuracy in three datasets. Meanwhile, ACO/PSO2, PART, and Decision Table classifiers obtain the highest classification accuracy in two datasets. Conversely, the Ant-Miner, CAnt-MinerPB, Conjunctive Rule, Conjunctive Rule, and OneR classifiers still obtain low accuracy results in all datasets.

Table 5.13

Average classification accuracy (average +/− standard deviation, performance rank) obtained using 10-fold cross-validation method for all classifiers with the proposed adaptive ACO classifier

Dataset		Ant-Miner	ACO/PSO 2	ACO/SA	CAnt-MinerPB	PART	FURIA	Conjunctive Rule	Decision Table	OneR	AGI-AntMiner
Balance Scale	Accuracy	69.73% +/- 1.58%	68.66 % +/- 4.97	71.04 +/- 3.91	70.238 % +/- 1.607	69.92 % +/- 0.25	70.24 % +/-0.37	60.96 % +/- 0.41	71.04 % +/- 0.37	56.32%	**72.93% +/- 1.87%**
	Rank	7	8	2	5	6	4	9	2	10	**1**
Breast Cancer (Ljubljana)	Accuracy	72.32% +/- 1.73%	70.94 % +/- 5.37	72.39 % +/- 9.09	72.833 % +/- 2.872	71.32 % +/- 0.36	73.076 % +/- 0.49	65.73 % +/- 0.46	73.42 % +/- 0.44	65.73%	**75.56% +/- 2.59%**
	Rank	6	8	5	4	7	3	9	2	9	**1**
Breast Cancer (Wisconsin)	Accuracy	94.43% +/- 1.17%	93.86 % +/- 4.56	96.14 % +/- 2.93	95.565 % +/- 0.688	94.70 % +/- 0.06	**96.28 % +/- 0.19**	88.26 % +/- 0.30	95.27 % +/- 0.19	92.41%	95.57% +/- 0.84%
	Rank	7	8	2	4	6	**1**	10	5	9	3
Credit-a	Accuracy	84.49% +/- 1.04%	84.69 % +/- 4.39	85.80 % +/- 2.58	84.348 % +/- 1.009	86.37 % +/- 0.19	86.52 % +/-0.36	85.50 % +/- 0.34	84.63 % +/- 0.32	85.50%	**87.1% +/- 0.93%**
	Rank	9	7	4	10	3	2	5	8	5	**1**

Credit-g	Accuracy	70.7% +/- 1%	71.0 % +/- 4.52	**75.50 % +/- 3.29**	72.00 % +/- 1.282	71.9 % +/- 0.32	72.8 % +/- 0.4	69.7 % +/- 0.45	71.8 % +/- 0.43	71.70%	73.3% +/- 1.73%
	Rank	9	8	**1**	4	5	3	10	6	7	2
Diabetes	Accuracy	71.12% +/- 2.01%	76.31 % +/- 4.32	76.70 % +/- 4.11	76.832 % +/- 1.560	76.3 % +/- 0.30	77.3438 % +/-0.4	72.13 % +/- 0.44	76.17 % +/- 0.39	74.73%	**78.15% +/- 1.49%**
	Rank	10	5	4	3	6	2	9	7	8	**1**
Heart (Cleveland)	Accuracy	76.17% +/- 2.85%	78.51 % +/- 6.16	81.78 % +/- 7.29	78.52 % +/- 1.167	78.8 % +/- 0.09	83.1683 % +/- 0.24	72.27 % +/- 0.28	77.55 % +/- 0.26	71.61%	**83.49% +/- 1.83%**
	Rank	8	6	3	5	4	2	9	7	10	**1**
Heart (Statlog)	Accuracy	77.78% +/- 2.41%	78.89 % +/- 7.78	81.11 % +/- 9.14	80.370 % +/- 3.027	81.85 % +/- 0.23	81.85 % +/-0.39	73.70 % +/- 0.43	**85.18 % +/- 0.35**	72.59%	82.22% +/- 2.91%
	Rank	8	7	5	6	3	3	9	**1**	10	2
Hepatitis	Accuracy	80.03% +/- 3.68%	76.13 % +/- 8.34	**83.25 % +/- 5.79**	80.792 % +/- 2.738	82.58 % +/- 0.21	79.35 % +/-0.4	79.35 % +/- 0.4	76.12 % +/- 0.38	83.22%	82.04% +/- 3.84%
	Rank	6	9	**1**	5	3	7	7	10	2	4
Ionosphere	Accuracy	86.03% +/- 1.77%	65.51 % +/- 7.46	**90.89 % +/- 3.98**	90.881 % +/- 1.112	90.02 % +/- 0.12	89.45 % +/- 0.3	80.62 % +/- 0.3	88.03 % +/- 0.29	87.74%	90.54% +/- 1%
	Rank	8	10	**1**	2	4	5	9	6	7	3

Iris	Accuracy	94% +/- 1.85%	94.0 % +/- 8.14	93.33 % +/- 8.43	94.00 % +/- 1.556	95.33 % +/- 0.04	95.33 % +/- 0.18	66.66 % +/- 0.33	94 % +/- 0.19	94%	**96% +/- 1.47%**
	Rank	4	4	9	4	2	2	10	4	4	**1**
Lymphography	Accuracy	71.37% +/- 1.87%	77.19 % +/- 12.59	78.29 % +/- 6.88	74.286 % +/- 3.461	79.72 % +/- 0.11	79.05 % +/- 0.2	73.64 % +/- 0.31	75.67 % +/- 0.30	75%	**81.13% +/- 2.74%**
	Rank	10	5	4	8	2	3	9	6	7	**1**
Mushroom	Accuracy	97.14% +/- 0.42%	**100.0** % +/- **0.0**	99.01 % +/- 2.55	98.929 % +/- 0.214	**100 % +/- 0.0**	**100 % % +/- 0.0**	88.67 % +/- 0.3	**100 % +/- 0.02**	98.52%	98.36% +/- 0.21%
	Rank	9	**1**	5	6	**1**	**1**	10	**1**	7	8
Segment	Accuracy	80.04% +/- 1.4%	82.08 % +/- 4.64	92.42 % +/- 1.60	80.173 % +/- 0.460	94.41 % +/- 0.02	**95.541 % +/- 0.104**	28.57 % +/- 0.31	90.73 % +/- 0.17	65.80%	89.61% +/- 0.86%
	Rank	8	6	3	7	2	**1**	10	4	9	5
Sonar	Accuracy	75.61% +/- 2.64%	54.86 % +/- 3.87	80.36 % +/- 7.40	76.857 % +/- 3.253	**81.25 % +/- 0.19**	77.884 % +/- 0.4	68.75 % +/- 0.46	75.48 % +/- 0.40	74.03%	79.32% +/- 2.13%
	Rank	6	10	2	5	**1**	4	9	7	8	3
Tic-tac-toe	Accuracy	73.58% +/- 1.72%	**100.0 % +/- 0.0**	97.18 % +/- 1.88	75.361 % +/- 1.033	94.25 % +/- 0.06	98.12 % +/- 0.09	69.93 % +/- 0.44	73.38 % +/- 0.41	69.93%	83.18% +/- 1.9%
	Rank	7	**1**	3	6	4	2	9	8	9	5

Table 5.14 presents the results of the average number of rules produced by all classifiers. The number in bold displays the smallest number of rules achieved by all classifiers. On the basis of the results, the AGI-AntMiner obtains the smallest number of rules in 15 datasets compared with the ACO/SA classifier. The proposed classifier achieves in 13 datasets compared with PART and CAnt-MinerPB classification algorithms. Moreover, the AGI-AntMiner achieves the smallest number of discovered rules over 12 and 11 datasets compared with FURIA and ACO/PSO2, respectively. The comparison results show that the Ant-Miner classifier achieves better results than our classifiers in 14 datasets. Furthermore, the evaluation results shows that the Ant-Miner classifier obtains the best results in eight datasets. The second best performance is achieved by ACO/PSO2 in three datasets. The FURIA classifier obtains the smallest number of rules in two datasets. The CAnt-MinerPB, PART, and AGI-AntMiner classifiers achieves only one dataset.

Considering the average size of the classification model (see Table 5.15), which is the average number of terms for the 10-fold cross-validation, the proposed classifier performs better than ACO/SA, PART, FURIA, CAnt-MinerPB, and ACO/PSO2 in 16, 14, 14, 12, and 11 datasets, respectively. Moreover, the AGI-AntMiner is better than Ant-Miner in four datasets. The experimental results show that Ant-Miner obtains the best results in 10 datasets. The second best performance is achieved by the proposed classifier and ACO/PSO2 in two datasets. CAnt-MinerPB and PART classifiers obtain the best result in one dataset. With the same token, ACO/SA and FURIA classifiers do not obtain the highest result in any dataset.

Table 5.14

Average number of rules (average +/− standard deviation, performance rank) obtained using 10- fold cross-validation method for all classifiers and the AGI-AntMiner classifier

Dataset		Ant-Miner	ACO/ PSO 2	ACO/SA	CAnt-MinerPB	PART	FURIA	AGI-AntMiner
Balance Scale	Accuracy	8 +/- 0	22 ± 0	19.20 +/- 1.72	7.3 +/- 0.48	**5.2 ± 0.632**	8	6.5 +/- 0.27
	Rank	4	7	6	3	**1**	4	2
Breast Cancer (Ljubljana)	Accuracy	6.1 +/- 0.1	11.3 ± 2.05	16.40 +/- 1.02	7.4+/- 1.07	19.4 ± 5.10	**4**	7.8 +/- 0.68
	Rank	2	5	6	3	7	**1**	4
Breast Cancer (Wisconsin)	Accuracy	7.7 +/- 0.21	9.9 ± 1.37	11.90 +/-0.83	9.5 +/- 1.64	10.0 ± 1.41	21	**7.1 +/- 0.18**
	Rank	2	4	6	3	5	7	**1**
Credit-a	Accuracy	7.6 +/- 0.16	20.1 ± 1.37	20.40 +/- 2.33	14.1 +/- 1.91	22.1 ± 3.24	**6**	*10.6 +/- 0.34*
	Rank	2	5	6	4	7	**1**	3
Credit-g	Accuracy	**9.4 +/- 0.22**	12.9 ± 4.84	48.00 +/- 3.97	23.1 +/- 3.98	56.3 ± 4.21	20	17 +/- 0.58
	Rank	**1**	2	6	5	7	4	3
Diabetes	Accuracy	**9.7 +/- 0.21**	37.1 ± 2.23	29.30 +/- 1.10	10.50 +/- 0.53	15.8 ± 2.74	11	13.5 +/- 0.65
	Rank	**1**	7	6	2	5	3	4
Heart (Cleveland)	Accuracy	**6.3 +/- 0.21**	10.6 ± 1.42	12.80 +/- 0.87	10.50 +/- 2.07	14.4 ± 2.27	11	7.7 +/- 0.33
	Rank	**1**	4	6	3	7	5	2

Heart (Statlog)	Accuracy	**6.3 +/- 0.15**	9.7 ± 1.70	12.50 +/- 1.12	8.50 +/- 1.78	12.9 ± 2.84	9	7.6 +/- 0.37
	Rank	**1**	5	6	3	7	4	2
Hepatitis	Accuracy	5.6 +/- 0.22	**5.5 ± 0.84**	7.80 +/- 0.98	8.70 +/- 0.67	7.4 ± 1.64	13	7.6 +/- 0.31
	Rank	2	**1**	5	6	3	7	4
Ionosphere	Accuracy	6.2 +/- 0.2	**1.0 ± 0**	12.60 +/- 1.36	11.30 +/- 2.11	9.8 ± 2.08	9	6.7 +/- 0.37
	Rank	2	**1**	7	6	5	4	3
Iris	Accuracy	4.4 +/- 0.27	4.7 ± 0.48	4.70 +/- 0.46	**3.40 +/- 0.84**	4.6 ± 0.84	4	4.8 +/- 0.13
	Rank	3	5	5	**1**	4	2	7
Lymphography	Accuracy	**6 +/- 0.21**	15.4 ± 1.34	7.90 +/- 0.83	8.60 +/- 1.26	11.7 ± 1.15	12	7.4 +/- 0.22
	Rank	**1**	7	3	4	5	6	2
Mushroom	Accuracy	**8.1 +/- 0.53**	17.6 ± 1.42	24.90 +/- 1.76	18.20 +/- 6.71	11.8 ± 1.813	13	8.2 +/- 0.39
	Rank	**1**	5	7	6	3	4	2
Segment	Accuracy	**16.5 +/- 0.5**	33.2 ± 4.36	57.60 +/- 2.42	38.40 +/- 4.97	59.9 ± 5.89	84	27.5 +/- 0.58
	Rank	**1**	3	5	4	6	7	2
Sonar	Accuracy	5.7 +/- 0.15	**1.4 ± 0.51**	11.90 +/- 1.04	10.0 +/- 1.63	13.5 ± 2.01	9	7.8 +/- 0.25
	Rank	2	**1**	6	5	7	4	3
Tic-tac-toe	Accuracy	**8.4 +/- 0.54**	18.9 ± 2.13	33.30 +/- 3.32	12.70 +/- 3.02	40.1 ± 3.28	22	11 +/- 0.49
	Rank	**1**	4	6	3	7	5	2

Table 5.15

Average model size (average +/− standard deviation, performance rank) obtained using 10-fold cross-validation method for all classifiers and the AGI-AntMiner classifier

Dataset		Ant-Miner	ACO/ PSO 2	ACO/SA	CAnt-MinerPB	PART	FURIA	AGI-AntMiner
Balance Scale	Accuracy	11 +/- 0	52 ± 0	42.90 +/- 5.15	9.2 +/- 2.29	**7.4 ± 1.77**	24	8.1 +/- 0.46
	Rank	4	7	6	3	**1**	5	2
Breast Cancer (Ljubljana)	Accuracy	**7.8 +/- 0.29**	26.8 ± 6.196	33.20 +/- 3.74	9.3 +/- 2.98	38.5 ± 12.9	11	11.9 +/- 1.37
	Rank	**1**	5	6	2	7	3	4
Breast Cancer (Wisconsin)	Accuracy	8.4 +/- 0.22	17.1 ± 2.42	18.90 +/- 2.02	10 +/- 2.49	13.6 ± 2.67	55	**7.9 +/- 0.43**
	Rank	2	5	6	3	4	7	**1**
Credit-a	Accuracy	**10.6 +/- 0.4**	70.6 ± 7.6	53.50 +/- 8.88	20.9 +/- 6.96	41.9 ± 8.79	12	12.7 +/- 0.72
	Rank	**1**	7	6	4	5	2	3
Credit-g	Accuracy	**14.7 +/- 0.58**	30.5± 16.33	127.90 +/- 14.82	39.7 +/- 12.12	173 ± 11.30	59	36.2 +/- 1.7
	Rank	**1**	2	6	4	7	5	3
Diabetes	Accuracy	**10.5 +/- 0.4**	112.5 ± 9.312	65.70 +/- 3.90	11.30 +/- 1.34	32.7 ± 7.51	25	17.6 +/- 1.33
	Rank	**1**	7	6	2	5	4	3
Heart (Cleveland)	Accuracy	**9.6 +/- 0.69**	28.3 ± 4.347	29.10 +/- 3.53	19.80 +/- 9.40	33.8 ± 9.48	29	14.8 +/- 1.17
	Rank	**1**	4	6	3	7	5	2

Heart (Statlog)	Accuracy	**9.6 +/- 0.58**	25.9 ± 4.30	27.60 +/- 3.98	13.30 +/- 6.38	30.3 ± 9.68	23	12.8 +/- 0.93
	Rank	**1**	5	6	3	7	4	2
Hepatitis	Accuracy	**8.1 +/- 0.48**	11.6 ± 2.31	15.70 +/- 3.47	11.40 +/- 3.41	14.9 ± 3.31	31	12.1 +/- 0.66
	Rank	**1**	3	6	2	5	7	4
Ionosphere	Accuracy	7.4 +/- 0.64	**2.2 ± 0.42**	24.50 +/- 3.75	12.70 +/- 4.47	15.4 ±5.69	16	7.3 +/- 0.5
	Rank	3	**1**	7	4	5	6	2
Iris	Accuracy	3.4 +/- 0.27	3.3 ± 0.94	4.80 +/- 1.08	**2.50 +/- 1.08**	3.7 ± 0.94	5	4.1 +/- 0.23
	Rank	3	2	6	**1**	4	7	5
Lymphography	Accuracy	**9.1 +/- 0.5**	42.8 ± 6.48	16.50 +/- 2.97	16.10 +/- 5.11	26.2 ± 3.96	25	13 +/- 0.58
	Rank	**1**	7	4	3	6	5	2
Mushroom	Accuracy	9.3 +/- 1.25	33.4 ± 2.87	37.00 +/- 2.90	18.30 +/- 7.63	19 ± 1.490	28	**6.9 +/- 0.41**
	Rank	2	6	7	3	4	5	**1**
Segment	Accuracy	**21.9 +/- 0.69**	59.3 ± 7.9	121.60 +/- 5.97	61.60 +/- 14.58	107.9 ± 14.12	220	34.2 +/- 1.03
	Rank	**1**	3	6	4	5	7	2
Sonar	Accuracy	10 +/- 0.49	**0.9 ± 1.97**	25.70 +/- 3.61	16.10 +/- 6.54	31.9 ± 5.95	26	16 +/- 0.58
	Rank	2	**1**	5	4	7	6	3
Tic-tac-toe	Accuracy	**10.7 +/- 1.69**	53.6 ± 7.306	96.50 +/-11.14	17.70 +/- 7.62	105.22 ± 10.62	76	17.6 +/- 1.33
	Rank	**1**	4	6	3	7	5	2

In accordance with Kudo and Sklansky (2000) for the classification of attributes size in the dataset, the size of the majority of datasets used in our experiments (12 out of 16) are classified as small with sizes in the range of 0–19 attributes. The proposed AGI-AntMiner classifier algorithm produces best results in small size datasets (highest classification accuracy) for 7 datasets out of 12 datasets, which means that 58.3% of the results obtained highest classification accuracy than the best-known results achieved by several classifiers proposed over the years for these datasets.

An example of the final classification model from the Credit-a dataset introduced by ILS-AntMiner and other compared classifiers are shown in the following Figures 5.11a and 5.11b.

Ant-Miner

IF A9 = 'f' THEN '-'
IF A10 = 't' THEN '+'
IF A4 = 'u' AND A14 = '\(-inf-105]\' THEN '+'
IF A7 = 'v' AND A15 = '\(-inf-492]\' THEN '-'
IF A8 = '\(1.02-inf)\' AND A13 = 'g' THEN '+'
IF A7 = 'bb' THEN '-'
IF A1 = 'b' THEN '-'
Default rule: +
Classification Accuracy : 84.49% +/- 1.04%
Rules Number : 7.6 +/- 0.16
Model size: 10.6 +/- 0.4

AGI-AntMiner

IF A9 = 't' AND A10 = 't' THEN '+'
IF A6 = 'i' AND A8 = '\(-inf-1.02]\' THEN '-'
IF A6 = 'w' AND A7 = 'v' THEN '+'
IF A3 = '\(-inf-4.2075]\' AND A13 = 's' THEN '-'
IF A9 = 'f' THEN '-'
IF A14 = '\(-inf-105]\' THEN '+'
IF A4 = 'u' AND A7 = 'h' THEN '+'
IF A6 = 'aa' THEN '-'
IF A1 = 'b' AND A4 = 'y' AND A13 = 'g' AND A15 = '\(-inf-492]\' THEN '-'
Default rule: -

Classification Accuracy : 87.1% +/- 0.93%
Rules Number : 10.6 +/- 0.34
Model size: 12.7 +/- 0.72

ACO/ PSO2

IF a1 = b a3 = '\'(-inf-4.2075]\" a7 = v a8 = '\'(-inf-1.02]\" a9 = f a11 = '\'(-inf-0.5]\" a14 = '\'(105-inf)\" THEN -
IF a9 = t a11 = '\'(2.5-inf)\" a15 = '\'(492-inf)\" THEN +
IF a9 = f a13 = g a14 = '\'(-inf-105]\" a15 = '\'(-inf-492]\" THEN -
IF a9 = t a10 = t a12 = f a14 = '\'(-inf-105]\" THEN +
IF a1 = a a3 = '\'(-inf-4.2075]\" a5 = g a7 = v a13 = g a14 = '\'(105-inf)\" a15 = '\'(-inf-492]\" THEN -
IF a3 = '\'(4.2075-inf)\" a5 = g a9 = t a13 = g THEN +
IF a7 = v a9 = t a10 = t THEN +
IF a11 = '\'(-inf-0.5]\" a14 = '\'(105-inf)\" THEN -
IF a13 = g a14 = '\'(105-inf)\" a15 = '\'(-inf-492]\" THEN -
IF a11 = '\'(-inf-0.5]\" THEN +
IF a5 = p a8 = '\'(-inf-1.02]\" a9 = f a13 = g THEN -
IF a9 = f a11 = '\'(0.5-2.5]\" a13 = g a15 = '\'(-inf-492]\" THEN -
IF a2 = '\'(-inf-38.96]\" a9 = f a13 = g THEN -
IF a8 = '\'(-inf-1.02]\" a10 = f a13 = s THEN -
IF a9 = t a14 = '\'(105-inf)\" THEN +
IF a11 = '\'(-inf-0.5]\" a13 = g a15 = '\'(-inf-492]\" THEN -
IF a10 = f a12 = t THEN +
IF a14 = '\'(-inf-105]\" THEN -
Default rule: -
Classification Accuracy : 84.69 % +/- 4.39 Rules Number: 20.1 +/- 1.37 Model size: 70.6 +/- 7.6

ACO/ SA

IF A4 = 'u' AND A8 = '\(1.02-inf)\' AND A13 = 'g' AND A14 = '\(105-inf)\' THEN '+'
IF A4 = 'u' AND A8 = '\(1.02-inf)\' AND A10 = 't' THEN '+'
IF A1 = 'b' AND A4 = 'u' AND A7 = 'v' AND A13 = 'g' THEN '+'
IF A1 = 'b' AND A7 = 'v' AND A8 = '\(1.02-inf)\' AND A13 = 's' THEN '+'
IF A7 = 'v' AND A8 = '\(-inf-1.02]\' AND A13 = 'g' THEN '+'
IF A4 = 'y' AND A10 = 't' AND A12 = 't' THEN '+'
IF A3 = '\(-inf-4.2075]\' AND A6 = 'm' AND A13 = 's' THEN '-'
IF A15 = '\(492-inf)\' THEN '+'
IF A9 = 'f' THEN '-'
IF A7 = 'bb' THEN '-'
IF A4 = 'y' AND A7 = 'h' THEN '+'
IF A4 = 'u' AND A7 = 'h' AND A11 = '\(-inf-0.5]\' THEN '+'
IF A1 = 'b' AND A6 = 'c' AND A13 = 'g' THEN '-'
IF A2 = '\(38.96-inf)\' AND A7 = 'v' THEN '+'
IF A6 = 'aa' AND A13 = 'g' THEN '-'
IF A14 = '\(-inf-105]\' THEN '+'
IF A6 = 'w' THEN '+'
IF A12 = 't' THEN '-'
Default rule: -
Classification Accuracy: 85.80 % +/-2.58 Rules Number: 20.40 +/- 2.33 Model size: 53.50 +/- 8.88

Figure 5.11a. An example of the classification model obtained by AGI-AntMiner and the other classification algorithms

CAnt-MinerPB

IF A6 = w AND A8 = \ (1.02-inf)\ AND A13 = g AND A3 = \ (4.2075-inf)\ THEN +
IF A6 = q AND A14 = \ (105-inf)\ AND A5 = g THEN +
IF A9 = f THEN -
IF A6 = d THEN -
IF A15 = \ (492-inf)\ THEN +
IF A14 = \ (-inf-105]\ THEN +
IF A1 = a THEN -
IF A15 = \ (-inf-492]\ AND A12 = f THEN +
IF A14 = \ (-inf-105]\ AND A4 = y AND A13 = g THEN -
IF A11 = \ (2.5-inf)\ THEN +
IF A10 = t THEN +
IF A6 = w THEN +
IF A8 = \ (-inf-1.02]\ THEN -
IF A13 = s THEN -
IDefault rule: -
Classification Accuracy : 84.348 % +/- 1.009 Rules Number: 14.1 +/- 1.91 Model size: 20.9 +/- 6.96

PART

IF A6 = c AND A2 = '(-inf-38.96]' AND A3 = '(-inf-4.2075]' AND A4 = u: THEN=+
IF A3 = '(4.2075-inf)' AND A8 = '(1.02-inf)' AND A4 = u: THEN=+
IF A9 = f AND A13 = g AND A6 = ff: THEN= -
IF A10 = t AND A15 = '(492-inf)': THEN=+
IF A15 = '(492-inf)': THEN=+
IF A10 = t AND A6 = x: THEN=+
IF A13 = s: -
IF A6 = aa AND A1 = b: THEN=-
IF A7 = v AND A8 = '(-inf-1.02]': THEN=+
IF A11 = '(0.5-2.5]': THEN=+
IF A6 = c AND A3 = '(4.2075-inf)': THEN=+
IF A6 = c:THEN= -
A6 = x: THEN=-
IF A9 = f AND A13 = g AND A6 = i: THEN=-
IF A10 = t AND A14 = '(-inf-105]': THEN=+
IF A10 = t AND A6 = cc: THEN=+
IF A7 = h AND A14 = '(-inf-105]': THEN=+
IF A9 = f: -
IF A7 = v AND A6 = c: THEN=-
IF A6 = q: THEN=-
IF A6 = ff: THEN=-
IF A6 = w: THEN=-
IF A6 = k: THEN=-
IF A1 = b: THEN=+
Default rule: -
Classification Accuracy : 86.37 % +/- 0.19 Rules Number: 22.1 ± 3.24 Model size: 41.9 ± 8.79

FURIA

IF(A9 = t) and (A10 = t) => THEN=+
IF(A9 = t) and (A14 = '(-inf-105]') => THEN =+
IF(A9 = t) and (A7 = h) => THEN =+
IF(A9 = t) and (A6 = w) => THEN =+
IF(A9 = f) => THEN =-
IF(A10 = f) and (A14 = '(105-inf)') and (A15 = '(-inf-492]') => THEN =-
Default rule: -
Classification Accuracy : 86.52 % +/-0.36 Rules Number: 6 Model size: 12

Figure 5.11b. An example of the classification model obtained by AGI-AntMiner and the other classification algorithms

Table 5.16 and Figures 5.12 and 5.13 show the results of the nonparametric Friedman test with Holm's post-hoc test in the second scenario. This scenario computes the average classification accuracy rank, average number of discovered rules rank, and average model size rank of the statistical results across the 16 datasets, as reported in Table 5.16. Figure 5.12 displays the results of the average classification accuracy rank versus the average number of rules rank. Figure 5.13 presents the results of the average classification accuracy rank versus the average results of model size rank. In all cases, the lowest rank indicates a good algorithm performance.

Table 5.16

Test results of AGI-AntMiner and other classifiers based on average performance rank on all datasets

	Ant-Miner	**ACO/ PSO 2**	**ACO/SA**	**AGI-AntMiner**	**CAnt-MinerPB**	**PART**	**FURIA**
Accuracy	6.312	5.5	3.062	**2.312**	4.625	3.5	2.687
Rule	**1.718**	4.156	5.781	2.875	3.812	5.375	4.281
Terms	**1.625**	4.312	5.937	2.562	3	5.375	5.187

Figure 5.12 shows that the AGI-AntMiner classifier dominates the other six classifiers. The AGI-AntMiner only performs slightly worse than Ant-Miner in the average number of discovered rules but it is better in the classification accuracy. Therefore, the AGI-AntMiner outperforms the other classifiers.

Figure 5.13 proves that the results obtained by the AGI-AntMiner dominate those by the other classifiers considering the classification accuracy and model size ranks. The

AGI-AntMiner performs only slightly worse than the Ant-Miner in terms of model size, but it is better than the Ant-Miner and other classifiers in terms of classification accuracy. Therefore, the AGI-AntMiner is the best classifier that balances the classification accuracy and model size.

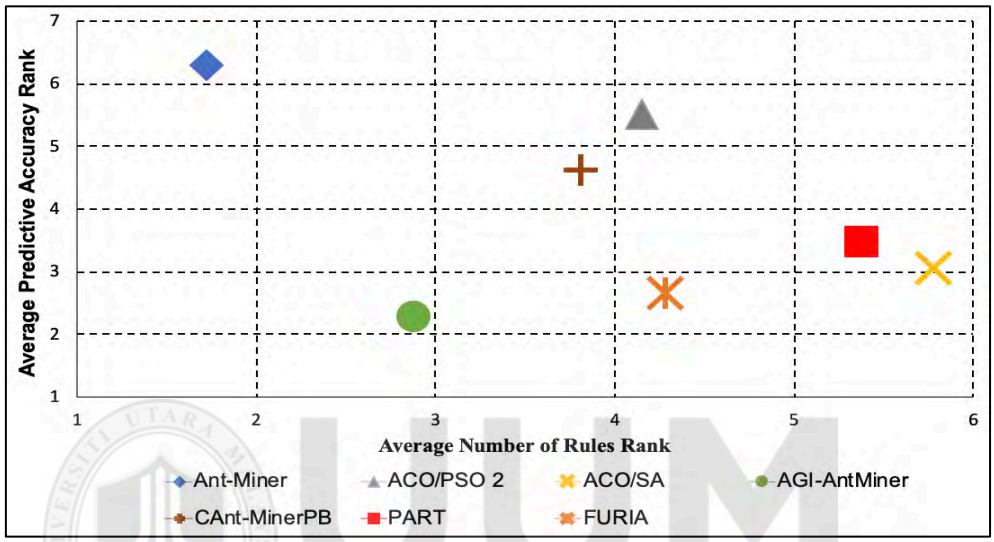

Figure 5.12. Results of AGI-AntMiner classifier on the average classification accuracy rank versus the average number of discovered rules rank

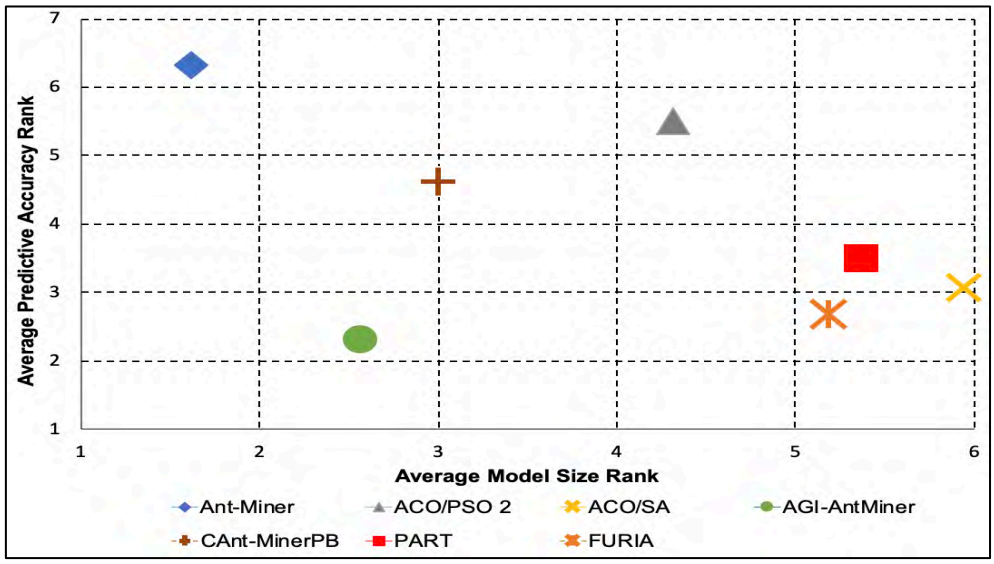

Figure 5.13. Results of AGI-AntMiner classifier on the average classification accuracy rank versus the average model size rank

Considering the classification accuracy ranking and comprehensibility (i.e., number of discovered rules and model size) ranking, the AGI-AntMiner classifier achieves the best performance among all included classifiers. However, the AGI-AntMiner classifier needs additional rules and terms per rule to outperform the other rules-based classifiers. Thus, the balance between accuracy and comprehensibility is clearly present in our results: the most accurate results are obtained by FURIA, ACO/SA, and PART models, whereas the Ant-Miner classifier needs additional rules and terms per rule to outperform them. Therefore, the improved results of the AGI-AntMiner can be attributed to the ACO algorithm, which has a proven record of providing high-performing solutions, together with our proposed modifications that aim to find the optimal fitting for each dataset by minimising the number rules and terms per rule and maximising classification accuracy.

5.7 Chapter Summary

A balance between classification accuracy and comprehensibility in rules-based classification was achieved by utilising the properties of improved pre-pruning and post-pruning techniques and the hybridisation with ILS to obtain the optimal results. Specifically, we investigated three rules-based classification algorithms, namely A-AntMiner, GA-AntMiner, and ILS-AntMiner. The experimental results show that the A-AntMiner outperforms the other classifiers that used different rule pruning techniques in classification performance and comprehensibility. The reason is that the A-AntMiner can find the appropriate pruning criteria for each dataset by considering the feedback (local and global) in the learning process. In addition, the GA-AntMiner balances the classification accuracy and model size due to the

enhancement on the post-pruning strategy by using the concept of the GA algorithm. Furthermore, ILS-AntMiner can effectively search the training space on the basis of the multiple neighbourhoods structure to escape from local optima for improving classification rules. Finally, the benchmarking experiments show that the AGI-AntMiner algorithm, which integrates the abovementioned three modifications techniques, outperforms several state-of-the-art rules-based classification algorithms. This algorithm has a remarkable capability to reduce the size of terms while yielding significant classification accuracy. Hence, the AGI-AntMiner algorithm exhibits the overall best performance among the compared rules-based algorithms. The next chapter will summarise the key findings and provide promising research directions.

CHAPTER SIX
CONCLUSIONS, LIMITATION AND FUTURE WORK

6.1 Introduction

The main goal of this research is to develop an adaptive ant colony optimization algorithm for rule-based classification. The development consists of three main stages i.e. new pre-pruning technique, new post-pruning technique and hybridising with ILS. The first and second stages are improved in such a way that the characteristics of pre-pruning and post-pruning is combined to produce a new hybrid rule pruning technique that features rule accuracy and comprehensibility. This hybrid method involves a parameter control mechanism to select an appropriate pre-pruning threshold value (i.e., term strength or importance), wherein the threshold value can be automatically adapted. The search strategy of post-pruning has been improved by using the characteristic of GA population-based optimisation. Finally, the proposed classifier has been improved by incorporating ILS component to the best rule discovered in training stage and successively modifying it with a better rule in its neighbourhood. Section 6.2 highlights the main contributions of this study, Section 6.3 shows the main limitation of this research, and Section 6.4 provides some suggestions for future research directions.

6.2 Contributions

The performance of ACO for rules-based classification algorithms can be improved by enhancing the pruning procedure and combining it with other local search algorithms. In this research, parameter adaptation strategy and GA are found to be

very useful to enhance the pruning procedure of the ACO for rules-based classification algorithms. In addition, the hybridisation with ILS can improve the performance of the classifier. This study has two main contributions: knowledge and practical contributions.

6.2.1 Knowledge Contribution

i. A new classifier, which is called the A–AntMiner based on the pre-pruning technique is developed. This classifier prunes the irrelative terms from the rule during the construction stage by using a new pruning parameter, which is called the importance rate (ζ). This parameter is adapted using the ACO concept. The classifier can collect feedback from the classification search space. The feedback mechanisms are then used to evaluate the impact of the current parameter value on the classification performance and perform suitable reactions for adjustment. The experimental results show that the A–AntMiner, which uses the pre-pruning technique based on the ζ parameter and online adaptation algorithm, has a better performance than other pruning techniques that use the Ant-Miner classification algorithm.

ii. The GA–AntMiner, which is based on a new post-pruning technique, is introduced. This classifier is used to prune the irrelative terms after the rule is constructed by the ant. The classifier used the GA principles for finding the optimal pruning rule. The results show that the Genetic-based post-pruning technique used in GA–AntMiner classifier has a better performance than other pruning techniques that use in the literature of the Ant-Miner classification algorithms.

iii. ILS–AntMiner, which is the hybridisation between the Ant-Miner and ILS algorithms, is proposed. This hybrid algorithm can refine the solution produced by the Ant-Miner algorithm by using the ILS capabilities (i.e., perturbation and local search). The ILS modifies the current rule by using a perturbation procedure to generate a new, promising starting classification rule for the next local search application. In the local search procedure, the local improvement for the current rule is based on a simple modification to enhance the classification rule. The experimental results show that ILS–AntMiner has better a performance than other local search hybridisation techniques.

iv. The three contributions mentioned above are merged into the Ant-Miner to form the AGI-AntMiner algorithm for rules-based classification. Experimental results show that the contributed components work harmoniously and improve the quality of solutions. The final algorithmic approach is compared with five ant-mining classification algorithms and three other rules-based classification algorithms.

In summary, rule pruning is a framework used to avoid overfitting. The pruning can detect the significant terms in the rule and prune the irrelative terms that provide minimal quality in classifying the instances. Therefore, the influence of rule pruning in Ant-Miner classification algorithms plays an important role in improving the classifier performance. Thus, the characteristics of pre- and post-pruning that are combined to produce a new hybrid rule pruning technique in this study improve rule accuracy and comprehensibility. In addition, ILS can improve the discovered rule by

using the ILS components (i.e., perturbation and local search). The experimental results show that ILS can reduce the local optima problem of Ant-Miner algorithms.

6.2.2 Practical Contribution

The proposed AGI-AntMiner classification algorithm can be used in real applications and allows experts to review the result due to the simplicity of the classification model "if-then condition". In this way, the AGI-Antminer algorithm provides a deep insight into the relationship between the key factors that lead to predicting the class labels. For example, in disease diagnoses, several variables affect illnesses. Therefore, finding the specific variables is very important and can assist in the control of disease. This classifier algorithm will be the future direction for different domains with an overall goal to extract information from the data. These domains may include, text mining, DNA sequence classification, business, economics classification, library classification, education systems, credit scoring, medical diagnosis, biological classification, health care systems and manufacturing engineering.

6.3 Limitation

The AGI-AntMiner algorithm for rules-based classification can be improved in terms of its capability to cope with continuous attributes. The proposed classifier can only cope with discrete attributes and, thus, needs a pre-processing step to deal with continuous attributes.

6.4 Future Work

The results offer various directions for future research in several areas as follows.

i. The UCI secondary datasets are used to evaluate the performance of our classifiers. These datasets are varied in instances, attributes and class label numbers. Therefore, they include categorical and continuous attribute types. Although the effect of overfitting is imminent using the secondary datasets, related improvements in popular datasets have been implemented to produce an explicit indication of algorithmic superiority. Correspondingly, real-world application in various domains, such as DNA sequence classification, medical diagnosis, credit scoring and text mining, is required. These real-life applications can provide extensive knowledge in the behaviour and performance of the Ant-Miner algorithm.

ii. Other adaptation techniques can automatically control the ζ parameter during the learning process. Such adaptations can draw inspiration from other stochastic local search algorithms that automatically optimise parameters effectively. Other swarm-intelligent optimisation algorithms, such as PSO, ABC, BA and FA, can be utilised to optimise the ζ parameter online.

iii. Other stochastic local search algorithms, such as randomised iterative improvement, iterated greedy and evolutionary algorithms, can be hybridised with Ant-Miner algorithms.

iv. The number of predicted classes should be considered in developing an Ant-Miner classifier for large databases. The AGI-AntMiner algorithm for rules-based classification can deal with the classical single-classification task. The proposed classifier can be improved for multi-label classification.

v. The values of several parameters (i.e., mutation rate, crossover rate and perturbation length) should be controlled when GA and ILS are used to find the best classification rule. This task is important in the rules classification technique for adjusting the parameter values for the dataset in designing a classification model.

REFERENCE:

Aarts, E. H. L., Korst, J. H. M., & Laarhoven, P. J. M. Van. (1997). Simulated annealing. *Local Search in Combinatorial Optimization*, *27*(3), 797–802.

Abbas, A. R., Juan, L., & Mahdi, S. O. (2007). Improved variable precision rough set model and its application to distance learning. *Proceedings - 2007 International Conference on Computational Intelligence and Security, CIS 2007*, 191–195. https://doi.org/10.1109/CIS.2007.156

Abdel-Basset, M., Mohamed, M., Smarandache, F., & Chang, V. (2018). Neutrosophic association rule mining algorithm for big data analysis. *Symmetry*, *10*(4), 1–19. https://doi.org/10.3390/sym10040106

Abdoos, A. A., Mianaei, P. K., & Ghadikolaei, M. R. (2016). Combined VMD-SVM based feature selection method for classification of power quality events. *Applied Soft Computing Journal*, *38*, 637–646. https://doi.org/10.1016/j.asoc.2015.10.038

Abuhamdah, A. F. (2018). Adaptive elitist-ant system for medical clustering problem. *Journal of King Saud University - Computer and Information Sciences*, 0–8. https://doi.org/10.1016/j.jksuci.2018.08.007

Agravat, D., Vaishnav, U., & Swadas, P. B. (2010). Modified Ant Miner for intrusion detection. In *ICMLC 2010 - The 2nd International Conference on Machine Learning and Computing* (pp. 228–232).

Al-Abadi, A. M. (2017). A novel geographical information system-based Ant Miner algorithm model for delineating groundwater flowing artesian well boundary: a case study from Iraqi southern and western deserts. *Environmental Earth*

Sciences, *76*(15). https://doi.org/10.1007/s12665-017-6876-2

Al-behadili, H. N. K., Ku-Mahamud, K. R., & Sagban, R. (2018). Rule pruning techniques in the ant-miner classification algorithm and its variants: A review. In *2018 IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE)* (pp. 78–84). Batu Feringghi, Penang, Malaysia: IEEE.

Ali, S., & Smith, K. A. (2006). On learning algorithm selection for classification. *Applied Soft Computing Journal*, *6*(2), 119–138. https://doi.org/10.1016/j.asoc.2004.12.002

AlMana, A. M., & Aksoy, M. S. (2014). An Overview of Inductive Learning Algorithms. *International Journal of Computer Applications*, *88*(4), 20–28.

Almana, A. M., Aksoy, M. S., & Alzahrani, R. (2014). A Survey On Data Mining Techniques In Customer Churn Analysis For Telecom Industry. *Journal of Engineering Research and Applications*, *4*(5), 165–171.

Alwan, H. B. (2013). *Hybrid Aco and Svm Algorithm for Pattern Classification Doctor of Philosophy Permission to Use*. UUM.

Amato, F., López, A., Peña-Méndez, E. M., Vaňhara, P., Hampl, A., & Havel, J. (2013). Artificial neural networks in medical diagnosis. *Journal of Applied Biomedicine*, *11*(2), 47–58.

Amir, C., Badr, A., & Farag, I. (2007). A Fuzzy Logic Controller for Ant Algorithms. *Computing and Information Systems*, *2*(11), 26–34.

An, A. (2003). Learning Classification Rules from Data. *Computers and Mathematics with Applications*, *45*(4–5), 737–748.

Arif-Ul-Islam, & Ripon, S. H. (2019). Rule Induction and Prediction of Chronic Kidney Disease Using Boosting Classifiers, Ant-Miner and J48 Decision Tree.

In *International Conference on Electrical, Computer and Communication Engineering (ECCE)* (pp. 1–6). IEEE. https://doi.org/10.1109/ECACE.2019.8679388

Asadi, S., & Shahrabi, J. (2016). ACORI: A novel ACO algorithm for rule induction. *Knowledge-Based Systems*, *97*(March), 175–187. https://doi.org/10.1016/j.knosys.2016.01.005

Au, W. H., Chan, C. C., & Yao, X. (2003). A novel evolutionary data mining algorithm with applications to churn prediction. *IEEE Transactions on Evolutionary Computation*, *7*(6), 532–544. https://doi.org/10.1109/TEVC.2003.819264

Ayub, U., Ikram, A., & Shahzad, W. (2019). AM clr — An Improved Ant-Miner to Extract Comprehensible and Diverse Classification Rules. In *GECCO '19 Proceedings of the Genetic and Evolutionary Computation Conference* (pp. 4–12).

Azar, A. T., & El-Metwally, S. M. (2013). Decision tree classifiers for automated medical diagnosis. *Neural Computing and Applications*, *23*(7–8), 2387–2403.

Azar, A. T., & El-Said, S. A. (2014). Performance analysis of support vector machines classifiers in breast cancer mammography recognition. *Neural Computing and Applications*, *24*(5), 1163–1177.

Baig, A. R., & Shahzad, W. (2012). A correlation-based ant miner for classification rule discovery. *Neural Computing and Applications*, *21*(2), 219–235.

Battiti, R., & Brunato, M. (2014). *The LION way. Machine Learning plus Intelligent Optimization*.

Beheshti, Z., & Shamsuddin, S. M. H. (2013). A review of population-based meta-

heuristic algorithm. *Int. J. Adv. Soft Comput*, *5*(1). https://doi.org/2074-8523

Blum, C., & Roli, A. (2001). Metaheuristics in combinatorial optimization: Overview and Conceptual Comparison. *ACM Computing Surveys*, *140*(1), 189–213. https://doi.org/10.1007/s10479-005-3971-7

Bramer, M. (2002). An information-theoretic approach to the pre-pruning of classification rules. *Intelligent Information Processing*, 201–212.

Brest, J., Greiner, S., Boskovic, B., Mernik, M., & Zumer, V. (2006). Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE Transactions on Evolutionary Computation*, *10*(6), 646–657.

Cepukenas, J., Lin, C., & Sleeman, D. (2015). Applying Rule Extraction & Rule Refinement techniques to (Blackbox) Classifiers. In *8th International Conference on Knowledge Capture. ACM*. https://doi.org/10.1145/2815833.2816950

Chan, A., & Freitas, A. (2006a). A new classification-rule pruning procedure for an Ant Colony Algorithm. In *In International Conference on Artificial Evolution (Evolution Artificielle)* (Vol. 3871 LNCS, pp. 25–36). Berlin, Heidelberg: Springer. https://doi.org/10.1007/11740698_3

Chan, A., & Freitas, A. (2006b). A new ant colony algorithm for multi-label classification with applications in bioinfomatics. In *the 8th annual conference on Genetic and evolutionary computation* (pp. 27–34). https://doi.org/10.1145/1143997.1144002

Chang, R. S., Chang, J. S., & Lin, P. S. (2009). An ant algorithm for balanced job scheduling in grids. *Future Generation Computer Systems*, *25*(1), 20–27.

https://doi.org/10.1016/j.future.2008.06.004

Chennupati, G. (2014). eAnt-Miner: An Ensemble Ant-Miner to Improve the ACO Classification. *ArXiv Preprint ArXiv:1409.2710*.

Chorbev, I., Mihajlov, D., & Jolevski, I. (2009). Web based medical expert system with a self training heuristic rule induction algorithm. In *Proceedings - 2009 1st International Conference on Advances in Databases, Knowledge, and Data Applications, DBKDA 2009* (pp. 143–148). IEEE. https://doi.org/10.1109/DBKDA.2009.21

Cohen, W. (1995). Fast effective rule induction. *In Machine Learning Proceedings*, *2435*, 115–123.

Cover, T. M., & Thomas, J. (2006). *Introduction and preview Elements of information theory*. *John Wiley & Sons*. https://doi.org/10.1002/0471200611.ch1

Danenas, P., & Garsva, G. (2015). Selection of Support Vector Machines based classifiers for credit risk domain. *Expert Systems with Applications*, *42*(6), 3194–3204.

Datta, R. P., & Saha, S. (2016). Applying rule-based classification techniques to medical databases : an empirical study. *International Journal of Business Intelligence and Systems Engineering*, *1*(1), 32–48.

De La Cruz, J. J., Paternina-Arboleda, C. D., Cantillo, V., & Montoya-Torres, J. R. (2013). A two-pheromone trail ant colony system - Tabu search approach for the heterogeneous vehicle routing problem with time windows and multiple products. *Journal of Heuristics*, *19*(2), 233–252.

Demsar, J. (2006). Statistical Comparisons of Classifiers over Multiple Data Sets.

Journal of Machine Learning Research, *7*, 1–30.

Ding, C., & Peng, H. (2003). Minimum redundancy feature selection from microarray gene expression data. *Bioinformatics Conference, 2003. CSB 2003. Proceedings of the 2003 IEEE*, *3*(2), 523–528. https://doi.org/10.1109/CSB.2003.1227396

Dorigo, M., Birattari, M., & Stützle, T. (2006). *Ant Colony Optimization: Artificial Ants as a Computational Intelligence Technique*. *IRIDIA, Institut de Recherches Interdisciplinaires et de Developpements en Intelligence Artificiell* (Vol. 1). Bruxelles, Belgium. https://doi.org/http://dx.doi.org/10.1109/CI-M.2006.248054

Dorigo, M., & Colorni, A. (1996). The Ant System: Optimization by a Colony of Cooperating Agents. *IEEE Transactions on Systems, Man, and Cybernetics–Part B*, *26*(1), 1–13.

Dorigo, M., & Gambardella, L. M. (1997). Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, *1*(1), 53–66.

Dorigo, M., & Stutzle, T. (2003). The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances. *Handbook of Metaheuristics SE - International Series in Operations Research & Management Science*, *57*, 250–285. https://doi.org/doi: 10.1007/0-306-48056-5_9

Dorigo, M., & Stutzle, T. (2010). *Ant Colony Optimization: Overview and Recent Advances*. *Handbook of Metaheuristics* (Vol. 146).

Dorigo, M., & Stützle, T. (2004). *Ant Colony Optimization*. Cambridge, MA, USA: MIT Press.

Dorigo, M., & Stützle, T. (2010). Ant Colony Optimization: Overview and Recent

Advances. In *Handbook of Metaheuristics* (Vol. 146, pp. 227–264). Springer US.

Drias, Y., Kechid, S., & Pasi, G. (2015). A Novel Framework for Medical Web Information Foraging Using Hybrid ACO and Tabu Search. *Journal of Medical Systems*, *40*(1), 5.

Dua, D., & Karra, T. (2017). UCI Machine Learning Repository. Retrieved from http://archive.ics.uci.edu/ml

Durgadevi, M., & Kalpana, R. (2017). Medical distress prediction based on Classification Rule Discovery using ant-miner algorithm. *Proceedings of 2017 11th International Conference on Intelligent Systems and Control, ISCO 2017*, 88–92. https://doi.org/10.1109/ISCO.2017.7855959

Durgadevi, M., & Kalpana, R. (2018). Performance Analysis of Classification Approaches for the Prediction of Type II Diabetes. *2017 9th International Conference on Advanced Computing, ICoAC 2017*, 339–344. https://doi.org/10.1109/ICoAC.2017.8441197

Eiben, A., Michalewicz, Z., Schoenauer, M., & Smith, J. (2007). Parameter Control in Evolutionary Algorithms. *Parameter Setting in Evolutionary Algorithms, 19-46.*, Parameter setting in evolutionary algorithms, 19-4.

Elgibreen, H., & Aksoy, M. S. (2013). Rules - TL: A simple and improved rules algorithm for incomplete and large data. *Journal of Theoretical and Applied Information Technology*, *47*(1), 28–40.

Eswaramurthy, V. P., & Tamilarasi, A. (2009). Hybridizing tabu search with ant colony optimization for solving job shop scheduling problems. *International Journal of Advanced Manufacturing Technology*, *40*(9–10), 1004–1015.

https://doi.org/10.1007/s00170-008-1404-x

Ezzat, A., Abdelbar, A. M., & Wunsch, D. C. (2015). An extended EigenAnt colony system applied to the sequential ordering problem. *IEEE SSCI 2014 - 2014 IEEE Symposium Series on Computational Intelligence - SIS 2014: 2014 IEEE Symposium on Swarm Intelligence, Proceedings*, 276–282.

Fakhar, B. (2014). A Novel Method for Extracting Classification Rules based on Ant-Miner. *Journal of Mathematics and Computer Science*, *8*, 377–386.

Falaghi, H., & Haghifam, M. R. (2007). ACO based algorithm for distributed generation sources allocation and sizing in distribution systems. *2007 IEEE Lausanne POWERTECH, Proceedings*, 555–560. https://doi.org/10.1109/PCT.2007.4538377

Fan, W., & Bifet, A. (2013). Mining Big Data : Current Status , and Forecast to the Future. *ACM SIGKDD Explorations Newsletter*, *14*(2), 1–5.

Farid, D. M., Zhang, L., Rahman, C. M., Hossain, M. A., & Strachan, R. (2014). Hybrid decision tree and naïve Bayes classifiers for multi-class classification tasks. *Expert Systems with Applications*, *41*(4 PART 2), 1937–1946.

Fauzi bin Othman, M., & Moh Shan Yau, T. (2007). Comparison of Different Classification Techniques Using WEKA for Breast Cancer. In *rd Kuala Lumpur International Conference on Biomedical Engineering* (Vol. 06, pp. 520–523). Berlin, Heidelberg: Springer.

Fox, B., Xiang, W., & Lee, H. P. (2007). Industrial applications of the ant colony optimization algorithm. *International Journal of Advanced Manufacturing Technology*, *31*(7–8), 805–814. https://doi.org/10.1007/s00170-005-0254-z

Frank, E., & Witten, I. (1998). Generating accurate rule sets without global

optimization.

Franzin, A., & Stutzle, T. (2018). *Revisiting Simulated Annealing: a Component-Based Analysis*. Bruxelles, Belgium.

Freitas, A. (2002). Data Mining and Knowledge Discovery with Evolutionary Algorithms. *Advances in Evolutionary Computation*, *105*, 819–845. https://doi.org/10.1007/s13398-014-0173-7.2

Freitas, A. (2014). Comprehensible Classification Models – a position paper. *ACM SIGKDD Explorations Newsletter*, *15*(1), 1–10. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.124.6470&rep=rep1&type=pdf

Freitas, A., & Parpinelli, R. S. (2008). Ant ColonyAlgorithms for Data Classification. *Encyclopedia of Information Science and Technology*, 154–159. Retrieved from http://www.springerlink.com/index/J07734581405Q133.pdf

Fürnkranz, J. (1997). Pruning Algorithms for Rule Learning. *Machine Learning*, *27*(2), 139–172.

Fürnkranz, J., Gamberger, D., & Lavrač, N. (2012). Foundations of Rule Learning. *Cognitive Technologies*, *6*(2003), 19–55.

Gaertner, D., & Clark, K. (2005). On Optimal Parameters for Ant Colony Optimization algorithms TSP classifications. *IC-AI*, 83–89. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.96.6751&rep=rep1&type=pdf

Gamarra, C., Guerrero, J. M., & Montero, E. (2016). A knowledge discovery in databases approach for industrial microgrid planning. *Renewable and Sustainable Energy Reviews*, *60*, 615–630.

Gambardella, L. M. (2015). *Coupling Ant Colony System with Local Search*. UNIVERSIT´E LIBRE DE BRUXELLES.

Gambardella, L. M., Montemanni, R., & Weyland, D. (2012). Coupling ant colony systems with strong local searches. *European Journal of Operational Research*, *220*(3), 831–843. https://doi.org/10.1016/j.ejor.2012.02.038

Gavrilovski, A., Jimenez, H., Mavris, D. N., Rao, A. H., Shin, S., Hwang, I., & Marais, K. (2016). Challenges and Opportunities in Flight Data Mining: A Review of the State of the Art. In *AIAA Infotech @ Aerospace* (pp. 1–18). https://doi.org/10.2514/6.2016-0923

Geem, Z. W., Kim, J. H., & Loganathan, G. V. (2001). A new heuristic optimization algorithm: Harmony search. *Simulation*, *76*(2), 60–68. https://doi.org/10.1177/003754970107600201

Ghahramani, Z. (2015). Probabilistic machine learning and artificial intelligence. *Nature*, *521*(7553), 452–459.

Ghosh, A., & Nath, B. (2004). Multi-objective rule mining using genetic algorithms. *Information Sciences*, *163*(1–3), 123–133. https://doi.org/10.1016/j.ins.2003.03.021

Gillis, J. M., & Morsi, W. G. (2016). Non-Intrusive Load Monitoring Using Semi-Supervised Machine Learning and Wavelet Design. *IEEE Transactions on Smart Grid*, 1–8. https://doi.org/10.1109/TSG.2016.2532885

Glover, F. (1986). Paths for Integer Programming. *Computers and Operations Research*, *13*(5), 533–549. https://doi.org/http://dx.doi.org/10.1016/0305-0548(86)90048-1

Guan, J., & Lin, G. (2016). Hybridizing variable neighborhood search with ant

colony optimization for solving the single row facility layout problem. *European Journal of Operational Research*, *248*(3), 899–909.

Han, J. ., & Kamber, M. (2006a). *Data Mining: Concepts and Techniques*. *Elsevier* (Vol. 12).

Han, J., & Kamber, M. (2006b). *Data Mining: Concepts and Techniques*. *Elsevier* (Vol. 12). https://doi.org/10.1007/978-3-642-19721-5

Han, J., & Kamber, M. (2011). *Data Mining Concept and Techniques*. (D. M. C. and T. By, Ed.). United States of America: Morgan Kaufmann.

Hao, Z., Cai, R. C., & Huang, H. (2006). An adaptive parameter control strategy for ACO. *Proceedings of the 2006 International Conference on Machine Learning and Cybernetics*, (August), 203–206.

Harvey, J. P. (2009). *GPU acceleration of object classification algorithms using NVIDIA CUDA*.

He, J., Long, D., & Chen, C. (2007). An improved ant-based classifier for intrusion detection. *Third International Conference on Natural Computation, ICNC 2007*, *4*(60273062), 819–823.

Holden, N., & Freitas, A. A. (2008). A Hybrid PSO/ACO Algorithm for Discovering Classification Rules in Data Mining. *Journal of Artificial Evolution and Applications*, *2008*, 1–11. https://doi.org/10.1155/2008/316145

Holzinger, A., Plass, M., Holzinger, K., Crisan, G. C., Pintea, C. M., & Palade, V. (2017). A glass-box interactive machine learning approach for solving NP-hard problems with the human-in-the-loop. *ArXiv Preprint ArXiv:1708.01104*, 1–26. Retrieved from http://arxiv.org/abs/1708.01104

Holzinger, K., Palade, V., Rabadan, R., & Holzinger, A. (2014). Darwin or

Lamarck? Future Challenges in Evolutionary Algorithms for Knowledge Discoveryand Data Mining. In *Interactive Knowledge Discovery and Data Mining in Biomedical Informatics* (Vol. 8401). Verlag Berlin Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-43968-5

Hoos, H. H., & Stutzle, T. (2015). Stochastic Local Search Algorithms: An Overview. In W. (Eds. . Kacprzyk, Janusz, Pedrycz (Ed.), *Springer Handbook of Computational Intelligence* (pp. 1085–1105). Springer. https://doi.org/10.1007/978-3-662-43505-2_54

Hota, S., Satapathy, P., & Jagadev, A. K. (2015). Modified Ant Colony Optimization Algorithm (MAnt-Miner) for Classification Rule Mining. In *Intelligent Computing, Communication and Devices.* (Vol. 1, pp. 267–275). New Delhi, India: Springer. https://doi.org/10.1007/978-81-322-2012-1

Hssina, B., Merbouha, A., Ezzikouri, H., & Erritali, M. (2014). A comparative study of decision tree ID3 and C4.5. *International Journal of Advanced Computer Science and Applications*, (2), 13–19.

Huang, H., Yang, X., Hao, Z., & Cai, R. (2006). A novel AGO algorithm with adaptive parameter. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *4115 LNBI*(December 2013), 12–21. https://doi.org/10.1007/11816102

Hühn, J., & Hüllermeier, E. (2009). FURIA: An algorithm for unordered fuzzy rule induction. *Data Mining and Knowledge Discovery*, *19*(3), 293–319. https://doi.org/10.1007/s10618-009-0131-8

Irani, K., & Fayyad, U. (1993). Multi-lnterval Discretization of Continuous-Valued Attributes for Classification learning. In *IJCAI–93* (pp. 1022–1027).

https://doi.org/10.1109/TKDE.2011.181

Jacobson, L., & Kanber, B. (2015). *Genetic Algorithms in Java Basics*. New York, USA: APress.

Jaganathan, P., Thangavel, K., Pethalakshmi, A., & Karnan, M. (2007). Classification rule discovery with ant colony optimization and improved quick reduct algorithm. *IAENG International Journal of Computer Science*, (February), 286–291. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-84888273942&partnerID=tZOtx3y1

Jain, A., Mao, J., & Mohiuddin, K. M. (1996). Artificial neural networks: A tutorial. *Computer*, *29*(3), 31–44. https://doi.org/0018-9162/96/$5.000

Jaradat, G. M. (2016). Hybrid elitist-ant system for a symmetric traveling salesman problem: case of Jordan. *Neural Computing and Applications*, *29*(2), 565–578. https://doi.org/10.1007/s00521-016-2469-3

Ji, J., Zhang, N., Liu, C., & Zhong, N. (2006). An Ant Colony Optimization Algorithm for Learning Classification Rules. *WI '06: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, 1034–1037. https://doi.org/http://dx.doi.org/10.1109/WI.2006.35

Jiang, H., Yang, A., Yan, F., & Miao, H. (2016). Research on Pattern Analysis and Data Classification Methodology for Data Mining and Knowledge Discovery. *International Journal of Hybrid Information Technology*, *9*(3), 179–188. https://doi.org/10.14257/ijhit.2016.9.3.17

Jiang, W. J., Xu, Y., & Xu, Y. (2005). A novel data mining algorithm based on ant colony system. In *Fourth International Conference on Machine Learning and*

Cybernetics, (pp. 18–21). Guangzhou.

Jin, P., Zhu, Y., Hu, K., & Li, S. (2006). Classification rule mining based on ant colony optimization algorithm. *Intelligent Control and Automation*, (1), 654–663.

Kanj, S., Abdallah, F., Denœux, T., & Tout, K. (2015). Editing training data for multi-label classification with the k -nearest neighbor rule. *Pattern Analysis and Applications*, *19*(1), 145–161.

Kanj, S., Abdallah, F., Denœux, T., & Tout, K. (2016). *Editing training data for multi-label classification with the k-nearest neighbor rule*. *Pattern Analysis and Applications* (Vol. 19).

Karafotias, G., Hoogendoorn, M., & Eiben, A. E. (2015). Parameter Control in Evolutionary Algorithms: Trends and Challenges. *IEEE Transactions on Evolutionary Computation*, *19*(2), 167–187.

Katsis, C. D., Goletsis, Y., Boufounou, P. V, Stylios, G., & Koumanakos, E. (2012). Using Ants to Detect Fraudulent Financial Statements. *Journal of Applied Finance and Banking*, *2*(6), 73–81. Retrieved from http://search.proquest.com.ezaccess.library.uitm.edu.my/docview/1318921901?accountid=42518

Kazi, J. I., & Mubarak, M. (2014). Use of artificial neural network in diagnostic pathology. *Springer International Publishing*, *56*(10), 467–469. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/17144397

Kesavaraj, G., & Sukumaran, S. (2013). A study on classification techniques in data mining. *2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*, 1–7.

https://doi.org/10.1109/ICCCNT.2013.6726842

Khichane, M., Albert, P., & Solnon, C. (2009). An ACO-based Reactive Framework for Ant Colony Optimization:First Experiments on Constraint Satisfaction Problems. In *In: St utzle T (ed) Learning and Intelligent Optimization, Third International Conference, LION 3, LNCS, vol 5851* (pp. 119–133). Springer, Heidelberg, Germany,: Springer, Heidelberg, German.

Kong, R., & Zhang, B. (2008). A Fast Least Squares Support Vector Machine Classifier. *International Journal of Remote Sensing 26.5*, 749–752.

Kotsiantis, S. B. (2007). Supervised Machine Learning: A Review of Classification Techniques. *Informatica*, *31*, 249–268. https://doi.org/10.1115/1.1559160

Kramer, O. (2010). Evolutionary self-adaptation: A survey of operators and strategy parameters. *Evolutionary Intelligence*, *3*(2), 51–65.

Kubat, M., Holte, R. C., & Matwin, S. (1998). Machine Learning for the Detection of Oil spills in Satellite Radar Images. *Machine Learning*, *30*(2–3), 195–215.

Kudo, M., & Sklansky, J. (2000). Comparison of algorithms that select features for pattern classifiers. *Pattern Recognition*, *33*(1), 25–41. https://doi.org/10.1016/S0031-3203(99)00041-2

Kumar, R., Aggarwal, R. K., & Sharma, J. D. (2015). Comparison of regression and artificial neural network models for estimation of global solar radiation. *Renewable and Sustainable Energy Reviews*, *52*(August), 1294–1299. https://doi.org/10.1016/j.rser.2015.08.021

Lai, C., Shao, Q., Chen, X., Wang, Z., Zhou, X., Yang, B., & Zhang, L. (2016). Flood risk zoning using a rule mining based on ant colony algorithm. *Journal of Hydrology*, *542*, 268–280. https://doi.org/10.1016/j.jhydrol.2016.09.003

Lam, H. K., Ekong, U., Liu, H., Xiao, B., Araujo, H., Ling, S. H., & Chan, K. Y. (2014). A study of neural-network-based classifiers for material classification. *Journal of LATEX CLASS FILES*, *144*, 367–377.

Lavanya, D., & Rani, K. U. (2011). Performance Evaluation of Decision Tree Classifiers on Medical Datasets. *International Journal of Computer Applications*, *26*(4), 975–8887.

Lazarov, V., & Capota, M. (2007). Churn Prediction. *TUM Computer Science*.

Lean, Y., Wang, S., & Kin, K. L. (2006). An Integrated Data Preparation Scheme for Neural Network Data Analysis. *IEEE Transactions on Knowledge and Data Engineering*, *18*(2), 217–230. https://doi.org/10.1109/TKDE.2006.22

Levine, J., & Ducatelle, F. (2004). Ant colony optimization and local search for bin packing and cutting stock problems. *Journal of the Operational Research Society*, *55*(7), 705–716.

Li, Y., Li, G., & Wang, Z. (2015). Rule extraction based on extreme learning machine and an improved ant-miner algorithm for transient stability assessment. *PLoS ONE*, *10*(6), 1–18.

Liang, S. C., Lee, Y. C., & Lee, P. C. (2011). The application of ant colony optimization to the classification rule problem. *IEEE International Conference on Granular Computing, GrC 2011*, 390–392. https://doi.org/10.1109/GRC.2011.6122628

Liang, Z., Sun, J., Lin, Q., Du, Z., Chen, J., & Ming, Z. (2016). A novel multiple rule sets data classification algorithm based on ant colony algorithm. *Applied Soft Computing Journal*, *38*, 1000–1011. https://doi.org/10.1016/j.asoc.2015.10.046

Liao, Stützle, T., Marco, A., & Dorigo, M. (2013). A Unified Ant Colony

Optimization Algorithm for Continuous Optimization. *European Journal of Operational Research*, (February).

Liao, T., Montes, M., Aydin, D., Stützle, T., & Dorigo, M. (2011). An Incremental Ant Colony Algorithm with Local Search for Continuous Optimization. *Gecco-2011: Proceedings of the 13th Annual Genetic and Evolutionary Computation Conference*, (January), 125–132.

Liao, Y., & Vemuri, V. R. (2002). Use of k-nearest neighbor classifier for intrusion detection. *Computers and Security*, *21*(5), 439–448.

Lin, P., Dai, R., Contreras, M. A., & Zhang, J. (2017). Combining ant colony optimization with 1-opt local search method for solving constrained forest transportation planning problems. *Artificial Intelligence Research*, *6*(2), 27.

Liu. (2015). *Rule Based Systems for Classification in Machine Learning Context*. University of Portsmouth.

Liu, Abbass, H. A., & Mckay, B. (2002). Density-Based Heuristic for Rule Discovery With Ant-Miner. *6th Australasia-Japan Joint Workshop on Intelligent and Evolutionary Systems*, 1–6.

Liu, B., Abbass, H. A., & Mckay, B. (2004). Classification rule discovery with ant colony optimization algorithm. In *IEEE Computational Intelligence Bulletin* (Vol. 7435 LNCS, pp. 678–687). https://doi.org/10.1007/978-3-642-32639-4_81

Liu, Gegov, A., & Stahl, F. (2015). Unified framework for construction of rule based classification systems. *Information Granularity, Big Data, and Computational Intelligence*, 209–230. Retrieved from http://link.springer.com/chapter/10.1007/978-3-319-08254-7_10

Liu, J., & Lampinen, J. (2005). A Fuzzy Adaptive Differential Evolution Algorithm.

Soft Computing, *9*(6), 448–462.

Liu, S. (2014). Remote Sensing Classification based on Improved Ant Colony Rules Mining Algorithm. *Journal of Multimedia*, *9*(9), 1105–1112. https://doi.org/10.4304/jmm.9.9.1105-1112

Liu, S. , Liu, S., Cai, W., Pujol, S., Kikinis, R., & Feng, D. (2014). Early diagnosis of Alzheimer's Disease with Deep Learning. *2014 IEEE 11th International Symposium on Biomedical Imaging (ISBI)*, 1015–1018.

Liu, Y. W., & Lin, C. L. (2008). Ideas of control design for cellular signal transduction pathway of ras. *Proceedings - International Conference on Computational Intelligence and Multimedia Applications, ICCIMA 2007*, *1*, 561–563.

López-Ibáñez, M., Stützle, T., & Dorigo, M. (2016). *Ant Colony Optimization: A Component-Wise Overview*. *Handbook of heuristics*. Bruxelles, Belgium. https://doi.org/10.1007/978-3-319-07153-4_21-1

Lopez-Ibnez, M., Stutzle, T., & Dorigo, M. (2016). Ant Colony Optimization: A Component-Wise Overview. In *Springer* (pp. 1–37). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-07153-4_21-1

Lotte, F., Congedo, M., Anatole, L., Lamarche, F., & Arnaldi, B. (2007). A review of classification algorithms for EEG-based brain – computer interfaces To cite this version : A Review of Classification Algorithms for EEG-based Brain-Computer Interfaces. *Journal of Neural Engineering*, *4*, 24.

Lourenco, H., Martin, O. C., & Stutzle, T. (2003). Iterated Local Search. *International Series in Operations Research and Management Science (2003)*, 1–49.

Madhusudhanan, S., Karnan, M., & Rajivgandhi, K. (2010). Fuzzy based ant miner algorithm in datamining for hepatitis. *2010 International Conference on Signal Acquisition and Processing, ICSAP 2010*, 229–232. https://doi.org/10.1109/ICSAP.2010.54

Mangasarian, O. L. (2003). Data mining via support vector machines. In *In System Modeling and Optimization* (Vol. 130, pp. 23–54). Boston,MA: Springer. https://doi.org/10.1007/978-0-387-35699-0

Maojo, V., & Sanandres, J. (2000). A Survey of Data Mining Techniques. In *International Symposium on Medical Data Analysis*. Berlin, Heidelberg: Springer Berlin Heidelberg.

Martens, D., Backer, M. De, Haesen, R., Vanthienen, J., Snoeck, M., & Baesens, B. (2007). Classification With Ant Colony Optimization. *IEEE Transactions On Evolutionary Computation*, *11*(5), 651–665. https://doi.org/10.1109/TEVC.2006.890229

Martens, D., Baesens, B., & Fawcett, T. (2011). Editorial survey: Swarm intelligence for data mining. *Machine Learning*, *82*(1), 1–42. https://doi.org/10.1007/s10994-010-5216-5

Mashhour, E. M., El Houby, E. M. F., Wassif, K. T., & Salah, A. I. (2018). A Novel Classifier based on Firefly Algorithm. *Journal of King Saud University - Computer and Information Sciences*. https://doi.org/10.1016/j.jksuci.2018.11.009

Mavrovouniotis, M., Müller, F. M., Yang, S., & Yang, S. (2016). Ant Colony Optimization With Local Search for Dynamic Traveling Salesman Problems. *IEEE TRANSACTIONS ON CYBERNETICS*, *XX*(April), 1–14.

Merkle, D., Middendorf, M., & Schmeck, H. (2002). Ant colony optimization for resource-constrained project scheduling. *IEEE Transactions on Evolutionary Computation*, *6*(4), 333–346.

Meyer, B. (2004). Convergence control in ACO. In *In Genetic and Evolutionary Computation Conference (GECCO), Seattle, WA, late-breaking paper available on CD*.

Michelakos, I., Papageorgiou, E., & Vasilakopoulos, M. (2010). A hybrid classification algorithm evaluated on medical data. *Proceedings of the Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE*, 98–103. https://doi.org/10.1109/WETICE.2010.22

Mohammad, R. M., Thabtah, F., & McCluskey, L. (2014). Intelligent Rule based Phishing Websites Classification. *IET Information Security*.

Moustakidis, S. P., & Theocharis, J. B. (2010). SVM-FuzCoC: A novel SVM-based feature selection method using a fuzzy complementary criterion. *Pattern Recognition*, *43*(11), 3712–3729.

Mukhopadhyay, A., Maulik, U., Bandyopadhyay, S., & Coello, C. A. C. (2014). A survey of multiobjective evolutionary algorithms for data mining: Part I. *IEEE Transactions on Evolutionary Computation*, *18*(1), 4–19. https://doi.org/10.1109/TEVC.2013.2290086

Murthy, H. S. N., & Meenakshi, M. (2015). Comparison between ANN-Based Heart Stroke Classifiers Using Varied Folds Data Set Cross-Validation. In *Intelligent Computing, Communication and Devices* (Vol. 1, pp. 693–699). New Delhi: Springer, New Delhi. https://doi.org/10.1007/978-81-322-2012-1_72

Nallaperuma, S., Wagner, M., & Neumann, F. (2014). Parameter Prediction Based

on Features of Evolved Instances for Ant Colony Optimization and the Traveling Salesperson Problem. In *In International Conference on Parallel Problem Solving from Nature (pp. 100-109). Springer, Cham* (Vol. 8672, pp. 100–109). Cham: Springer.

Nayar, N., Ahuja, S., & Jain, S. (2019). Swarm intelligence and data mining: A review of literature and applications in healthcare. In *Third International Conference on Advanced Informatics for Computing Research* (pp. 1–7). https://doi.org/10.1145/3339311.3339323

Negulescu, A., Negulescu, S., & Dzitac, I. (2017). Balancing Between Exploration and Exploitation in ACO. *INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL*, *12*(April), 265–275.

Olden, J. D., & Jackson, D. A. (2002). Illuminating the "black box": a ramdomization approach for understanding variable contributions in artificial neuronal networks. *Ecological Modelling*, *154*, 135–150. https://doi.org/10.1016/S0304-3800(02)00064-9

Otero, F. E. B., & Freitas, A. A. (2013). Improving the interpretability of classification rules discovered by an ant colony algorithm. *Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation Conference - GECCO '13*, 73. https://doi.org/10.1145/2463372.2463382

Otero, F. E. B., Freitas, A. A., & Johnson, C. G. (2013a). A new sequential covering strategy for inducing classification rules with ant colony algorithms. *IEEE Transactions on Evolutionary Computation*, *17*(1), 64–76. https://doi.org/10.1109/TEVC.2012.2185846

Otero, F. E. B., Freitas, A. A., & Johnson, C. G. (2013b). A new sequential covering

strategy for inducing classification rules with ant colony algorithms. *IEEE Transactions on Evolutionary Computation*, *17*(1), 64–76. https://doi.org/10.1109/TEVC.2012.2185846

Otero, F., Freitas, A. A., & Johnson, C. G. (2008). cAnt-miner: An ant colony classification algorithm to cope with continuous attributes. In *International Conference on Ant Colony Optimization and Swarm Intelligence.* (pp. 48–59). Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-540-87527-7_5

Otero, F., Freitas, A., & Johnson, C. (2009). Handling continuous attributes in ant colony classification algorithms. In *2009 IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2009 - Proceedings* (pp. 225–231). Morgan Kaufmann.

Otero, O. E. B., Freitas, A. A., & Johnson, C. G. (2012). Inducing Decision Trees with an Ant Colony Optimization Algorithm. *Applied Soft Computing*, *12*(11), 3615–3626. https://doi.org/10.1016/j.asoc.2012.05.028

Parpinelli, R., Lopes, H., & Freitas, A. A. (2002a). Data mining with an ant colony optimization algorithm. *IEEE Transactions on Evolutionary Computation*, *6*(4), 321–332. https://doi.org/10.1109/TEVC.2002.802452

Parpinelli, R., Lopes, H., & Freitas, A. A. (2002b). Data Mining With an Ant Colony Optimization Algorithm. *IEEE Transactions on Evolutionary Computation*, *47*(6 (4)), 321–332.

Patil, T. R., & Sherekar, S. (2013). Performance Analysis of Naive Bayes and J48 Classification Algorithm for Data Classification. *International Journal Of Computer Science And Applications*, *6*(2), 256–261.

Peng, W., Chen, J., & Zhou, H. (2009). An Implementation of IDE3 Decision Tree

Learning Algorithm.

Phyu, T. N. (2009). Survey of Classification Techniques in Data Mining. *International MultiConference of Engineers and Computer Scientists*, *I*, 18–20.

Pilat, M. L., & White, T. (2002). Using genetic algorithms to optimize ACS-TSP. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *2463*, 282–287.

Prabha, G. M., & Balraj, E. (2014). A HM Ant Miner Using Evolutionary Algorithm. *International Journal of Innovative Research in Science, Engineering and Technology*, *3*(3), 1687–1692.

Qin, A. K., Huang, V. L., & Suganthan, P. N. (2009). Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE Transactions on Evolutionary Computation*, *13*(2), 398–417. https://doi.org/10.1109/TEVC.2008.927706

Quinlan, J. R. (1987). Generating production rules from decision trees. *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, *30107*, 304–307.

Quinlan, J. R. (2014). *C4. 5: programs for machine learning*. Elsevier.

Rajpiplawala, S., & Singh, D. K. (2014). Enhanced cAntMiner PB Algorithm for Induction of Classification Rules using Ant Colony Approach. *IOSR Journal of Computer Engineering (IOSR-JCE)*, *16*(3), 63–72.

Ramalingam, S., & Sujatha, P. (2018). An extensive work on stock price prediction using Ant Colony Optimization algorithm (ACO-SPP). *International Journal of Intelligent Engineering and Systems*, *11*(6), 85–94. https://doi.org/10.22266/IJIES2018.1231.09

Raymer, M. L., Punch, W., Goodman, E., Kuhn, L., & Jain, A. (2000). Dimensionality reduction using genetic algorithms - Evolutionary Computation. *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION*, *4*(2), 164–171.

Robu, R., Vacar, C., Robu, N., & Holban, S. (2016). A study on Ant Miner parameters. In *6th International Conference on Information, Intelligence, Systems and Applications*. https://doi.org/10.1109/IISA.2015.7388032

Rokach, L., & Maimon, O. (2005). Top-Down Induction of Decision Trees Classifiers—A Survey. *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, *35*(4), 476–487.

Romero, C., Ventura, S., Espejo, P. G., & Hervás, C. (2008). Data mining algorithms to classify students. *Educational Data Mining 2008*, 8–17.

Sabri, S. N., & Saian, R. (2017). Predicting Flood in Perlis Using Ant Colony Optimization. *Journal of Physics: Conference Series*, *855*(1), 1–8. https://doi.org/10.1088/1742-6596/855/1/012040

Sagban, R. (2016). *Reactive Approach for Automating Exploration and Exploitation in Ant Colony Optimization*. Unpublished doctoral dissertation.Universiti Utara Malaysia, Malaysia.

Sagban, R., Ku-Mahamud, K. R., & Abu Bakar, M. S. (2016). Reactive max-min ant system with recursive local search and its application to TSP and QAP. *Intelligent Automation and Soft Computing*, *23*(1), 127–134.

Sagban, R., Ku-Mahamud, K. R., & Shahbani Abu Bakar, M. (2015). Nature-inspired Parameter Controllers for ACO-based Reactive Search. *Research Journal of Applied Sciences, Engineering and Technology*, *11*(1), 109–117. https://doi.org/10.19026/rjaset.11.1682

Saian, R. (2013). *A HYBRID OF ANT COLONY OPTIMIZATION ALGORITHM AND SIMULATED ANNEALING FOR CLASSIFICATION RULES*. UUM.

Saian, R., & Ku-Mahamud, K. R. (2011). Hybrid Ant Colony Optimization and Simulated Annealing for Rule Induction. *2011 UKSim 5th European Symposium on Computer Modeling and Simulation*, (March 2016), 70–75. https://doi.org/10.1109/EMS.2011.17

Saian, R., & Ku-Mahamud, K. R. (2012). Ant colony optimization for rule induction with simulated annealing for terms selection. *Proceedings - 2012 14th International Conference on Modelling and Simulation, UKSim 2012*, (March 2012), 33–38. https://doi.org/10.1109/UKSim.2012.115

Sakthipriya, N., & Kalaipriyan, T. (2015). Variants of Ant Colony Optimization- A State of an Art. *Indian Journal of Science and Technology*, *8*(November).

Salama, K., & Freitas, A. (2012). ABC-Miner: An ant-based Bayesian classification algorithm. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *7461 LNCS*, 13–24. https://doi.org/10.1007/978-3-642-32650-9_2

Salama, K. M., & Abdelbar, A. M. (2010). Extensions to the Ant-Miner classification rule discovery algorithm. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *6234 LNCS*, 167–178.

Salama, K. M., Abdelbar, A. M., & Freitas, A. A. (2011a). Multiple pheromone types and other extensions to the Ant-Miner classification rule discovery algorithm. *Swarm Intelligence*, *5*(3–4), 149–182.

Salama, K. M., Abdelbar, A. M., & Freitas, A. A. (2011b). Multiple pheromone

types and other extensions to the Ant-Miner classification rule discovery algorithm. *Swarm Intelligence*, *5*(3–4), 149–182. https://doi.org/10.1007/s11721-011-0057-9

Salama, K. M., Abdelbar, A. M., Otero, F. E. B., & Freitas, A. A. (2013). Utilizing multiple pheromones in an ant-based algorithm for continuous-attribute classification rule discovery. *Applied Soft Computing Journal*, *13*(1), 667–675. https://doi.org/10.1016/j.asoc.2012.07.026

Salama, K. M., & Otero, F. E. B. (2013). Using a unified measure function for heuristics, discretization, and rule quality evaluation in Ant-Miner. *2013 IEEE Congress on Evolutionary Computation, CEC 2013*, 900–907.

Sarkar, B. K., Sana, S. S., & Chaudhuri, K. (2012). A genetic algorithm-based rule extraction system. *Applied Soft Computing*, *12*(1), 238–254. https://doi.org/10.1016/j.asoc.2011.08.049

Schwefel, H. (2004). On the Evolution of Evolutionary Algorithms. *Genetic Programming: 7th European Conference, EuroGP 2004 (Coimbra, Portugal, April 5-7, 2004)*, *3003*, 389–398. https://doi.org/10.1007/978-3-540-24650-3_37

Seerat, B., & Qamar, U. (2015). Rule Induction Using Enhanced RIPPER Algorithm for Clinical Decision Support. In *Sixth International Conference on Intelligent Control and Information Processing* (Vol. 33, pp. 83–91). Wuhan, China: IEEE. https://doi.org/10.1109/ICICIP.2015.7388149

Selvi, V., & Umarani, R. (2010). Comparative Analysis of Ant Colony and Particle Swarm Optimization Techniques. *International Journal of Computer Applications*, *5*(4), 975–8887.

Shahzad, W., & Baig, A. R. (2010). Compatibility As a Heuristic for Construction of

Rules By Artificial Ants. *Journal of Circuits, Systems and Computers*, *19*(01), 297–306.

Smaldon, J. (2006). *Data mining with an ant colony optimization algorithm*. *Evolutionary Computation, IEEE Transactions* (Vol. 6).

Srinivas, M., & Patnaik, L. M. (1994). Genetic Algorithms: A Survey. *Computer*, *27*(6), 17–26.

Strecht, P. (2015). A Survey of Merging Decision Trees Data Mining Approaches. *10th Doctoral Symposium in Informatics Engineering (DSIE'15)*, 36–47.

Stutzle, T. (1998). *Local Search Algorithms for Combinatorial Problems- Analysis, Improvements, and New Applications*. *Technische Universit¨ at Darmstadt*. Technische Universiti at Darmstadt. https://doi.org/10.1007/BF00586221

Stutzle, T., & Hoos, H. (1997). MAX MIN Ant System and Local Search for the Traveling Salesman Problem. *IEEE International Conference on Evolutionary Computation (Icec'97)*, 309–314.

Stutzle, T., & Hoos, H. (2000). MAX-MIN Ant System. *Future Generation Computer Systems*, *16*(8), 889–914.

Stutzle, T., Lopez-Ibnez, M., Pellegrini, P., Maur, M., Oca, M. M., Birattari, M., & Dorigo, M. (2010). *Parameter Adaptation in Ant Colony Optimization*. Bruxelles,Belgium.

Stützle, T., & Ruiz, R. (2017). Iterated Local Search. In *Handbook of Heuristics* (pp. 1–27). Springer International Publishing. https://doi.org/10.1007/978-3-319-07153-4

Thangavel, K., & Jaganathan, P. (2007). Rule Mining Algorithm with a New Ant Colony Optimization Algorithm. *International Conference on Computational*

Intelligence and Multimedia Applications, ICCIMA 2007, *1*, 561–563.

Tkáč, M., & Verner, R. (2015). *Artificial neural networks in business: Two decades of research*. *Applied Soft Computing* (Vol. 38). Elsevier B.V. https://doi.org/10.1016/j.asoc.2015.09.040

Toksari, M. D. (2016). A hybrid algorithm of Ant Colony Optimization (ACO) and Iterated Local Search (ILS) for estimating electricity domestic consumption: Case of Turkey. *International Journal of Electrical Power and Energy Systems*, *78*, 776–782. https://doi.org/10.1016/j.ijepes.2015.12.032

Tripathy, S., Hota, S., & Satapathy, P. (2013). MTACO-Miner : Modified Threshold Ant Colony Optimization Miner for Classification Rule Mining. In *Emerging Reserch in Computing, Information, Communication and Application* (pp. 1–6). Elsevier.

Uthayakumar, J., Vengattaraman, T., & Dhavachelvan, P. (2017). Swarm intelligence based classification rule induction (CRI) framework for qualitative and quantitative approach: An application of bankruptcy prediction and credit risk analysis. *Journal of King Saud University - Computer and Information Sciences*. https://doi.org/10.1016/j.jksuci.2017.10.007

Venkatesh, B., & Anuradha, J. (2019). A Review of Feature Selection and Its Methods. *Cybernetics and Information Technologies*, *19*(1), 3–26. https://doi.org/10.2478/cait-2019-0001

Waal, I. (2017). Skin Cancer Diagnosed by Using Artificial Intelligence on Clinical Images. *Oral Diseases*, *12*(10), 3218–3221. https://doi.org/10.1111/ijlh.12426

Wah, Y. B. W., Ibrahim, N., Hamid, H. A., Abdul-Rahman, S., & Fong, S. (2018). Feature selection methods: Case of filter and wrapper approaches for

maximising classification accuracy. *Pertanika Journal of Science and Technology*, *26*(1), 329–340. Retrieved from http://www.pertanika.upm.edu.my/

Wahid, J., & Al-Mazini, H. F. A. (2018). Classification of Cervical Cancer Using Ant-Miner for Medical Expertise Knowledge Management. In *Knowledge Management International Conference (KMICe)*. Miri Sarawak, Malaysia.

Wang, G., Chu, H. C. E., Zhang, Y., Chen, H., Hu, W., Li, Y., & Peng, X. J. (2015). Multiple parameter control for ant colony optimization applied to feature selection problem. *Neural Computing and Applications*, *26*(7), 1693–1708. https://doi.org/10.1007/s00521-015-1829-8

Wang, Y., Cai, Z., & Zhang, Q. (2011). Differential evolution with composite trial vector generation strategies and control parameters. *IEEE Transactions on Evolutionary Computation*, *15*(1), 55–66.

Wang, Z., & Feng, B. (2004). Classification Rule Mining with an Improved Ant Colony Algorithm. In *Australian conference on artificial intelligence* (pp. 357–367).

Weston, J., & Watkins, C. (1998). *Multi-class Support Vector Machines*. *Department of Computer Science, Royal Holloway, University of London*.

Witten, I. E., & Frank, E. (2005). *Data Mining: Practical machine learning tools and techniques*. Elsevier.

Witten, I., Eibe, F., Mark, H., & Christopher, P. (2016). *Data Mining: Practical Machine Learning Tools and Techniques* (Vol. 40). Elsevier.

Wu, C. (2008). Ant colony optimization on building an online delayed diagnosis detection support system for emergency department. In *The 7th International Conference on Computational Intelligence in Economics and Finance (CIEF*

2008) (pp. 1–7).

Wu, H., & Sun, K. (2012). A simple heuristic for classification with ant-miner using a population. In *the 2012 4th International Conference on Intelligent Human-Machine Systems and Cybernetics, IHMSC 2012* (Vol. 1, pp. 239–244). https://doi.org/10.1109/IHMSC.2012.67

Wu, X., Kumar, V., Ross, Q. J., Ghosh, J., Yang, Q., Motoda, H., … Steinberg, D. (2008). Top 10 algorithms in data mining. *Knowledge and Information Systems*, *14*(1), 1–37.

Wu, X., Zhu, X., Wu, G. Q., & Ding, W. (2014). Data mining with big data. *IEEE Transactions on Knowledge and Data Engineering*, *26*(1), 97–107.

Xu, Y., Zhu, Q., Fan, Z., Qiu, M., Chen, Y., & Liu, H. (2013). Coarse to fine K nearest neighbor classifier. *Pattern Recognition Letters*.

Yadav, A. K., & Chandel, S. S. (2013). Solar radiation prediction using Artificial Neural Network techniques: A review. *Renewable and Sustainable Energy Reviews*, *33*, 772–781.

Yang, L., Li, K., Zhang, W., & Ke, Z. (2016). Ant colony classification mining algorithm based on pheromone attraction and exclusion. *Soft Computing*, 1–13. https://doi.org/10.1007/s00500-016-2151-9

Yang, L., LI, K., Zhang, W., Wang, Y., & Ke, Z. (2015). Application of a New mAnt-MinerPR Algorithm in Classification Rule Mining. *Journal of Digital Information Management*, *13*(5).

Yang, X.S. (2010). A New Metaheuristic Bat-Inspired Algorithm. *Springer Nature Inspired Cooperative Strategies for Optimization*, 65–74.

Yildiz, O. T. (2013). Omnivariate rule induction using a novel pairwise statistical

test. *IEEE Transactions on Knowledge and Data Engineering*, *25*(9), 2105–2118. https://doi.org/10.1109/TKDE.2012.155

Zahiri, S. H. (2012). Fuzzy gravitational search algorithm an approach for data mining. *Iranian Journal of Fuzzy Systems*, *9*(1), 21–37.

Zhang, X., & Sun, W. (2016). An Adaptive Ant Colony Algorithm for Classification Rule Mining. In *2nd International Conference on Artificial Intelligence and Industrial Engineering (AIIE2016)* (Vol. 133, pp. 295–299).