Meta-heuristic search methods for big data analytics and visualization of frequently changed patterns.

Submitted in fulfilment of the requirements of the degree of Doctor of Philosophy in Information Technology (IT) in the Faculty of Accounting and Informatics at Durban University of Technology, Durban - South Africa.

Student Name: Israel Edem Agbehadji

Date Submitted:

Signature:[redacted] Date: March 20, 2019

Prof. Richard Charles Millham

(**Supervisor**)

Signature: [redacted] Date: March 20, 2019

Prof. Hongji Yang

(Co-Supervisor)

Abstract

Throughout the world, data plays a prominent role in making decisions relevant to the socio-economic growth of organizations. As organizations grow, they tend to use diverse technologies or platforms to collect data and make data readily available for quick decision-making. These technologies have resulted in exponential growth of data whereby the problem of managing this data in a limited time interval increases in complexity, starting from the preprocessing stage to the visualization stage. Apart from the issue of managing the huge growth of data, finding a suitable method to manage certain aspects of this frequently changed data has been overlooked. These frequent changes in data form the topic of interest of this thesis. Consequently, there is a need to develop a framework both to manage big data at different stages of processing, from preprocessing to visualization, and to handle frequently changed data. The need to develop such a framework arises because traditional methods/algorithms are limited to finding frequent patterns of frequently occurring items while overlooking frequently changed data, which has a numeric and time dimension that can provide interesting business insights. Additionally, traditional visualization methods are challenged with performance scalability and response time. This thesis looked at resolving this limitation by using a meta-heuristic/bio-inspired algorithm that is modelled based on observation of the behavior and characteristics of two different animals, namely the kestrel and the dung beetle. The motivation behind the use of these animals is their ability to explore, exploit and adapt to different situations in their natural environment. The development of the computational model and testing with actual data were formulated as a six-step procedure. Based on the six steps, the proposed computational model was evaluated against selected comparative algorithms, namely BAT, WSA-MP, PSO, Firefly and ACO. The main findings on optimal value/results suggest that, in handling frequently changed data during the data preprocessing, pattern discovery and visualization stages, the proposed computational models performed optimally against the comparative meta-heuristic algorithms on test datasets. Further statistical tests, using the Wilcoxon signed rank test, were conducted on optimal results from the comparative meta-heuristic algorithms. The basis for using the statistical procedure was to select the best choice of algorithm without making any underlying assumption on accuracy of results from the comparative meta-heuristic algorithms. Theoretically, the study contributes to enhancing frequency of item frameworks by including time and numeric dimensions of item occurrence. Practically, the contribution of the study lies in its finding frequently changed patterns in big data analytics. Additionally, the concept of half-life of substances/trails was applied as part of the computational model, and this also forms part of the unique contribution of this thesis. The half-life constitutes the lifetime of interestingness of recent patterns that were discovered. In summary, this thesis is about the mathematical formulation of animal behavior and characteristics into an implementable big data management algorithm and its application to frequently changed patterns.

Declaration

Name: ISRAEL EDEM AGBEHADJI

Student Number: 21648757

Ph.D. (Information Technology)

I hereby declare that this research project is the result of my own work, except for quotations and summaries that have been duly acknowledged.

Signature: [redacted] Date: March 20, 2019.

(**Author**)

APPROVED FOR FINAL SUBMISSION

[redacted] March 20, 2019.

NAME DATE

Dedication

I dedicate this piece of work to the Almighty God, my brother, Selom Coffie Agbehadzi, my father, Thomas Kwashie Agbehadzi, and my mother, Mary Dagodzo, for their support and encouragement during this journey to greatness.

The journey to greatness always starts with a step, but without sacrifice it is impossible. If you decide to take a step, then you need diligence, an open mind, perseverance, patience, networking with people, forgiveness and courage. In all your endeavor, be confident, believe in yourself and have faith in God.

“I speak not of you all: I know whom I have chosen: but that the scripture may be fulfilled, He that eateth bread with me hath lifted up his heel against me.” John 13:18 (King James Version).

Acknowledgements

I wish to thank the Almighty God for his infinite mercies and guidance in the completion of this research work. I am deeply grateful to my supervisor, Prof. Richard Charles Millham of Durban University of Technology, and my co-supervisor, Prof. Hongji Yang of Leicester University, for their immense support and guidance during the past two years. Prof. Millham guided every aspect of my Ph.D. study and I am deeply grateful for his guidance during the face-to-face meeting on the progress of my studies. His perspective and mindset have made me an independent researcher ready to confront the challenges of the world. Without his guidance and support I could not have completed this thesis. I am most grateful for the assistance of Prof. Fong, of the University of Macau, with the thesis and publications emanating from this thesis.

I am deeply grateful to my parents, Mary Dagodzo and Thomas Kwashie Agbehadzi, for their love and continuous encouragement and prayers. Many thanks go to my lovely brother Selom Coffie Agbehadzi for the financial support, love, care and encouragements. Many thanks also go to my niece, Elorm Osei for her constant love and prayers. I am very thankful to Londiwe Purity Mndaba, a very good and supportive friend who assisted me in time of difficulties. I also thank Dr. Hillar Addo and Mr. Daniel Obuobi (Central University of Ghana, Miotso Campus) for their support and guidance. I also thank Mr. Evan Dogbe for his assistance. Many thanks go to Suzzana Agbehadzi, Sis Ama and her family, brother Steven and his family, brother Fieldgate and his family, Mr. George Stephenson, Mr. Richard Kwame Asare and his family, Mr. Jude N. Danbo and his family, Mr. Emmanuel Freeman, Mr. Daniel Nettey etc. I also thank all my loved ones for their words of encouragement. Your sacrifices have enabled me to pursue this path of greatness to completion.

Research emanating from this thesis

Conference papers:

- Agbehadji, I. E., Millham, R. and Fong, S. 2016. Wolf search algorithm for numeric association rule mining**.** 2016 *IEEE International Conference on Cloud Computing and Big Data Analysis (ICCCBDA 2016).* Chengdu, China. doi: 10.1109/ICCCBDA.2016.7529549
- Agbehadji, I. E., Millham, R. and Fong, S. 2016. Kestrel-based search algorithm for association rule mining and classification of frequently changed items. *IEEE International Conference on Computational Intelligence and Communication Networks*, Dehadrun, India. doi:10.1109/CICN.2016.76
- Agbehadji, I. E., Millham, R., Fong, S. J. and Yang, H. 2018. The comparative analysis of Smith-Waterman algorithm with Jaro-Winkler algorithm for the detection of duplicate health related records. *IEEE International Conference on Advances in Big Data, Computing and Data Communication Systems (icABCD 2018), Durban, South Africa.* 1-10. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8465458
- Agbehadji, I. E., Millham, R., Fong, S. J. and Yang, H. 2018. Kestrel-based Search Algorithm (KSA) and Long Short Term Memory (LSTM) Network for feature selection in classification of high-dimensional bioinformatics datasets. *Federation Conference of Computer Science and Information systems (FedCSIS).* Poznan, Poland, Poland. 15:15-20. Available: https://ieeexplore.ieee.org/document/8511220

Journal publications:

- Agbehadji, I. E., Millham, R., Fong, S. J. and Yang, H. 2018. Bioinspired computational approach to missing value estimation. *Mathematical Problems in Engineering-Hindawi.* doi.org/10.1155/2018/9457821

- Agbehadji, I. E., Millham, R., Surendra, T., Yang, H. and Addo, H. 2018. Visualization of frequently changed patterns based on the behaviour of dung beetle. *Fourth International Conference on Soft Computing in Data Science 2018 (SCDS2018), Chulalongkorn University, Bangkok, Thailand from 15-16 August 2018*: *Communications in Computer and Information Science, Springer Nature Singapore***.** 937: 230-245. Available**:** https://link.springer.com/chapter/10.1007/978-981-13-3441-2_18
- Agbehadji, I. E., Millham, R., Fong, S. J. and Yang, H. (In press). Integration of Kestrel-based search algorithm with Artificial Neural Network (ANN) for feature subset selection in classification. (Accepted for publication: *International Journal of Bio-inspired computation*).
- Agbehadji, I. E., Millham, R., Fong, S. J. and Yang, H. 2018. Dung beetle-based search algorithm for missing value estimation in variable dimensions of dataset. (Under review: *International Journal of Bio-Inspired Computation*).

- Eight book chapters in *Research Trends in Bioinspired Algorithms for Bigdata Analytics*, Taylor and Francis, UK (under review)

Table of contents

List of Figures

List of Tables

Abbreviations

ACO	Ant Colony Optimization
BIDE	Bi-Directional Extension
BI-TSP	Top-k closed sequential patterns with Bi-Directional checking scheme
BLAST	Basic Local Alignment Search Technique
COBRA	Closed sequential pattern mining using a Bi-phase Reduction Approach
DBA	Dung Beetle-based Search algorithm
FASTA	FAST All
FP	Frequent-pattern
GA	Genetic algorithm
KSA	Kestrel-Based Search algorithm
MAR	Missing At Random
MCAR	Missing Completely At Random
MCPsc	Modified Closeness Preference Confidence value
MNAR	Missing Not At Random
MSF	Multiple species flocking
Par-CSP	Parallel Closed Sequential Pattern mining
PCA	Principal components analysis
PSO	Particle Swarm Optimization
SIBA	Sampling Improved Bat Algorithm
SPADE	Sequential Pattern Discovery
SPAM	Sequential Pattern Mining using bitmap representation
SODSS	Service-Oriented Decision Support System
SOM- MBP	Self-Organizing Map-Multiple Back-Propagation
TF-IDF	Frequency-Inverse Document Frequency
WSA	Wolf Search Algorithm
WSA-MP	Wolf Search Algorithm with Minus step Previous
k-NN	kth Nearest Neighbor

CHAPTER 1: GENERAL INTRODUCTION AND SUMMARY

1.1 Introduction

The chapter starts with the background of the study on the current dispensation of big data. In subsequent sections, the general objectives of the study, specific objectives, research questions, problem statement, justification of the study, significance of the study and outline of the thesis are discussed.

1.2 Background of the study

Generally, large volumes of data are generated continuously from heterogeneous and autonomous sources (that is, when each data source collects information without relying on any centralized control) (Banupriya and Vijayadeepa 2015), where each source has a different set of data residing on its platforms, including data that is frequently changed. As lots of data is generated and collected, it needs to be analyzed to identify patterns, including those of special interest, such as frequently changed data. For instance, industries such as retail, financial services, and healthcare institutions have frequently changed data that needs specific handling to find interesting patterns/trends from the volume of data generated. Mostly, data repositories are unable to process demands from a big dataset that needs to be updated frequently within specified time (Rouse 2018). This frequent updating of data is because of frequent changes that occur in a specified dataset. It is significant that as data continues to increase in volume, exploring the evolving relationship in frequently changed data creates an opportunity for finding new methods of searching for interesting frequently changed patterns. The application of meta-heuristic search methods could help create new computational models that adapt to frequently changed data in order to discover actionable sequences from data for decision-making. This study seeks to research the application of meta-heuristic search methods (also referred to as bio-inspired search methods) for big data analytics and visualization of frequently changed patterns. The reason for carrying out a study into meta-heuristic search methods is that the traditional methods (such as existing data mining algorithms, which will be discussed in Chapter Two) for exploring evolving relationships are not well adapted to processing update demands because of frequent change from large volumes of data.

Massive amounts of data can be analyzed and interpreted using different techniques, algorithms, tools and models. Mostly, existing search algorithms for discovering patterns are focused on frequency of items without considering the frequently changed items. Often, existing data mining algorithms (such as the Apriori algorithm) are focused on the frequency of items without considering the numeric value and time dimensions. Basically, the frequency of items is computed by counting the occurrence of items in transaction (Rajasekaran and Song 2006) to determine a pattern without indicating the change that happened on an item, for instance considering buying behavior of consumers of a retail shop without considering the changing total value(s) of items bought (in terms of the price of items).

Meanwhile, frequency counts on changing numeric value with time may give more meaning to a predicted pattern. Usually, in frequent item mining, an itemset is regarded as interesting if its occurrence frequency exceeds a user-specified threshold (Fung, Wang and Liu 2012). However, the use of frequency counts to measure pattern interestingness is insufficient (Tseng, Liang and Chu 2006) in selecting actionable sequences associated with expected quality and business impact. This is because the patterns identified under the frequency framework do not disclose the business value (such as profit) (Yin *et al.* 2013). As businesses are interested in business value, the frequency framework should be expanded to cover both time and numeric value dimensions. Thus, a multiple-dimension approach for business value analysis may yield more interesting patterns and could be sufficient for a business to select actionable sequences. A meta-heuristic search algorithm could play a significant role in helping to select actionable sequences for business decision-making. Thus, this study seeks to research the application of meta-heuristic search methods (also referred to as bio-inspired search methods) to discover frequently changed patterns from large datasets. The study also uses meta-heuristic search methods to visualize frequently changed patterns.

1.3 Field of research

The field of study is Information Technology, with a focus on meta-heuristic search methods for the analysis and visualization of frequently changed patterns in big data analytics.

1.4 General objective

This study seeks to develop a largely bio-inspired approach to manage and analyze big data, notably frequently changed patterns/items. This approach consists of the use of bio-inspired algorithms that are implemented in three phases, namely data cleansing, data mining and data visualization. These phases are used to build a computational model that depicts the characteristics of animals for analysis and visualization of frequently changed patterns/items.

The first phase, data cleansing, uses enhanced algorithms to extrapolate likely values on missing data, identify and eliminate duplicate text from frequently changed items, and select relevant features. During the first phase, an innovative search algorithm based on hunting behavior of a bird called the kestrel (a bio-inspired algorithm) is mathematically expressed to depict its characteristics. This mathematical expression is applied to extrapolate missing values at random, while an enhanced non-bio-inspired algorithm based on the Smith-Waterman algorithm is applied to identify and eliminate duplicate text. Finally, the mathematical formulation based on kestrels' characteristics is combined with other search methods (that is, deep learning methods that will be discussed in Chapter Two in the literature review) for the selection of relevant features. Phase two is the data mining of frequently changed items with numeric value and time dimensions. During this stage, the kestrel-based algorithm and closeness preference interestingness model is applied to find frequently changed patterns. Phase three is the data visualization of association rules from phase two. A novel search algorithm based on behavior of dung beetles is applied to visualize data on the association rules.

In each phase of the proposed computational model, mathematical formulations of the selected animal behavior are applied, which are translated into algorithmic structure and evaluated by comparing them with other comparative meta-heuristic search methods. The phases of the methodological framework are summarized in Table 1.1.

Table 1.1: Methodological framework on Phases, stages and algorithms

Phases	**Stages**	**Proposed algorithm**	**Comparative algorithms**
Phase 1: Data cleansing/preprocessing	Stage 1: Identify and eliminate duplicate text	Enhanced Smith-Waterman algorithm	Jaro-Winkler distance metrics
	Stage 2: Extrapolating missing data values	KSA	WSA-MP, BAT and Firefly algorithms
	Stage 3: Feature selection	KSA	WSA-MP, BAT, ACO and PSO algorithms
Phase 2: Data mining	-	KSA	ACO, BAT, PSO and WSA-MP algorithms
Phase 3: Data visualization (using linear graph)	-	DBA	ACO for data visualization, Bee algorithm

Source: (Researcher 2018).

The phases in Table 1.1 represents a methodological framework for this thesis that will be discussed in chapter 3.

1.4.1 Specific objectives

The specific objectives of the study are to:

- model a meta-heuristic/bio-inspired data preprocessing approach to extrapolate missing values, identify and remove duplicates text, and select features in subsets
- model an association rule mining approach based on the hunting behavior of kestrel to discover patterns that are frequently changed with numeric value and time dimensions
- based on the frequently changed rules, model a bio-inspired algorithm for visualization of frequently changed items
- empirically validate the models and algorithmic structures against comparative meta-heuristic methods using benchmark datasets.

1.5 Research questions

The importance of the research question is that it enables the researcher to make a claim about knowledge (referred as ontology) so as to understand a knowledge claim, the strategy that informs a procedure, and the method of data collection and analysis (Creswell 2013). The researcher is thus able to take a different approach to solving the research problem. In this context the different approach relates to the search method and its underlying algorithmic structure that could be applied.

The research questions were formulated as follows:

- Can a largely meta-heuristic/bio-inspired data preprocessing approach be modelled to extrapolate missing values, identify and remove duplicate text, and select features in subsets?
- Can a mathematical expression and subsequent algorithm be formulated based on the hunting behavior of the kestrel to discover association rules on frequently changed patterns within numeric value and time dimensions?
- Based on the frequently changed rules, can a bio-inspired algorithm for the visualization of these association mining results be modelled?

- Can the model and algorithmic structure be empirically validated on a benchmark dataset and evaluated against comparative meta-heuristic algorithms?

1.6 Problem statement

Pattern interestingness suffers from the issues of inherent subjectivity and from the need to sort through large volumes of data and report on a huge number of potentially interesting patterns that meet predefined criteria (Vreeken and Tatti 2014). Tseng, Liang and Chu (2006) indicate that the use of frequency of occurrence of items, as a sole criterion, to measure pattern interestingness is insufficient in selecting actionable sequences for an organization (Yin *et al.* 2013). Similarly, Vreeken and Tatti (2014) describe the use of frequency of items as not a very good measure of interestingness. The main reason for this problem with using only the frequency of occurring items for pattern selection is that actionable patterns can change with time (Huynh 2010). Thus, relying on frequency of occurring items alone for pattern selection – rather than on additional criteria such as frequency of change, which entails a time dimension within the data items of a pattern – is a limitation of current big data management approaches and a gap that this thesis hopes to address. Hence, there is a gap in the current occurrence framework since it does not address the time dimension of frequently changed data.

Within the current dispensation of big data, when data becomes very large (that is, big data), it is possible that current approaches to data preprocessing/cleansing (to find missing values, identify duplicates and select relevant features) to find interesting patterns could lose its business value, in terms of having to determine in a timely manner (that is, speedily) the usefulness of an action. For this reason, new algorithms must be developed to accurately preprocess data and discover patterns that are interesting. The lack of accuracy in existing big data preprocessing frameworks is a problem that needs to be solved in order to improve on performance of the data processing aspect of big data platforms.

In view of the challenges of data preprocessing, this study seeks to fill the gaps identified by proposing a new computational model to find the best possible approach. The study uses the hunting behavior of kestrels (bio-inspired), which involves perch hunting (for exploitation of the search area) and hover hunting (for exploration of the search area). Hover hunting involves exploring very large search areas that were left unexploited during perch hunting. Again, hover hunting is quick and involves cross-territorial search for interesting patterns, while perched hunting involves thorough search in a local territory. This bio-inspired approach is a random search algorithm that is used to extrapolate missing values at random, select relevant feature subsets in the data cleansing phase and discover association rules to form a global schema of interesting patterns to disclose actionable sequences. The association rules, when used, can disclose relationships between subtle patterns and group patterns based on the nature of frequent change. In fact, no single pattern can prove to be interesting until all related patterns are grouped to disclose interesting frequently changed patterns. The disclosed pattern will provide meaningful insights when selecting actionable sequences for businesses, irrespective of the volume of data.

Having disclosed the patterns, an actionable sequence could be visualized by using a simple and low computational cost visualization approach that takes into consideration volumes of data and that displays results in less computational time. This suggests that current approaches (such as dense pixel display, stacked display (Keim 2000) and cellular ant-based approaches (Moere, Clayden and Dong 2006)) to visualization, which are discussed in subsequent chapters, are computationally costly. The present approach applies the orientation and navigation mechanism of dung beetles in natural environments to provide a simple algorithm for data visualization with less computational time in a simple two-dimensional graph.

1.7 Justification of the study

The justification is based on the fact that as data frequently changes, and businesses are interested in business value (in terms of disclosing action that should be undertaken within a time dimension). In view of this, the current frequency framework should be expanded to cover both time and numeric value dimensions of frequently changed items so as to bridge the gap identified in the occurrence of item frameworks. Thus, a kestrel behavior is an approach that uses flight and perch behavior to exploit and explore rules to disclose interesting patterns, which may be sufficient for a business to select actionable sequences from large volumes of data so as to add value to the business.

1.8 Significance of the study

This study expresses in mathematical terms the behavior of kestrels and dung beetles, and implements it as a search algorithm. The advantage of the kestrel-based approach is that there is a quick cross-boundary pattern search when there is limited knowledge on characteristics of data as it becomes large/small. This study proposes a novel search approach to data preprocessing that is based on the kestrel. The search approach referred to as the Kestrel-based Search Algorithm (KSA) looks across different search spaces for possible solutions in different aspects of big data analytics (such as volume, velocity and value). It is significant to note that as users of big data analytics acquire and have access to more data (large volume or quantity), they often become more interested in finding patterns that can perfectly explain a frequent change. In so doing, big data users could focus on making valuable conclusions from a large quantity of high-quality data. However, making valuable conclusions may require selection of relevant data from irrelevant data to ensure data is reliable and approximately accurate. The significance of the KSA is to provide a means of making approximately accurate estimations in different problem dimensions of concern as the same data item is collected over and over with different values based on different times.

Theoretically, the proposed meta-heuristic approach seeks to enhance the occurrence framework of items in data mining by generating basic rules from kestrels' behavior. These basic rules are used to evaluate pattern interestingness of frequently changed data/items to select actionable sequences (Yin *et al*. 2013) in a form of patterns from large volumes of data. The outcome could be of practical significance to large stock market items analysis or similar domains in terms of disclosing patterns that have frequently changed within numeric value and time dimensions. The outcome may also be of practical significance to the retail industry for disclosing patterns that are frequently changed. In this context, items with numeric value (such as price) and a time dimension (for instance, in seconds or hours) can be discovered when this algorithm is built.

Another theoretical significance is the use of navigation and orientation by dung beetles for data visualization of frequently changed patterns. The dung beetle is well adapted to navigate different environments to get its food to its home with less energy. As it navigates and orients, it leaves traces that can be seen. This navigation and orientation are mathematically modelled and translated into an algorithm for visualization of frequently changed patterns.

This study contributes to knowledge in terms of the conceptualization of a big data analytics framework that is based on the unique behavior of animals (namely kestrels and dung beetles) and the half-life of trails to develop a search method. The search algorithm is used to find interesting patterns (within both numeric value and time dimensions) and for the visualization of frequently changed patterns/items in large datasets. The advantage of this search algorithm is its ability to self-tune its parameter(s) and select optimal or near-optimal results.

1.9 Outline of the thesis

The thesis is structured as follows:

Chapter 1: General introduction and summary. This chapter presents the general objective of the study, the specific objectives, the research questions, the problem statement, justification of the study, the significance of the study and structure of thesis.

Chapter 2: Literature review. This chapter reviews the current relevant literature and theoretical underpinning of big data analytics frameworks, data cleansing, data mining and data visualization, in order to identify the current challenges and gaps in literature that need to be filled. Additionally, the review also finds innovative ways to address the challenges in aspects of big data analytics (that is, volume, velocity and value) within the context of data cleansing, data mining and data visualization of patterns that might be interesting and that may lead to an action being taken by a user of a big data analytics platform.

Chapter 3: Methodology. This chapter presents the different methods of data cleansing, data mining and data visualization. A bio-inspired method based on selected behavior of chosen animals was mathematical modelled in order to address the gap in frequency frameworks and to propose innovative ways to address the challenges in big data analytics frameworks. Thus, data cleansing, data mining and data visualization form the three phases that constitute the structure of the proposed model for disclosing interesting patterns.

Chapter 4: Developing, testing and evaluating data cleansing. The chapter demonstrates different algorithms for extrapolation of missing values, duplicate text detection with results compared with related algorithms, and selection of relevant features in subset. During the implementation, mathematical expressions were written and translated into an algorithm for testing. Test data was used by the algorithm, and different parameters were used to fine-tune the outcome of the algorithm to provide best results/optimal solutions.

Chapter 5: Developing, testing and evaluating data mining. The chapter presents different bio-inspired algorithms, namely ACO, PSO, BAT and WSA-MP, and compares the results from each algorithm with the proposed algorithm for data mining. During the testing of the newly proposed algorithm, different parameters were fine-tuned in order to generate the best possible solution. The results of data mining were tabulated for each comparative algorithm.

Chapter 6: Developing, testing and evaluating data visualization. The chapter presents the method for data visualization based on the bio-inspired behavior of dung beetles. The dung beetle algorithm (DBA) was tested with different related algorithms, namely the Bee algorithm and ACO, for data visualization. Different parameters were applied to test the output of the algorithm on different datasets. The results of optimal value are tabulated and presented in two-dimension graphs.

Chapter 7: Discussion, challenges and conclusions. The chapter presents how research questions were addressed, discusses experimental results, and outlines challenges of the proposed model. Conclusions are drawn and future work proposed.

1.10 Definition of terms

Meta-heuristic search method is a general algorithmic framework that is applied to different optimization problems with relatively few modifications (both on an algorithm and its parameters) so that the algorithm can be adapted to a specific problem (Iglesia and Reynolds 2005).

Flocking is the phenomenon where individuals all move with approximately the same velocity, so that they remain together as a group (Sinkovits 2006).

Pattern interestingness is defined as follows: it is (1) easily understood by humans, (2) valid on new or test data with some degree of certainty, (3) potentially useful and (4) novel. Furthermore, a pattern is interesting if it validates a hypothesis that the user sought to confirm.

Frequent itemset refers to a set of items that frequently appear together in a dataset (Han and Kamber 2006).

Data mining is the process of finding hidden and complex relationships present in data so as to help businesses discover patterns for future use (Sumathi and Sivanandam 2006).

Big data analytics (or big data mining) is the discovery of actionable knowledge patterns from quality data (Wu, Buyya and Ramamohanarao 2016).

Volume is the amount/size of data that has to be processed. The size of this big amount of data ranges from thousands of terabytes to petabytes and exabytes (Devakunchari 2014).

Veracity is referred to in this research as the accuracy of results from a processing system (Garcia, Luengo and Herrera 2015).

Value relates to what the user will gain or the benefit from the analysis results (such as new revenue opportunities, effective marketing strategies, better customer service strategies and competitive advantage over rival businesses). Devakunchari (2014) refers to value as a measure of the usefulness of data in making decisions.

Variety is the different kinds of data being generated, such as structured or unstructured data (including unstructured text such as word documents, email messages, transcripts of call center interactions, posts from blogs and social media sites; images; audio; video files; and machine data such as log files from websites, servers, networks and applications from mobile systems) (Rouse 2018; Laney 2001).

Velocity is how fast incoming data is created, processed and updated and how quickly the user of information needs results from the processing system (Longbottom and Bamforth 2013).

Data visualization is the process of presenting data in pictorial or graphical format to help in the display of interesting patterns (Bikakis 2018).

Agent-based search algorithm refers to the proposed algorithms (namely kestrel-based and dung beetle-based).

Phase-based framework refers to the proposed methodological framework consisting of phases and corresponding agent-based search algorithm.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

This chapter reviews current literature on big data analytics frameworks, data cleansing, data mining algorithms and data visualization methods in line with the specific objectives of this study. Although this study is largely on bio-inspired/meta-heuristic approaches in analyzing data, other non-bio-inspired approaches to analyzing data are explored. The review of literature is significant for identifying gaps in literature that need to be filled regarding interestingness of frequently changed data.

2.2 Big data analytics framework

Big data arises when a large volume of data is produced and then collected from multiple autonomous data sources, without central control, (Banupriya and Vijayadeepa 2015); big companies quickly realized the value of analysis of this data for various spheres of their operation (Devakunchari 2014). The data coming from these different sources is characterized as having five Vs, as indicated by Longbottom and Bamforth (2013) and Laney (2001), although these characteristics may vary. These characteristics are used to describe big data analytics frameworks.

Wu, Buyya and Ramamohanarao (2016) define big data analytics (or big data mining) as the discovery of actionable knowledge patterns from quality data. Quality is defined in terms of accuracy, completeness and consistency of patterns (Garcia, Luengo and Herrera 2015). Wu, Zhu, Wu and Ding (2014) indicate that big data analytics is a quality improvement process where data is preprocessed and classified into a uniform format that could be useful in decision making. Several analysis frameworks and methods have been proposed to help with the quality improvement process, as presented by Tsai *et al.* (2015) and presented here in Table 2.1 as follows:

Table 2.1 Big data analysis frameworks and methods

Perspective	Methods or platform/ framework	Author(s)	Year	Goal of the study	Taxonomy
Analysis framework	DOT	Huai *et al.*	2011	Add more computation resources via a scale-out solution	Framework
	Generalized linear aggregates distributed engine (GLADE)	Rusu and Dobra	2011	Multi-level tree-based system architecture	
	Starfish	Wonner *et al.*	2012	Self-turning analytics system	
	ODT-MDC	Laurila *et al.*	2012	Privacy issues	
	MRAM	Essa, Attiya and El-Sayed	2013	Mobile agent technologies	
	CBDMASP	Ye *et al.*	2013	Statistical computation and data mining approaches	
	Service-oriented decision support system (SODSS)	Demirkan and Delen	2013	Decision support system issues	
	Big data architecture framework (BDAF)	Wonner *et al.*	2012	Data centric architecture	
	HACE	X. Wu, Zhu, G.-Q. Wu and Ding	2014	Data mining approaches	
	Hadoop	Apache Hadoop	2015	Parallel computing platform	Platform
	Storm	Apache Storm	2015	Parallel computing platform	
	Pregel	Malewicz *et al.*	2010	Large-scale graph data analysis	
	MLPACK	Curtin *et al.*	2013	Scalable machine learning library	Machine learning

	Mahout	Apache Mahout	2015	Machine learning algorithms	
	MLAS	Bu *et al*.	2012	Machine learning algorithms	
	PIMRU	Bu *et al*.	2012	Machine learning algorithms	
	Radoop	Radoop	2015	Data analytics, machine learning algorithms and R statistical tool	

Source: Tsai *et al*. (2015).

In Table 2.1, the “Perspective” column explains the focus of the analysis framework perspective, which indicates the current research focus, namely the use of machine learning for analysis to help guarantee accurate results. The study by Tsai *et al*. (2015) on these analysis frameworks indicates that the key advantage of machine learning as search algorithm is the ability to reduce redundant computational cost. However, within the current dispensation of big data, which is still in its early stages (Tsai *et al*. 2015; Nolan 1979), performance of algorithms may still be an issue/gap that can be looked at as part of the quality improvement process of big data analytics frameworks. Although, the quality improvement process, analysis framework “Perspective” and “Methods or platform/ framework” in Table 2.1 is not the focus of this thesis, it is presented to show the available frameworks, which are not mainly bio-inspired frameworks. Hence, a bio-inspired framework will be the main contribution of this thesis.

In the next subsections, the various phases of data preprocessing (such as cleansing, integrating, transforming and reduction) (Srivastava 2014) are discussed.

2.2.1. Phase 1: Data cleansing

The concept of data cleansing relates to the detection and removal of errors (Mong *et al*. 2002) in raw data to avoid inconsistency (that is, when data items referring to the same object contradict each other) that possibly reduces the quality of data analysis results with limited attention to volume of big datasets. In some instances, when raw data is extracted from data repositories, it could contain errors (such as incorrect spelling of words or, in terms of numeric value, there could be decimal errors) or incomplete data (such as missing data, null values or no data stored for a current observed attribute) (Rahm and Do 2000; Elmagarmid, Ipeirotis and Verykios 2007). The following subsections examine the stages that constitute the data cleansing phase, namely extrapolating missing data values, identifying and removing duplicate text, and selecting relevant feature subsets.

2.2.1.1 Stage 1: Extrapolating missing data values

Missing data occurs when some values of variables are not stored in a dataset. Estimating the missing values is an important step in the data cleansing phase of a big data analytics approach. Narang (2013) describes missing data as data that exists in the real world but was not provided by a user. Narang's (2013) method to address the missing data is to interpolate approximate values as missing data points from historical data (that is, real-time stock trading datasets) with incorrect timestamps. Agbehadji *et al*. (2018) indicate that one of the reasons for missing data is non-response or omitted entries, which may relate to optional attributes. There are three categories of missing data: data missing completely at random (MCAR), missing at random (MAR) and missing not at random (MNAR) (Acock 2005; Nelwamondo, Mohamed and Marwala 2007), all of which require different methods of handling the missing data. Moreover, the missing data may be a missing text and/or missing value at random.

The missing data category of MCAR occurs when the missing values are randomly distributed throughout a matrix such that a missing value in a row of a matrix is not dependent on any other row entry in a dataset (Acock 2005). In other words, neither the row entry, which is missing, nor any other row entry can predict whether a value is missing. When this happens the chances of the data being detected as missing are not dependent on either the missing or the complete value in the same row entry of a matrix. The list-wise method to handle MCAR is ideally used to remove all data that has one or more missing cases. However, by this removal, a problem is created in that the missing values produce both biased parameters and incorrect estimates in analysis (Rubin 1977; Little and Rubin 1989). The pairwise method is another method of handling MCAR. This method seeks to address the missing value problem by computing the covariance estimates from all samples of cases observed on relevant variables. The pairwise deletion method assumes that all data is completely missing at random. Therefore, variables with missing data are then deleted during computation. This deletion could cause error in computation because each element in the covariance matrix may have a different group of attributes (Kline 1998; Carter 2006; Rubin *et al*. 2007).

The MAR category occurs when the missing value in a row of a matrix depends on another known row entry in a dataset (Rubin *et al*. 2007). Due to dependency, the missing value can be predicted from a previously known value in a dataset. Thus, the missing value is dependent on the previously known value. When this happens, it becomes easy to trace a pattern of missing values in a row of a matrix through the traditional approach to handling MAR: the pairwise deletion method, as described previously.

The MNAR (also known as non-ignorable nonresponse) category occurs when the missing value in a row of a matrix depends on the other missing values in the row entry (Rubin *et al*. 2007). Due to multiple dependencies, the known data cannot be used to estimate the missing value. Thus, the chances that the value in question is detected as missing is dependent on the detection of previous missing values.

The traditional approaches to handling missing data are, however, not efficient at providing best optimal estimates for missing values. These approaches include list-wise deletion or case deletion, pairwise deletion and sample mean substitution (that is, *k*-NN and *k*-Means clustering) (Quinlan 1989; Acock 2005; Rubin *et al*. 2007).

The *sample mean of class* method replaces the missing values with the group mean of all known values of the attribute. The mean of each group represents a target class with attribute can belong. For instance, assume $x^{j}_{missing,i}$ is the j^{th} missing attribute of the i^{th} instance of the m^{th} class (Sim, Lee and Kwon 2015)

$$x^{j}_{(missing,i)} = \sum\nolimits_{k \in I(mth\ class\ incomplete)} \frac{x^{j}_{(missing,k)}}{n_{|\ I(\ mth\ class\ incomplete)|}} \qquad \textit{Equation 2.1}$$

where (m^{th} class incomplete) is a set of indices that are not missing in $x^{j}_{(missing,i)}$, and $n_{|\ I(\ mth\ class\ incomplete)}$ is the total number of instances where the j^{th} attribute of the mth class is not missing.

The *k-nearest neighbor (k-NN)* method searches for attributes among non-missing attributes using the *k*$^{\text{th}}$-NN method (Sim *et al*. 2015). This method imputes missing values based on the values of the attributes of the k most similar instances, as expressed in Equation 2.2:

$$x_{i}^{j} = \sum\nolimits_{P \in k-NN(x_i)} k\left(x_{i}^{I(complete)}, x_{P}^{I(complete)}\right) * x_{P}^{j} \qquad \textit{Equation 2.2}$$

where *k*-NN (*x_i*) represents the index set of the *k*$^{\text{th}}$ nearest neighbors of x_i based on the non-missing attributes, *k* (which is by default equal to 4) represents a number that decides how many neighbors (where neighbors is defined based on the distance metric) are considered in a cluster, and $k(x_i, x_j)$ is referred to as a kernel function that is proportional to the similarity between the two instances x_i and x_j. The similarity function computes an approximate value, which is used to indicate that two instances are the same. Thus, the smaller the similarity value between two instances, the more similar the two instances. Thirumuruganathan (2010) indicates that the *k*-NN method is non-parametric, meaning the *k*-NN algorithm does not make any underlying assumption on the distribution of data. However, the *k*-NN method makes a decision of the approximated value based on the entire training dataset.

The *k-Means clustering* method (Hartigan and Wong 1979) partitions an entire dataset *M* with several dimensions *N* into *k* clusters so that within each cluster of missing values, the sum of squares is minimized against all partitions. This means, in the context of missing values, that these partitions are categorized as clusters of the same attributes of missing values from other attributes with non-missing values. Based on these attributes, the algorithm based on *k*-Means finds missing values that have to be imputed (Sim *et al*. 2015). This *k*-cluster then forms two different clusters, namely homogeneous and heterogeneous clusters. The challenge with a *k*-cluster is that when there is large volume of data, it leads to a large number of clusters and the computational time thus increases (Hartigan and Wong 1979). The *k*-cluster is expressed as:

$$arg \min_{c^{h(complete)}} \sum_{i=1}^{k} \sum_{(x_j^{I(complete)} \in C_i^{h(complete)})} || x_j^{I(complete)} - C_i^{h(complete)} ||^2$$

Equation 2.3

where $C_i^{I(complete)}$ represents the centroid, and the union of all cluster is represented as $C_i^{I(complete)} = C_1^{I(complete)} \cup \ldots \cup C_i^{I(complete)}$). For a missing value x_i^j, the mean value of the attribute for the instances in the same cluster with $x_i^{I(complete)}$ is imputed as

$$x_i^j = \frac{1}{|C_i^{I(complete)}|} * \sum_{(x_P^{I(complete)} \in C_k^{I(complete)})} x_P^j$$ *Equation 2.4*

subject to: $k = \arg\min_i |x_j^{I(complete)} - C_i^{I(complete)}|$

where $x_j^{I(complete)}$ represents the value of an instance/attribute, and $C_i^{I(complete)}$ represents the centroid.

The approach of Grzymala-Busse *et al.* (2005) to addressing missing data is to use the same attributes from similar cases to determine the approximate value of missing attributes through a closest fit algorithm. Based on the closest fit algorithm, the proximity between cases (such as case *x* and *y*) are calculated using the Manhattan distance, formulated as:

$$distance(x, y) = \sum_{i=1}^{n} distance(x_i, y_i)$$ *Equation 2.5*

where:

$$distance(x, y) = \begin{cases} 0 & if\ x = y \\ 1 & if\ x\ and\ y\ are\ symbolic\ and\ x \neq y, or\ x = ?\ or\ y = ? \\ \dfrac{|x - y|}{r} & if\ x\ and\ y\ are\ numbers\ and\ x \neq y \end{cases}$$

where *r* represents the differences between the maximum and minimum of the unknown values of numerical attributes with a missing value. A similar approach based on cases is the use of the hot-deck method (Sim *et al*. 2015), which is formulated as:

$$x_i^j = x_k^j, k = arg \min_P \sqrt{\sum_{j \in I\,(complete)} sd_j \left(x_i^j - x_P^j\right)^2} \qquad \textit{Equation 2.6}$$

where sd_j represents the standard deviation of the j^{th} attribute that is not missing, and x_i^j and x_P^j are two different attribute values. The challenge of both closest fit and hot-deck is non-similar cases are not considered.

In summary, the sample mean substitution method requires that each data point clustered around a centroid needs computation to find the best estimates. Thus, the number of clusters, the number of data points and the dimensions involved to compute missing values make it inefficient. In pairwise deletion, since the method assumes all data is missing at random, it uses the average sample size to estimate its standard error, which either results in underestimation or overestimation of the standard error in the analysis of missing values, and this makes it inefficient. The computational time may be a challenge in finding missing data when there is a large volume of data.

Other efficient methods have been proposed to handle MAR. They are the maximum likelihood (Allison 2012) and multiple imputation method (for MAR) (Lakshminarayan, Harp and Samad 1999), the Expectation-Maximization algorithm (Zhao, MacKinnon and Gallup 2005; Acock 2005), dynamic programming (Bellman 1957) machine learning approaches (such as autoencoder neural networks) (Bishop 1995), meta-heuristic algorithms (such as genetic algorithms (GA)) (Goldberg 1986), the Firefly algorithm (Yang 2010) and the Wolf algorithm (Tang *et al*. 2012). Other algorithms that combine

GA and machine learning approaches include GAs and auto-associative neural networks (Abdella and Marwala 2006; Goldberg 1986). The advantage of applying meta-heuristic algorithms to missing value approximation is the ability to escape from local optima by using randomization to help reduce high computation cost.

The *maximum likelihood* method is a statistical method of estimating missing values based on likelihood of independent observation (Allison 2012). The process of estimation starts with the formulation of a likelihood function, which is expressed in terms of a probability of observed data and the missing value. Parameters are used in the function and when a parameter is assumed as true, it must maximize the probability of observed value. This is expressed in terms of likelihood in Equation 2.7 as:

$$L(\theta|Y_{observed}) = \int f\left(Y_{observed}, Y_{missing}|\theta\right) dY_{missing} \qquad \textit{Equation 2.7}$$

where $Y_{observed}$ represents the observed data, $Y_{missing}$ is the missing data and Θ is the parameter of interest to be estimated (Little and Rubin 1987). Thus, the likelihood function is expressed in Equation 2.8 as:

$$L(\theta) = \prod_{i=1}^{n} f(y_i|\theta) \qquad \textit{Equation 2.8}$$

where *f(y|ϴ)* is the joint probability or probability density function of the observation *y*, while θ is the set of parameters that has to be estimated given *n* number of independent observations (Allison 2012). The maximum likelihood estimate is obtained by finding the value of θ, which then maximizes the likelihood function. The parameter θ is further expressed as a vector to indicate the variance and the mean of data distribution as:

$$\theta = (m, \sigma^2)^T \qquad \textit{Equation 2.9}$$

where sigma (σ^2) is a parameter that represents the variance and *m* is the mean. The variance is further expressed in Equation 2.10 as:

$$\sigma^2(x) = \frac{1}{2\,N} \sum_i^N {x_i}^2 \qquad \textit{Equation 2.10}$$

where *N* is total independent observation on data *x* (x_1, x_2, …, x_k). The variable *x* is the sample of data being considered.

The estimated parameter $\hat{\theta}$ is expressed in Equation 2.11 as:

$$\hat{\theta} = arg \max_{\sigma^2 \in R^+, m \in R} Ln(\theta : x) \qquad \textit{Equation 2.11}$$

Thus,

$$Ln(\theta; x) = -\frac{N}{2} In(\sigma^2) - \frac{N}{2} In(2\pi) - \frac{1}{2\sigma^2} \sum_{i=1}^{N} (x_i - m)^2 \qquad \textit{Equation 2.12}$$

Given *n* independent observation on *k* variables (*y_1, y_2, …, y_k*) with no missing data, the likelihood function is expressed in Equation 2.13 as

$$L = \prod_{i=1}^{n} f(y_{i1}, y_{i2}, \dots, y_{ik}\ ; \theta) \qquad \textit{Equation 2.13}$$

When data is missing for individual observation *i* for y_1 and y_2, then the likelihood for the individual is expressed as the probability of observing the other remaining variables in the dataset such as y_3, …, y_k. There are two instances of individual observation on data: either data is discrete, or it is continuous. This means that if y_1 and y_2 are discrete, then the joint probability is the summation of all possible values of the two variables that have the missing values in the dataset. This joint probability is expressed in Equation 2.14 as:

$$f_i^*(y_{i3}, \dots, y_{ik}\ ; \theta) = \sum_{y1} \sum_{y2} f_i(y_{i1}, \dots, y_{ik}\ ; \theta) \qquad \textit{Equation 2.14}$$

Again, if the missing variables are continuous, then the joint probability is the integral of all possible values of the two variables that have the missing values in the dataset. This joint probability is expressed in Equation 2.15 as:

$$f_i^*(y_{i3}, \dots, y_{ik}\ ; \theta) = \int_{y1} \int_{y2} f_i(y_{i1}, y_{i2}\ \dots, y_{ik}) dy_2 dy_1 \qquad \textit{Equation 2.15}$$

Moreover, when each observation contributes to finding the likelihood function, then the summation or integral is performed over the missing values in the dataset. Then the overall likelihood is the product of all observations. For instance, if there are *x* observations with complete data and *n-x* observations with data missing on y_1 and y_2, the likelihood function for the full dataset is expressed in Equation 2.16 as:

$$L = \prod_{i=1}^{x} f(y_{i1}, y_{i2}, \dots, y_{ik}\ ; \theta) \prod_{x+1}^{n} f_i^*(y_{i3}, \dots, y_{ik}\ ; \theta) \qquad \textit{Equation 2.16}$$

The maximum likelihood estimation of missing values produce estimates that are consistent (that is, for a given large dataset, it produces the same or approximately unbiased results), asymptotically efficient (meaning there is minimum sample variance, which indicates a high level of efficiency in the missing value dataset) and asymptotically normal (Allison 2012).

The expectation and maximization method learns from input data (particularly for unlabeled data) by iteratively approximating parameters (Zhao, MacKinnon and Gallup 2005) to indicate the close relationship between missing and observed data. This expectation and maximization method finds conditional expectation until a convergence (that is, most likely value) is reached on the missing data using the observed data and estimated parameters (Little and Rubin 1987). The distribution of a complete dataset Y is expressed using the function

$$f(Y|\theta) = f\left(Y_{observed}, Y_{missing}|\theta\right) = f(Y_{observed}|\theta)f(Y_{missing}\,|Y_{observed}, \theta)$$

Equation 2.17

where $f\left(Y_{observed}, Y_{missing}|\theta\right)$ represents the probability density of the observed data and $f(Y_{missing}\,|Y_{observed}, \theta)$ is the probability density of the missing data in a given dataset. The log-likelihood is then expressed in Equation 2.18 as

$$L(\theta|Y) = L\left(\theta|Y_{observed}, Y_{missing}\,\right) = L(\theta|Y_{observed}) + In(f\left(Y_{missing}\middle|\, Y_{observed}, \theta\right))$$

Equation 2.18

where θ represents the parameter of interest, which is controlled iteratively in two steps such that an optimized result is produced. The steps are Expectation (E) and Maximization (M) and are expressed as follows:

Expectation step:

This step indicates the expected log-likelihood of the data such that the parameter is the true current estimate θ^t. The expected estimate is expressed in Equation 2.19 as:

$$EM\left(\theta\middle|\theta^{(t)}\right) = \int L(\theta|Y)f\left(Y_{missing}\,\middle|Y_{observed}, \theta\right) = \theta^t dY_{missing}$$ *Equation 2.19*

Maximization step:

This step finds $\theta^{(t+1)}$ by maximizing the expectation, which is expressed as follows:

$$EM\left(\theta^{(t+1)}\middle|\theta^{(t)}\right) \geq EM\left(\theta\middle|\theta^{(t)}\right), \forall\theta \qquad \textit{Equation 2.20}$$

The *dynamic programming method* is based on step-wise calculation such that each reoccurring value is tracked to avoid recalculation and obtain an optimum result as a sequence of decisions. This concept of dynamic programming, when applied to missing values, allows the separation of a large problem space into sub-problems such that missing data in each sub-problem can be calculated until the best optimal estimate of missing data is obtained (Bellman 1957). This approach avoids repeating the calculation of missing values when similar problems are encountered by storing the results from each sub-problem. The formulation for dynamic programming can be expressed in Equation 2.21 as follows (Bellman 1957):

$$J(t) = \sum_{k=0}^{int} \gamma^k U(t+k) \qquad \textit{Equation 2.21}$$

where γ is the discount factor where $0 < \gamma < 1$, and U is the utility function. Dynamic programming is viewed as an optimization problem aimed at minimizing objective function for the best optimal estimates. This optimization problem is formulated (Bertsekas 2005) in equation 2.22 as follows:

$$x_{k+1} = f_k(x_k, U_k, r_k),\ k = 0,1,\dots,N-1 \qquad \textit{Equation 2.22}$$

where k represents the discrete time, x_k represents the observed data (known data), U_k is the sequence of decisions to be made from the known data, N is the number of times a control parameter is applied, and r_k is the error introduced in making a decision. When a cost function *g()* is added to the formation (Bertsekas 2005), it is then expressed in Equation 2.23 by:

$$E\{g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, U_k, r_k)\} \qquad \textit{Equation 2.23}$$

where *g()* is the cost function. The assumption is that U_k is selected based on the knowledge of x_k for the model to be feasible. Thus,

$$\left|\left|E\right|\right| = \sum_t \left(J[Y(t)] - \gamma J[Y(t+1)] - U(t)\right)^2 \qquad \textit{Equation 2.24}$$

where *Y(t)* is an observed vector, γ is the discount factor where 0 < γ < 1, and *U* is the utility function.

The *autoencoder neural network method* or *auto-associative multi-layer perceptron method* for missing data estimation identifies approximate parameters given a sparse representation of input space as a result of the cross-coupling of hidden units (Marwala 2006). The architecture of a multi-layer perceptron is such that mathematical functions are used to establish a relationship between input spaces through the hidden units to output space. The tangent basis function is used in the hidden units, while linear functions are used for output space. The non-linear mathematical relation is applied to map the output *y* (which is the estimated parameter or weight) and the input *x* in the neural network. It is expressed in Equation 2.25 (Bishop 1995) as

$$y_k = f_{outer}\left(\sum_{j=1}^{M} w_{kj}^{(2)} f_{inner}\ (\sum_{i=1}^{d} w_{ji}^{(1)} x_i + w_{j0}^{(1)}) + w_{k0}^{(2)}\right) \qquad \textit{Equation 2.25}$$

where $w_{ji}^{(1)}$ and $w_{kj}^{(2)}$ represent the respective weights in the first and second layers. For instance, moved from input *i* to hidden unit *j* or the output unit *k, M* is the number of hidden units, and *d* is the number of output units, while $w_{j0}^{(1)}$ represents the bias for the hidden unit *j*, and $w_{k0}^{(2)}$ represents the biases for the output unit *k*. The $f_{outer}(*)$ represents a logistic function, while $f_{inner}$is the hyperbolic tangent function.

Whereas f_{outer} is expressed in Equation 2.26 as

$$f_{outer}(v) = \frac{1}{1+e^{-v}} \qquad \textit{Equation 2.26}$$

f_{inner} is expressed in Equation 2.27 as

$$f_{inner}(v) = tan\,h(v) \qquad \textit{Equation 2.27}$$

The bias parameters in first layer are weights from extra input with a fixed value of $x_0 = 1$, while the bias parameter in the second layer are weights from an extra hidden unit with activation, which is fixed at $z_0 = 1$.

The weight parameter in the neural network is approximated using Bayesian methods. Given a dataset in matrix form, the weights are adjusted based on a probability function to find an approximate weight to represent the missing data. This function is expressed in Equation 2.28 as

$$P(\{w\}|D) = \frac{P(D|\{w\})P(\{w\})}{P(D)} \qquad \textit{Equation 2.28}$$

where *P(w|D)* is the probability distribution of weights called the posterior probability of *{w}* given dataset *D*. *P(D|{w})* is the likelihood function, that is, the conditional probability that shows the occurrence of *D* given *{w}*. *P({w})* is the prior probability of occurrence of *{w}*, independent of whether *{w}* occurs or not. *P(D)* represents the evidence and represents the normalized posterior probability distribution function.

The matrix dataset is expressed in Equation 2.29 as:

$$|D| = (\, x_P, \dots, x_N, y_P, \dots, y_N) \qquad \textit{Equation 2.29}$$

where the vector {x} is the input vector and the vector {y} is the output vector. Since the model is an auto-associative neural network, the assumption is that the input vector {x} is the same as the output vector {y}. The posterior probability distribution function of the weight from given matrix data can be formulated using the Gibbs distribution, given the input data (Marwala 2007).

The probability distribution of weight, given the input data, may be expanded in terms of the likelihood function, as in Equation 2.30:

$$P(\{w\}|\,[D]) = \frac{1}{Z_s}\, exp\,(-\beta \sum_{n=1}^{N} \sum_{k=1}^{K} \{\, t_{nk} - y_{nk}\}^2 - \frac{\alpha}{2} \sum_{j}^{W} w_j^2) \qquad \textit{Equation 2.30}$$

The first term in the formulation represents the likelihood function, that is, $\frac{1}{\mathrm{Z_s}} \exp\,(-\beta \sum_{\mathrm{n=1}}^{\mathrm{N}} \sum_{\mathrm{k=1}}^{\mathrm{K}} \{\, \mathrm{t_{nk}} - \mathrm{y_{nk}}\}^2$. Data contributing to error is represented by β (expressed using sum of square error), *n* represents the index for training patterns, t_{nk} is the observed output, y_{nk} is the estimated output, and *k* is the index for the output units (Bishop 1995). The second term, $\frac{\alpha}{2} \sum_{\mathrm{j}}^{\mathrm{W}} \mathrm{w_j^2}$, represents the prior information, which is the generalization parameter (or weight decay) for smooth mapping of function in the training process such that large weight magnitudes are penalized. The coefficient α represents the training error with the assumption that if α is a high value, then the generalization parameter over-smooths the weights (that is, in the neural network), thereby producing inaccurate results. Meanwhile, with a lower α value, the effect of the generalization parameter is negligible, so a stopping criterion can be defined to reduce the computational complexity of the model.

Moreover, Z_s is further expressed in Equation 2.31 and Equation 2.32 as:

$$Z_s(\alpha,\beta) = \int exp(-\beta \sum_{n=1}^{N} \sum_{k=1}^{K} \{ t_{nk} - y_{nk}\}^2 - \frac{\alpha}{2} \sum_{j}^{W} w_j^2) dw \qquad \textit{Equation 2.31}$$

$$= (\frac{2\pi}{\beta})^{N/2} + (\frac{2\pi}{\alpha})^{W/2} \qquad \textit{Equation 2.32}$$

The optimal weight vector corresponds to the maximum likelihood. The number of layers in the hidden unit is chosen on a trial-and-error basis. A small number of layers may introduce insufficient flexibility, leading to poor generalization because of high bias, while a large number of layers may introduce unnecessary flexibility, leading to poor generalization or over-fitting caused by high variance.

The advantage of the Bayesian approach is that it penalizes complex models by selecting the optimal model without using independent models (such as cross-validation). Additionally, it gives reliable output estimates of missing values in a dataset (Marwala 2006).

A *genetic algorithm* (GA) is an evolutionary approach on survival of the fittest. This survival depends on the mechanism of "natural selection" (Darwin 1868 cited in Agbehadji 2011), where species considered weak and unable to adapt to the conditions of the habitat are eliminated, while species considered strong and able to adapt to the habitat survive. Thus, natural selection is based on the notion that strong species have a greater chance of passing on their genes to future generations, while weaker species are eliminated by natural selection. Sometimes random changes occur in genes due to changes in species' external environments, which will cause new future species that are produced to inherit different genetic characteristics. Thus, successive generations are able to adapt to the habitat in respect of time.

The terminology used in GAs to refer to population members is "string" or "chromosomes". These chromosomes are made of discrete units called genes (Railean *et al*. 2013), which are binary representations such as 0 and 1. A GA is an adaptive search procedure (Agbehadji 2011) that depicts the mechanism of natural selection of populations. There are rules to steer the combination of parents to form children. These rules are referred as operators, namely crossover, mutation and selection methods. The notion of crossover consists of interchanging solution values of particular variables, while mutations consists of random value changes to a single parent. The children produced by the mating of parents are tested, and only children that pass the survival test are then chosen as parents for the next generation. The survival test acts as a filter for selecting the best species.

The computational approach to depicting the mechanism of natural selection or search procedure starts with an initial guess and attempts to improve the guess through evolution (Agbehadji 2011) by comparing the fitness of the initial generation of the population with the fitness obtained after application of operators to the current population, until the final optimal value is produced.

Another example of an algorithm is *Firefly algorithm* (Yang 2008). Firefly algorithm is modelled on the behavior of fireflies, specifically their ability to produce short and rhythmic flashing light to communicate with other fireflies. The flashing light is used to attract mating partners and potential prey, and it serves as warning mechanism. The firefly signaling system consists of rhythmic flash, frequency of flashing light and time period of flashing (Yang 2010). This signaling system is controlled by simplified basic rules underlying the randomness of behavior of fireflies, which can be summarized as follows: one firefly will be attracted to the other; attractiveness is proportional to brightness; and brightness is affected by landscape.

The attraction formulation is based on the following assumptions:

a. Each firefly draws any other fireflies with weaker flashes.
b. This draw is based on the brightness of the firefly's flash, which is inversely proportional to proximity to each other.
c. The firefly with the brightest flash is not attracted to any other firefly and its fligh t is random.

Fireflies follow the genetic/meta-heuristic approach of an initial randomization of individual fireflies in space, after which the brighter fireflies attract those closest to them. The fireflies whose flashes fall below a given threshold are then removed from the population, and the brightest fireflies form the next generation. The generations/iterations continue until a select criterion is reached or until a maximum number of generations is reached.

The firefly's flashes may also be used to extrapolate missing values. The variation of light intensity and the attractiveness are two major aspects of the firefly. Regarding these two aspects, attractiveness (β) is indicated as proportional to light intensity (γ), which is seen by other fireflies. Thus, the formula for the attractiveness between two fireflies is given in Equation 2.33 as follows:

$$\beta(r) = \beta_o e^{-\gamma r^2} \qquad \textit{Equation 2.33}$$

where β_o denotes the initial attractiveness, r denotes the Euclidean distance between two fireflies $x_j \ and \ x_i$., and attractiveness (β) is denoted as proportional to light intensity (γ), which is seen by other fireflies. The variation of attractiveness is calculated by γ, which affects firefly behavior and convergence.

The distance between two fireflies is calculated using the Euclidean distance, which is expressed in Equation 2.34 as:

$$r = \sqrt{\sum_{i=1}^{n} \left(x_j - x_i\right)^2} \qquad \textit{Equation 2.34}$$

where *n* indicates the total number of fireflies. If firefly x_i is attracted to brighter firefly x_j, it is denoted as in Equation 2.35:

$$x_{t+1}^{k} = x_t^k + \beta\left(x_j - x_i\right) + \alpha\left(rand - \frac{1}{2}\right) \qquad \textit{Equation 2.35}$$

where $\beta\left(x_j - x_i\right)$ indicates the attractiveness between the two fireflies x_i and x_j, $\alpha\left(rand - \frac{1}{2}\right)$ indicates the randomization, α is a randomization parameter (used to control the random movements of a firefly if there are no other fireflies) that governs the step length, and *rand* is a random number generator that produces random numbers from 0 to 1 (Leke and Marwala 2016). Similarly, x_j represents the observed data, and x_i is the estimated value. The optimized value obtained after several iterations is then used to estimate missing data in a dataset.

Another example of an algorithm is *Wolf algorithm* (Tang *et al*. 2012). Wolf algorithm is a meta-heuristic algorithm based on social animals that hunt for prey and watch out for predators in their habitat. Wolves have semi-cooperative characteristics that allow them to move in a loosely coupled group but hunt their prey individually. This natural behavior makes each wolf find its best position and then continuously move to a global best position and watch out for predators during hunting (Tang *et al*. 2012; Agbehadji, Millham and Fong 2016). The preying behavior of wolves is as a search for the best possible position for a successful hunt. The four basic rules formulated from the preying behavior of wolves are: preying initiatively, preying passively, escape from predators and scent marks.

During preying initiatively, a step is used to check the visual perimeter, and this indicates whether the wolf should change its current position with the highest value or after a random step in a random walk. Preferably a wolf moves to the highest best position within its visual perimeter. This is expressed in Equation 2.36 as:

$$x_{t+1}^{k} = x_t^k + \beta\big(x_j - f(x_i)\big) + escape \qquad \textit{Equation 2.36}$$

where x_j represents a peer with better position, x_i is a current position of a wolf, *r* is the distance between the wolf and its peer x_c with the better location within its visual perimeter V_p. The *escape* function calculates a random position to escape with a constraint of minimum length. Step size is less than visual distance. The attractiveness β of wolves to each other is expressed in Equation 2.37 as:

$$\beta = \beta_o e^{-r^2} \qquad \textit{Equation 2.37}$$

where β_o represents initial attractiveness, and *r* denotes the distance. This distance is expressed in Equation 2.38 as:

$$V_p \leq r(x_j, x_c) = (\sum_{k=1}^{n} |x_{j,k} - x_{c,k}|^{\lambda})^{\frac{1}{\lambda}} \qquad \textit{Equation 2.38}$$

where λ is the order (1 or 2).

Each wolf in a best position leaves a scent mark. This scent mark shows the number of times a mark is indicated in a given boundary by each wolf (Agbehadji *et al*. 2016). The mark helps other wolves in the group to note best positions that were marked by their peers in the search space. The higher the frequency, the more attractive the position, which leads to a successful hunt. Scent mark is formulated in Equation 2.39 as:

$$S_m = x(i) * \sum_{i=0}^{n} f \qquad \textit{Equation 2.39}$$

When preying passively, a wolf does not find food or a better position where its group members are from the previous step. It then positions itself in alert mode so as to move to a better position in comparison with the position of its peers in the group. This is expressed in Equation 2.40 as:

$$x_{t+1}^{k} = x_t^k + \alpha * r * rand() \qquad \textit{Equation 2.40}$$

where α and *r* are constants that represent velocity and distance, while *rand* is a random value from uniform distribution between 0 to 1.

Escape happens when a wolf detects an incoming predator. It then moves at random from its current position to a new position that is greater than its visual perimeter v_p or distance. Escape is an important step that allows wolves to avoid being stuck in local optima. The escape is expressed in Equation 2.41 as

$$x_{t+1}^{k} = x_t^k + \alpha * s * escape() \qquad \textit{Equation 2.41}$$

where s is step size, which is less than visual distance v_p, α is the velocity, and *escape ()* is a function that randomly generates a new position greater than v_p.

2.2.1.2 Stage 2: Identifying and eliminating duplicate text

The aim of the second stage of the data preprocessing model is to identify, match and eliminate duplicate text from data sources. Elmagarmid *et al*. (2007) define duplicate as when different strings refer to the same real-world entity. Naumann (2013) defines a framework for duplicate detection that is categorized into identity, similarity measure, algorithm used and evaluation aspects.

The aspect of identity with respect to the framework for duplicate detection indicates the source of data (e.g. relational database, XML, etc.) that would be fed into the duplicate detection process (Naumann 2013). The focus of the review is not on whether data comes from single or multiple source, however, but on whether the source of data is authentic so as to ensure features that are defined (for the similarity measure) to correspond to the nature of data being considered for duplicate detection (such as text/string).

The aspect of similarity measure with respect to the framework for duplicate detection compares two strings to find out if the strings are similar or not, hence the use of a string similarity measure to compute the similarity of strings (also known as character-based similarity metrics). A string similarity measure is the use of basic mathematical expressions to compute the distance between two strings (Moere 2004). The significance of this measure is that it shows the quality of strings of letters being compared in terms of the similarity value computed between a pair of strings. During the computation of the distance between a pair of strings, similar strings are assigned a large score while dissimilar strings (that is, strings that are not the same) are assigned a low score. In contrast, the distance measure assigns low weight to similar strings and high weight to dissimilar strings within the range (0, 1). For instance, given a distance measure with a weight in the range (0, 1), the translation to the similarity measure can be performed by using the basic mathematical formulation in Equation 2.42:

$$sim(a,b) = 1 - dist(a,b) \qquad \textit{Equation 2.42}$$

where a, b are strings that are compared. The translation from similarity measure to distance measure is computed in Equation 2.43 as

$$dist(a,b) = 1 - sim(a,b) \qquad \textit{Equation 2.43}$$

where a, b are strings that are compared.

The two methods that can be applied to duplicate detection of words in order to find the similarity measure include the pairwise method and the partition-based method (Matsakis 2010). The pairwise method was developed to find matches between two files or data sources (Matsakis 2010). The steps in computing a similarity measure in the pairwise method (Matsakis 2010) is as follows: First, the similarity function is expressed and used to scale the real-world dataset into a set of matched data. Secondly, a threshold is set for matched data. Mostly, in setting the threshold, the nature of data from the data source determines what threshold to set to define the “sensitive” nature of the data. For instance, data coming from a health institution may require non-disclosure on the kind of illness of a person, so in this context a threshold may be set to show records that are very close to each other in order to suggest who the data refers to and prevent it from being disclosed outside. Third, an algorithm links matched data together by computing a transitive closure of the match items. Afterwards, a binary representation is used to indicate the matched values of paired items or combine multiple attributes into a single value. Once paired items are found, a supervised learning algorithm could be used to build a classifier from labeled items to indicate whether the pair is a match (duplicate items). For instance, consider a problem of duplicate detection in a relational database where multiple library databases with publication and author records are to be merged. A likely way to achieve the match is to declare publication records as co-referent, which implies that related author(s) record(s) would also be co-referent. The co-referent decision enables the matching of related records so as to find duplicates. The limitation of the pairwise method is that it is unable to identify global constraints because it is likely for an entity to have a small number of distinct values for a field (e.g. email address) but unlikely for there to be several (Matsakis 2010).

During duplicate word/text detection, two strings/words are compared and either the first string is correct and the other string is considered as erroneous data that could be cleansed, or each matched string is considered as partial duplicate data that must be merged to produce a complete string (Monge 2000). Two strings are the same or equivalent if they are equal semantically, meaning strings obey the properties of transitivity, symmetry and reflectivity (Monge 2000). The transitive property states that for all real numbers *x*, *y* and *z*, if *x*=*y* and *y*=z, then *x*=z. The symmetric property states that for all real numbers *x* and *y*, if *x*=*y*, then *y*=*x*. The reflexive property states that for every real number *x*, *x*=*x*.

The partition-based method is used when a dataset is large (Naumann 2013). This method assigns a score to a candidate partition based on how close it is to the true partition (Matsakis 2010). There are different approaches for generating scores for each partition, such as the use of the generative Bayesian model and discriminative models. While the Bayesian model uses conditional probability distributions to model uncertainties in unobserved duplicates, discriminative models assign scores to partitions without having strict conditional independence assumptions on the model.

Identifying same real-world objects is achieved by the use of similarity measures in the data preprocessing stage in large datasets. The importance of the similarity measure is that it enables the identification of exact and approximate duplicates of real-world objects (Sauleau, Paumier and Buemi 2005). Also, the similarity measure helps identify misspelled words in large datasets. Several similarity measures that have been proposed, and they can be categorized into the edit-based method, token-based method, domain dependent method and hybrid-based method (Naumann 2013).

The edit-based method uses the distance between two words (in a record), which represents the same real-world object, to find a similarity value to suggest that two words are similar or incorrectly spelt. Examples of algorithms based on edit-based methods (also referred to as character-based methods) are Jaro-Winkler, Smith-Waterman, Hamming and Damerau–Levenshtein.

The token-based method forms words from sequences of characters in a string (words) and assigns different weights to characters. An example of an algorithm based on the token-based method is *n*-gram (i.e. substrings of length *n*) (Cohen, Ravikumar and Fienberg 2003). Bilenko *et al*. (2003) compared the effectiveness of character-based and token-based method used by the Monge-Elkan metric and indicate that algorithms based on the character-based and token-based methods have the highest average performance across datasets and across character-based distance metrics. However, metrics that are robust and show high average performance may perform poorly on different datasets because performance of similarity measures is affected by characteristics such as the length of text, accuracy of spelling, presence of abbreviations, etc. (Gali, Mariescu-Istodor and Fränti 2016).

The hybrid-based method applies both token-based and internal similarity function for tokens. An example of this algorithm is Monge-Elkan (Monge and Elkan 1997) and Soft Frequency-Inverse Document Frequency (TF-IDF). Monge and Elkan’s (1997) algorithm is a variant of the hybrid-based method that takes records as alphanumeric strings and uses the Smith-Waterman algorithm to compute the edit distance between two strings. This algorithm is used in applications where the key field values of records are alphanumeric strings (Tian *et al*. 2002). Although the algorithm is accurate in performing comparisons when there are abbreviations and minor syntactical differences, including typographical mistakes, the running time is proportional to the length of string, which is problematic when there is a high volume of records to be compared (Monge 2000).

The Soft TF-IDF is a hybrid-based method that combines the cosine distance with TF-IDF weighted vectors and the Jaro-Winkler algorithm to compute the distance metric for name-matching, such as first name, middle name and surname (Gali *et al*. 2016). Initially, the Jaro-Winkler algorithm is applied to all pairs of tokens that appear between two strings, after which the TF-IDF measure is applied to tokens that have a similarity score above the threshold ($\theta \geq 0.9$) based on the newly evaluated Jaro-Winkler distance (Gali *et al*. 2016). However, cosine similarity measures may fail to correctly determine similarity if two records are similar because of differences in the representation of characters in words as a result of re-ordering of words or misspelling (Gali *et al*. 2016).

The domain dependent method compares two numerical attributes and calculates their difference in order to find the absolute similarity value to suggest that two numbers are duplicate. Since the present study focuses on words that could be duplicates, more prominence is given to the edit-based method and the token-based method.

The concept of edit distance is based on insertion, deletion or substitution of characters in words (referred as edit operation) (Tian *et al*. 2002). Approximately 80% of all misspelled words (Tian *et al*. 2002) contain a single instance of one of the following types of error: insertion, deletion, substitution and transposition. Thus, edit operations enable the identification of these types of errors. An insertion error occurs when a character is mistakenly inserted in a different position of a word. This makes the length of the mistaken word longer than the equivalent correct word. A deletion error occurs when a character is omitted from a word that makes the length of the mistaken word shorter than the correct word, while a substitution error occurs when a character is erroneously replaced by another character. A transposition error erroneously interchanges the positions of two adjacent characters. Tian *et al*. (2002) indicate that the way to identify erroneous words is by comparing words. While similar words have the same similarity value, incorrectly spelt words have different similarity values, thus making the words far apart.

Tian *et al*.'s (2002) approach on duplicate record detection splits records into clusters and then map record numbers based on the *n*-gram of field value. Tian *et al*. (2002) aimed to address data quality challenges that arise from syntactic and typographical errors, and to resolve the complex semantic inconsistency among data values. Using the approach of Tian *et al*. (2002), the numbers obtained are put in clusters, and records within each cluster are taken as potential duplicate records. Thereafter, records in each cluster are compared with other clusters to identify true duplicate records. The advantage of this approach is that it does not require preprocessing to correct syntactic or typographical errors in the source data, thus helping to achieve highly accurate detection. Also, the approach ensures only a fixed number of database scans, thus making the algorithm more time efficient. The challenge with the clustering approach is that as the volume of data increases, separating records into clusters could result in a number of shared *n*-grams (Kondrak 2005), where an error present in one cluster tends to affect only a limited number of parts, leaving the other cluster intact (Cavnar and Trenkle 1994). However, when a large number of clusters are involved, it may overlook duplicate words as an error in a cluster may be underestimated while showing results in a limited amount of time, so duplicate words might be missed if many.

Hernandez and Stolfo (1995) apply equational theory, which consists of a set of rules to find whether two records are duplicates. The equational theory identifies equivalent records by a complex domain-dependent matching process, meaning the approach depends on the type of application being used. Although the approach achieves high accuracy of detection, it is application dependent. Hernandez and Stolfo (1998) apply equational theory and transitive closure to detect potential customer names in a direct marketing-type application. The approach uses the transitive closure to combine independent results after multiple database passes in order to produce accurate results in finding duplicate records in massive amounts of data at lower computational cost.

The framework used by Naumann (2013) indicates that one important aspect of the use of algorithms is that it links matched data together to show whether there are duplicate records/words or not. In order to find out whether strings are duplicate or not, several duplicate text detection techniques have been proposed, namely standard approach, sorted neighborhood, edit distance, and adaptive duplicate detection. These are discussed in the following subsections.

i. Standard approach

Monge (2000) indicates that the standard approach to detecting duplicates is to sort the data and perform pairwise comparison. When this is done, data is consecutively arranged, and duplicate data may be located in nearby or opposite extreme positions with increased index size of data (Kołcz and Chowdhury 2008).

ii. Sorted neighborhood technique

The sorted neighborhood technique uses merge and purge to find duplicate text (Hernandez and Stolfo 1995). The algorithmic process of the sorted neighborhood technique starts by creating key attributes, sorting the data based on key attributes and then merging the data.

iii. Adaptive duplicate detection technique

Adaptive duplicate detection (Monge 2000) is based on the concept of transitive closure such that duplicates are connected with each order to form an undirected graph. An undirected edge connecting each duplicate is found when each string corresponds to each order through a pairwise comparison method. This connection is then represented by a union and find approach (Cormen, Leiserson and Rivest 1990; Hopcroft and Ullman

1973). However, the adaptive approach can expand and shrink the cluster, depending on the data size by using priority queue of duplicate data instead of window size.

iv. Edit-distance technique

The edit distance technique uses the idea of the minimum number of operations (such as insert operations and update operations) performed on an individual alphabet to transform one string to another (Elmagarmid *et al*. 2007). The algorithm is domain independent and final collective matched results are displayed using the Union/Find algorithm (which keeps track of cluster of duplicates) instead of graph structure. For instance, the edit distance *dist* between to two strings *s1* and *s2* is matched to a minimum value ε. The two string attributes are required to be less than ε, formulated in Equation 2.44 as

$$\{(x, y, dist(x, y)) \mid x \in s1 \ \wedge y \in s2 \wedge dist(x, y) \leq \varepsilon\} \qquad \textit{Equation 2.44}$$

An optimized mapping function *f* (which finds the letters of a string) over *s1* and *s2* with a new distance function *ndist* much less than initial *dist* is defined such that:

$$\big(\forall x, \forall y, ndist(f(x), f(y)\big) \leq dist(x, y) \qquad \textit{Equation 2.45}$$

The determined *f* and *ndist* is used to compute the pairs *(x, y)* such that:

$$ndist(f(x), f(y)) \leq \varepsilon \qquad \textit{Equation 2.46}$$

where ε represents the minimum value that ensures that the distance between pairs does not exceed the control parameter.

The strength of each pair of string in terms of the distance between two strings/words can be achieved by calculating distances (in terms of score or weight between 0 and 1) between pairs (Elmagarmid *et al*. 2007) using algorithmic techniques such as Damerau–Levenshtein (Damerau 1964), Hamming (Hamming 1950), the Jaro-Winkler distance metric (Jaro 1995, 1989; Winkler 1999), the Smith-Waterman algorithm (Smith and Waterman 1981) and the Basic Local Alignment Search Technique (BLAST) (Altschul *et* *al*. 1990). In the following subsections, the study explores further algorithms that have been developed based on the edit-based methods as these form a significant portion of the study.

a. Damerau–Levenshtein

The Damerau–Levenshtein algorithm (Damerau 1964) is a modified version of the Levenshtein algorithm that considers the transposing of two adjacent characters. The challenge with the Damerau–Levenshtein algorithm is that when characters are transposed, there is additional computation cost. The Levenshtein (1966) algorithm counts the number of edits, such as insertion, deletion and substitution, that are needed to change a string another.

b. Hamming distance

The Hamming distance method (Hamming 1950; Bard 2007) allows only substitution of words with the same/fixed length. In comparing different lengths that not only involve substitution but also insertion or deletion, the Hamming distance method is not appropriate as datasets in the real world have different lengths of words.

c. Jaro distance metric

The Jaro distance is used to compute the distance between two strings (A and B) and is based on matching and transposition of short strings (Jaro 1995, 1989; Winkler 1999) instead of long strings. The Jaro distance (Jaro 1989) and Jaro-Winkler distance allow only transposition of characters and, consequently, it is more suited for comparing short strings like words and names, even though the length of the shortness is not universally defined. The Jaro distance metric compares a short string (such as first and last names) to find the match of characters, then computes a metric based on the two strings. A character is transposed if it occurs immediately after the expected location of a character in a string. The transposition of a character is included in the Jaro distance value calculation only if it occurs in both strings. The letter transposition used in Jaro distance metrics makes it robust at finding similarity measures in short letters. The distance value is computed using the following steps:

Step 1: computation of length |s1| and |s2| of the two strings.

Step 2: search matched (common) character k in strings s1[i] and s2[i] such that s1[i]=s2[j] and $|i-j| \leq \frac{1}{2}\text{min}\{|\text{s}1|, |\text{s}2|\}$

Step 3: compute the number of transpositions *t*, by comparing the *i*th common character in s1 with the *i*th common character in s2. Thus, each non-matched character is a transposition.

The distance value is computed in Equation 2.47 as

$$D_j = \frac{1}{3} * (\frac{k}{|s1|} + \frac{k}{|s2|} + \frac{k-\frac{t}{2}}{k}) \qquad \textit{Equation 2.47}$$

The Jaro distance was enhanced by William E. Winkler (1999) on the basis that two comparing strings are given a higher score if they start with the same letters. The reason being that mistypes of letters are not usually seen in the start of strings (Ilyankou 2014). This suggests that mistype errors mostly occur in the middle or at the end of words. The strings that have longer sets of characters in common at the start of comparison are given a higher similarity score (Ilyankou 2014). The final Jaro-Winkler score is obtained from the expression in Equation 2.48 as

$$D_{Jaro-Winkler} = D_j + l * p * (1 - D_j) \qquad \textit{Equation 2.48}$$

where D_j is the Jaro distance value for a pair of strings, *l* is number of repeating words at the start of two words, and *p* is the coefficient between [0, 1] which is define by a user. In this case, if *p* is arbitrarily set as 5, then it means two strings which start with 3 identical characters can be regarded as the same, thus coefficient *p*=1/3. After several experiments to define a standard coefficient value for *p*, a value of 0.1 was defined as the most appropriate standard value (Ilyankou 2014). The disadvantage of the Jaro-Winkler algorithm is that it works best with short strings or words such as personal names.

d. Smith-Waterman algorithm

The Smith-Waterman algorithm (Smith and Waterman 1981) is a dynamic programming that applies the pairwise method to compare sequence of two strings in order to find the best matching piecewise (local alignment). The Smith-Waterman algorithm performs deletion, updating or insertion (Smith and Waterman 1981) of characters in words, and it is more suited for performing local alignment of words. Generally, alignment is an arrangement of characters in a word. The significance of local alignment is that it either compares a short sequence to a large sequence or a partial sequence to a whole sequence, or it identifies newly determined sequences. Aligning sequences of words helps discover the relationship between the two words. This relationship is expressed as the minimum distance, so the more minimum a distance is, the better the chances to avoid missing matches that guarantees an optimal local alignment and gives the best performance on accuracy of results (Pearson 1991; 1995).

With the Smith-Waterman algorithm, instead of comparing the total sequence of strings as a whole, the algorithm group compares strings into sub-groups or local alignments (that is, sequences with maximum level of similarity) until the search for optimal alignment of strings within each group is complete. The change in alignment is expressed using deletion, updating or insertion of characters in a string (Monge 2000). During string matching, the algorithm considers the gap between two strings and then computes the alignment of strings using a matrix formulation. Three parameters used in the matrix formulation are: the score matrix E of the match of each symbol (that is, space, comma and period) in the alphabet, the cost of starting (s) a gap and the cost of continuing (c) a gap. The ratio of these parameters determines the efficiency of the algorithm. The approximate match in the optimal alignment is the maximum similarity between each string computed in Equation 2.49 as follows:

$$M[i][j] = max \begin{cases} M[i-1][j-1] + s(a_i, b_j) \\ M[i-1][j] - c; \ if\ (a_i, -) \\ M[i][j-1] - c; \ if\ (-, b_j) \end{cases} \qquad \textit{Equation 2.49}$$

where an entry into the matrix *E(i,j)* will produce the best possible match *m* of the prefix of two strings. When the prefixes match, their alignment *a* is found along the diagonal. The best possible score of the matrix is computed in Equation 2.50 as:

$$Score = Max_{i,j=1}^{n}(Matrix[i][j]) \qquad \textit{Equation 2.50}$$

A change on an item can be as a result of deletion, updating the characters in a string. Although computational time and complexity is a challenge with the Smith-Waterman algorithm, the search process tends to minimize the distance or maximize the similarity between the compared strings (Altschul *et al*. 1990) and gives the best performance on accuracy of results.

e. Basic Local Alignment Search Technique

The BLAST algorithm (Altschul *et al*. 1990; Shpaer *et al*. 1996) is a heuristic method that finds the highest score of local optimal alignments between a query sequence and a database. Basically, the BLAST for sequence alignment is based on computational biology for protein and DNA analysis (Guo, Wang and Devabhaktuni 2011). The BLAST algorithm is based on the assumption that a good alignment often contains short lengths of same matches (Altschul *et al*. 1990). This suggests that when two sequences of words are similar, there is a shorter length that results in high similarity.

The algorithm operates in three steps: the first step is to accept a word length and find the score; in the second step, the database is searched; and the third step finds each hit if it is within the threshold score for the Maximal Segment Pair. The BLAST algorithm was developed from the Smith-Waterman algorithm. The difference between the BLAST algorithm and the Smith-Waterman algorithm is that the BLAST algorithm finds short matches between sequences of string for optimal alignment without considering an entire sequence of string, so less computation time is involved. The Smith-Waterman algorithm also considers an entire sequence of strings to find local optimal alignments. Therefore, high computational time is involved. The advantage of Smith-Waterman is the accuracy of the entire sequence local alignment (Altschul *et al*. 1990). Thus, the Smith-Waterman algorithm avoids missing important data in the string of data. Heuristic methods are random search methods that help reduce the time and computational cost to speed up the local alignment search (Rajasekaran *et al*. 2001). Although the BLAST algorithm provides search results in a short time, it may not guarantee accurate results as compared to the Smith-Waterman algorithm (Shpaer *et al*. 1996).

In summary, application tools developed from the Smith-Waterman algorithm include FAST All (FASTA) (which works with any alphabet (Pearson 1991)) and FASTP (Lipman and Pearson 1985) for protein and DNA sequence alignment in a database (Pearson 2014). Smith-Waterman has also been applied in the development of hardware devices. An example of such a development is the "Fast data finder", which matches the accuracy of software versions while greatly speeding up its execution (Shpaer *et al*. 1996). Pearson (1995) compares the accuracy of BLAST, FASTA and Smith-Waterman on protein sequences, and the results suggest that both FASTA and Smith-Waterman are more sensitive than BLAST. Pearson (2014) indicates that the application of BLAST and FASTA can be challenged by today's very large protein databases. Thus, search sensitivity (in terms of accuracy) can be improved by searching smaller comprehensive databases for complete protein sets where a slight mismatch is not acceptable, which might be applied to datasets considered to contain "sensitive" information (e.g. health records).

CLC bio (2007) indicates that the current dispensation of big data requires fast and effective data analysis. Algorithms like BLAST have replaced the Smith-Waterman algorithm as demands for time to handle large amounts of data are stronger and more prevalent. However, the concern about the risk of missing important information, if not using the most sensitive algorithm for database searches, becomes even more relevant. Thus, the use of the Smith-Waterman algorithm is significant when accuracy of information is key. Moreover, the use of the Smith-Waterman algorithm is becoming more and more widespread when high accuracy is needed (CLC bio 2007). In the sense that

Smith-Waterman search guarantees to find optimal local alignments and returns only one result per comparison, however, the search process performs a larger number of computations than BLAST. Therefore, the Smith-Waterman algorithm may be enhanced to perform accurate comparison with less computing time when large volumes of data are required. Meanwhile, in principle, all pairs of records/words in a dataset should be compared, which is highly infeasible when a large volume of data is used in a big data analytics framework (Naumann and Herschel 2010).

CLC bio (2007) indicates that the Smith-Waterman algorithm should be used when obtaining exact search results on comparison is more important than time. This proposition was as a result of an empirical study that used the Smith-Waterman algorithm to compare query sequences and the sequences in the database on a character-to-character level. The study by CLC bio (2007) demonstrates that Smith-Waterman was able to find optimal local alignment instead of global alignment considering segments of all possible lengths by allowing deletion and insertion of arbitrary lengths to optimize the similarity measure (CLC bio 2007). The deletion and insertion process led to longer time to compute the optimal local alignment. However, this suggests that with the ever-increasing scale of data, enhancing the Smith-Waterman algorithm to improve both accuracy and time to compute deletion and insertion becomes relevant in duplicate detection.

Monge (2000) suggests that the accuracy of duplicate detection algorithms can be improved by defining a small window size where results of several database passes for duplicates are combined for the same cost, rather than one pass over the database with a large window size. One way to combine the results of multiple passes on words is by computing the transitive closure of all discovered pairwise alignments using an “is a duplicate of” relationship. However, in typical databases, duplicate words tend to be distributed sparely over the space of possible records, and the propagation of error is rare (Monge 2000).

This discussion above presented a review of related work on duplicate detection algorithms and their associated challenges. The review showed the following distinctions: while Jaro distance (Jaro 1989) allows only transposition of characters, the Smith-Waterman algorithm allows deletion, updating or insertion of characters (Smith and Waterman 1981). The challenge with the Smith-Waterman algorithm is that it is unable to perform global alignment of characters to reveal accurate duplicate words, whereas the Jaro-Winkler algorithm is used to compare short words such as names. The reason for choosing the Jaro-Winkler and Smith-Waterman algorithms is to demonstrate the accuracy of pairwise comparison of words in large datasets and determine whether pairwise comparison may lead to information loss if large amounts of data are involved. The Naumann's (2013) framework for duplicate detection as explained ealier will be adopted for this study because of its simplified process that is identity, similarity measure, algorithm used and evaluation aspects.

2.2.1.3 Stage 3: Data transformation

Data transformation is the process of converting or consolidating data through normalization, hierarchical representation of attributes (generalization) and attribute construction into a suitable format for mining and visualization (Panda, Nag and Jana 2014).

Data normalization resolves differences in choice of measurement by assigning equal weights to all data attributes considered for transformation. Normalization is significant for data classification algorithms because it puts attributes within a small and specified range (0.0, 1.0) for easy analysis of items. The approaches used in data normalization are min-max, z-score (zero-mean) and decimal scaling (Panda *et al*. 2014).

The min-max approach does linear transformation of the original data value and preserves the relationship with the original data through mapping. Given an observed value v_i of an attribute *A*, the mapping onto *v* in the new range [New$_{\min}$(A), New$_{\max}$ (A)] is computed in Equation 2.51 by:

$$v_i' = \frac{v_i - min(A)}{max(A) - min(A)}(new_{max}(A) - new_{min}(A)) + new_{min}(A) \qquad \textit{Equation 2.51}$$

where v_i' contains the min-max values.

Instances where the min-max values of attributes are unknown, the z-score approach is most suitable. This approach uses the mean and standard deviation (σ) of numeric attribute *A*. The observed numeric value v_i is then normalized using the expression in Equation 2.52 as:

$$v_i' = (v_i - mean)\ /\sigma \qquad \textit{Equation 2.52}$$

The standard deviation (σ) for the population is expressed in Equation 2.53 as:

$$\sigma = \sqrt{\frac{1}{N} * \sum_{i=1}^{N}\left(v_i - \overline{mean}\right)^2} \qquad \textit{Equation 2.53}$$

where *N* is the total population. When population standard deviation is unknown, samples are small.

The decimal scaling (Panda *et al*. 2014) is normalized by moving decimal points based on the absolute value of the attribute *A*. A numeric value v_i is normalized to v_i' as in Equation 2.54:

$$v_i' = \frac{v_i}{10^j} \qquad \textit{Equation 2.54}$$

where *j* is the smallest integer with the max ($|v_i'|$) < 1, v$_i$ is the range values, $v_i'$is the scale value, and the range of decimal scale is between (-1, 1).

The data transformation helps better understand data distribution and also results in data discretization in range.

2.2.1.4 Stage 4: Data reduction

Data reduction is the process of reducing volumes of data from original data sources (Rehman 2016). During the search process, relevant features are selected and condensed into groups/subsets by removing redundant features and condensing the size of data into subsets. The significance of this process is that relevant features are selected with less computational time (Crone, Lessmann and Stahlbock 2006) as a result of the efficient search process.

The approaches to data reduction when velocity and volumes matter are: dimensionality reduction, numerosity reduction and data compression (Rehman 2016). Dimensionality reduction is the removal of attributes that are considered not important. Although what constitutes "not important" is subjective, meaning it refers to users' discretion, techniques used to implement dimensionality reduction include Principal Components Analysis (PCA) and random forests/decision tree ensembles. PCA is a data preprocessing technique that combines similar or correlated attributes together and creates new attributes that are superior to the original attributes (Janecek and Gansterer 2008). The random forests/decision tree ensemble (Breiman 2001) approach to dimensionality reduction is useful in generating large and carefully constructed sets of trees against a target attribute and then use each attribute's usage statistics to find the most informative subset of features. The random forest approach is related to nearest neighbor predictors due to the set of trees emanating from a single neighbor. Thus, if a single attribute is often selected (based on the rate/frequency/calculated score) as best split, then that particular feature is selected and retained into a respective subset as the relevant feature. The split which forms an ensemble method is based on the divide-and-conquer approach. Random split selection (Dietterich 1998) is applied to split each selected node at random from among several *K* best splits (Breiman 2001). During the ensemble process, each iteration generates an output set of features. These outputs constitute new training sets, which are further randomized in the original training set used for the feature selection (Breiman 1999).

In contrast, numerosity reduction uses statistical techniques such as regression and log-linear models, sampling and clustering to reduce the size of data, while data compression techniques compress the original data (either string or video) into approximated data (Ramya and Pushpa 2016). In respect of these approaches, the following subsections review the methods of feature selection.

i. Methods of feature selection

Generally, features may be characterized as relevant, irrelevant or redundant. A feature may be considered as an attribute of data (e.g., a person has an attribute such as name). A feature is said to be a relevant feature when it has an influence on output features and its role cannot be assumed by other features in a dataset. A feature is irrelevant when it does not have any influence on output features and its values are generated at random. Finally, a redundant feature is one that assumes a role of another feature. The characteristic of a feature leads to the use of different methods for feature selection. These methods are categorized into the filter method (which is classifier-independent), the wrapper method (which is classifier-dependent) (Liu *et al*. 2017) and the embedded method (Elisseeff and Guyon 2003).

The filter method finds the relevance of a feature (Liu *et al*. 2017) in a class by evaluating each feature without the use of a learning algorithm. A feature classification algorithm that adopts the filter model evaluates the goodness of each feature and ranks features by distance measure (Almuallim and Dietterich 1994), information measure (Ben-Bassat 1982) and dependency measure (Hall 2000).

The distance measure (Almuallim and Dietterich 1994) finds the difference in value between two features. If the difference is equal to zero, then the features are indistinguishable, otherwise the features are distinguishable. Hence, features are separated into different subsets based on the distance computed. In contrast, the information measure finds the information gain of a feature. The information gain is expressed as the difference between uncertainties, defined as a situation where a decision to select a feature is based on the prior amount of information gain on each feature and the expected amount of information gain of each feature.

The dependency measure (also referred to as the correlation measure) predicts the value of one feature from the value of another feature. The prediction of a feature is based on how strongly the feature is associated with/dependent on a class/subset of features. A high dependency value could suggest that a feature is strongly associated with a subset, otherwise a feature is weakly associated with a subset. The measured values are ranked in terms of relative importance to select the relevant feature. Moreover, these measures used for evaluating and ranking features are the basis for not using a learning algorithm in the selection of features in the filter method. Since the filter method does not use a learning algorithm, less computational time is spent on selecting individual features. The challenge with the filter method is that it ignores the combination of features because it is unable to learn from features.

The wrapper method (Kohavi and John 1996) uses a learning algorithm to learn from every possible feature subset, trains the selected subset and evaluate its usefulness (Liu *et al.* 2017; Uncu and Turksen 2007). Selected features are ranked according to their usefulness and predictive power of the classification algorithm, which is measured in terms of performance of the classification algorithm (Fong, Yang and Deb 2013). The wrapper method uses a statistical re-sampling technique called Cross Validation and measures the accuracy of classification results. The approach to learning methods includes the artificial neural network and the support vector machine, which are discussed in subsequent paragraphs. The search strategies used in the wrapper search method are categorized into sequential search (forward selection and backward elimination search), exhaustive search and random search (Dash and Liu 1997). The sequential search strategy includes the use of forward and backward techniques to iteratively add or remove features. On the one hand, the forward selection algorithm starts an iteration with an empty set and uses specified objective functions to select features into subsets. On the other hand, the backward selection algorithm starts with a full set of features and uses a specified objective function to remove least significant features that do not met the set criteria (Marill 1963).

An exhaustive search performs a complete search of the entire feature subset and then selects the possible optimal result (Waad, Ghanzi and Mohamed 2013). When the number of features grow exponentially, the search takes more computational time (Aboudi and Benhlima 2016), thus leading to low performance during search. The random search strategy (also referred to as population-based search) is a meta-heuristic optimization approach based on the principle of evolution in search for a better solution. The advantage of the random search strategy over sequential and exhaustive search is the reduction in computation cost and time.

The third category of the feature selection method is the embedded method, which selects features by putting data into two sets, namely training and validation sets. When variables that define features are selected for training, retraining a predictor variable for every variable subset is avoided (Kumar and Minz 2014), and this makes the embedded method reach solutions fast. However, predictor variable selection is model specific.

As mentioned earlier, among the traditional approaches to learning methods/machine learning methods are the artificial neural network (ANN) and support vector machine (SVM). The neural network is an interconnected group of nodes (neurons) where each node receives inputs from other nodes and assigns weights between nodes to adapt so that the whole network learns to perform useful computations (Bishop 2006). The challenge with algorithms based on ANN is that it requires many iterations over the training set before choosing its parameter (Aamodt 2015), leading to high computation. The neural network structure and learning algorithms use the perceptron neural network (that is, an algorithm for supervised classification) and back-propagation. The advantage of a learning algorithm is that it helps in adapting weights of a neural network by minimizing error between a desired output and an actual output. The aim of the back-propagation algorithm is to train multi-layer neural networks by computing error derivatives in hidden activities and updating weights accordingly (Kim 2013). The back-propagation algorithm uses gradient descent to adjust the connections between units within the layers such that any given input tends to produce a corresponding output (Marcus 2018).

Another traditional approach to learning is the use of an SVM. The SVM performs classification by constructing an *N*-dimensional hyper-plane that optimally separates data into two categories (Boser, Guyon and Vapnik 1992). However, when large volumes of data are involved, it results in high computational cost in training and selection of features (Lin, 2006) and may not be efficient in providing optimal results. These challenges led to the concept of deep learning, which historically originated from ANNs (Deng and Yu 2013).

Deep learning as a method for feature selection is defined as a sub-field of machine learning that is based on learning several levels of representation, corresponding to a hierarchy of features where higher-level features are defined from lower-level ones, and the same lower-level features can help define many higher-level features (Deng and Yu 2013). Marcus (2018) indicates that deep learning is a statistical technique that helps classify patterns based on sampled data using neural networks with multiple layers. The neural networks used in deep learning consist of a set of input units that stand for things like pixels or words, multiple hidden layers (the more such layers, the deeper a network is said to be) containing hidden units (also known as nodes or neurons), and a set of output units, with connections running between those nodes (Marcus 2018) to form a mapped structure between inputs and outputs. This mapped structure that is formed between the input and output nodes gives an indication of how nodes are connected to form a complex representation of large data. In this regard, providing an efficient way to optimize a complex representation of data could ensure that the test dataset used in neural networks loosely resembles the training set. This close resemblance suggests a minimization of deviations between test and training sets in large datasets. Therefore, deep learning is a way to optimize complex systems to map inputs and outputs, given a sufficient amount of data (Marcus 2018).

In principle, deep learning uses multiple hidden layers of non-linear processing that is hierarchical, as well as parameters to learn from hidden layers using different algorithms (such as back-propagation algorithms) with large amounts of available training data (Patel, Nguyen and Baraniuk 2015). Based on these two principles, deep learning methods for classifying patterns are deep discriminative models/supervised-learning models (e.g. deep neural networks or DNNs, recurrent neural networks or RNNs, convolutional neural networks or CNNs, etc.) and generative/unsupervised models (e.g. deep belief networks or DBNs, deep Boltzmann machines or DBMs, etc.).

A deep neural network (DNN), sometimes referred to as DBN, is a multilayer network with many hidden layers, whose weights are fully connected and initialized (pre-trained) using stacked RBMs or DBN (Deng and Yu 2013). A recurrent neural network (RNN) is a discriminative model but has also been used as a generative model where "output" results from a model represent the predicted input data. When an RNN is used as a discriminative model, the output result from the model is assigned a label, which is associated with an input data sequence (Deng and Yu 2013). Recurrent nets (RNNs) have been applied on sequential data such as text and speech (LeCun, Bengio and Hinton 2015) to scale up large text and speech recognition. RNNs have been found to be very good at predicting the next character in the text or the next word in a sequence, but they can also be used for more complex tasks (LeCun *et al*. 2015).

Learning of parameters in RNN has been improved through the use of information flow in bi-directional RNNs and a cell of LSTM (long short-term memory). The challenge is that when training neural networks for deep learning classification problems, the backpropagated gradients approach often used either grows or shrinks (that is, decays exponentially in the number of layers (Schmidhuber 2014)) at each time step, so over many time steps it typically explodes or vanishes (that is, increases out of bounds or decreases at each iteration) (LeCun *et al*. 2015). Several methods to solve the exploding and shrinking of a learned parameter include the primal-dual training method, cross entropy (Deng and Chen 2014), echo state networks and sigmoid as activation functions (Sohangir *et al*. 2018), among others. While the primal-dual training method was formulated as an optimization problem, the cross entropy is maximized, subject to the condition that the infinity norm of the recurrent matrix of the RNN is less than a fixed value to guarantee the stability of RNN dynamics (Deng and Yu 2013). In the echo state network, the output layers are fixed to be linear instead of nonlinear, and the recurrent matrices are designed, not learned. Similarly, the input matrices are also fixed and not learned, due partly to the difficulty of learning. The sigmoid functions are mathematical expressions that define the output of a neural network given a set of data inputs. Meanwhile, the use of LSTM enables networks to remember inputs for a long time using a memory cell that acts like an accumulator, which has a connection to itself at the next time step (iteration) and has a weight, so it copies its own real-valued state and temporal weights. But this self-connection is multiplicatively gated by another unit that learns to decide when to clear the content of the memory (LeCun *et al*. 2015). LSTM networks have subsequently proved to be more effective, especially when they have several layers for each time step (LeCun *et al*. 2015).

A convolutional neural network (CNN) shares many weights, and pools outputs from different layers, thereby reducing the data rate from the lower layers of the network. Abdel-Hamid *et al*. (2014) indicate that time sharing and frequency are the two dimensions used in CNN that are useful in time delay neural networks, such as for speech recognition. The CNN has been found highly effective in computer vision, image recognition (LeCun *et al*. 1998; Krizhevsky, Sutskever and Hinton 2012) and speech recognition (Abdel-Hamid *et al*. 2014; Deng and Yu 2013) in order to analyze the internal structure of complex data through convoluted layers. The CNN has also gained attention in text data, such as sentence modeling, search engines, in systems for tagging (Weston, Chopra and Adams 2014), sentiment analysis (Sohangir *et al*. 2018) and stock market price prediction (Aamodt 2015).

The Deep Boltzmann machine (DBM) is a special Boltzmann machine where the hidden units are organized in a deep, layered manner. Only adjacent layers are connected, and there are no visible-visible or hidden-hidden connections within the same layer. A deep belief network (DBN) is a probabilistic generative model composed of multiple layers of stochastic, hidden variables. The top two layers of deep belief networks have undirected, symmetric connections between them, whereas the lower layers receive top-down, directed connections from the layer above. Table 2.2 shows a summary of related work on deep learning:

Table 2.2 Deep learning methods and problem domains

Deep learning method	Search/problem domain	Author(s)
Convolutional Deep Belief Networks	Unsupervised feature learning for audio classification	Lee, Largman, Pham and Ng
Convolutional Deep Belief Networks	Scalable unsupervised learning of hierarchical representations.	Lee, Grosse, Ranganath and Ng (2009)
Deep Convolutional Neural Networks (DCNN)	Huge number of high-resolution images	Krizhevsky *et al*. (2012)
Deep Convolutional Neural Networks	Event-Driven Stock Prediction	Ding *et al*. (2015)
Deep Neural Networks	Classification of stock and prediction of prices	Batres-Estrada (2015)
Deep Neural Network-Hidden Markov Models (DNN-HMMs)	Discovering features in speech signals	Jaitly (2014)
Training the CNN architecture based on the back-propagation algorithm	Character recognition in sequential text	LeCun *et al*. (1998)
Convolutional Neural Network	Stock trading	Siripurapu (2015)

From Table 2.2, it can be observed that recent research has applied deep learning to different search domains such as image processing, stock trading and character recognition in sequential text analysis, among others that demonstrate the unique capabilities of deep learning methods for classification of features in large dataset analysis.

The distinction between supervised and unsupervised learning is that, in supervised learning, a pre-classified example of features is available for learning and the task is to build a (classification or prediction) model that will work on unseen examples. In unsupervised learning, meanwhile, there are neither pre-classified examples nor feedback to the learning model (this technique is suitable for clustering and segmentation tasks) (Barto and Sutton 1997; Kotsiantis 2007). These networks are generally trained by a gradient descent algorithm designated back-propagation. The back-propagation algorithm computes the gradient (a vector of partial derivatives) of an objective function with respect to the parameters in a neural network (Le 2015). However, for deep networks, back-propagation alone has the problem of being trapped in local optima in the non-convex objective function (Patel *et al*. 2015).

Building classifiers from deep learning techniques integrated with meta-heuristic search methods (also referred to as random search strategy, as mentioned earlier) enhances computational efficiency and the quality of selecting useful and relevant features (Li *et al*. 2017). The advantage of meta-heuristic search methods is that they use random search strategies to avoid being trapped in local optima when the search space grows exponentially. Meta-heuristic algorithms that have been integrated with traditional machine learning methods include the following, as indicated by Fong *et al*. (2013) and summarized in Table 2.3:

Table 2.3: Meta-heuristics algorithms integrated with traditional method

Author(s)	**Traditional methods of classification**	**Meta-heuristics/bio-inspired algorithm**	**Search domain**
Ferchichi, Laabidi and Zidi (2009)	Support vector machine	Tabu search, GA	Urban transport
Unler and Murat (2010)	Logistic regression	Particle swarm optimization (PSO), Scatter search, Tabu search	General
Unler, Murat and Chinnam (2011)	Support vector machine	PSO	General
Abd-Alsabour, Randall and Lewis (2012)	Support vector machine	ACO	General
J. Wang, Hedar, S. Wang and Ma (2012)	Rough set	Scatter search	Credit scoring
Fong *et al*. (2013)	Neural Network	Wolf Search Algorithm	General

It can be observed from Table 2.3 that research is focused on traditional machine learning methods with meta-heuristic search methods. However, with the current dispensation of very large volumes of data, traditional machine learning methods are not suitable because of the risk of being stuck in local optima and the likelihood that the same results might be recorded as more data is generated, which might not give an accurate result on feature selection for a classification problem. Among the meta-heuristics-based algorithms used are GA, Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), the Wolf Search Algorithm (WSA), Multiple Species Flocking (MSF) model and the Bat Algorithm.

a. Genetic Algorithm

GAs are an evolutionary approach based on survival of the fittest. This survival mechanism, as explained earlier, helps formulate adaptive search procedures to select feature subsets by optimizing an objective function/fitness function in any given search problem.

b. Ant Colony Optimization (ACO)

Ant Colony Optimization (ACO) (Dorigo and Cambardella 1997) is a meta-heuristic inspired by the foraging behavior of real ants in their search for the shortest paths to food sources. When a source of food is found, ants deposit a pheromone to mark their path for other ants to traverse. A pheromone is an odorous substance used as a medium of indirect communication between ants. The quantity of pheromone depends on the distance, quantity and quality of the food source (Al-Ani 2007). However, the pheromone substance tends to decay or evaporate with time, which prevents ants from converging to sub-optimal positions (Stützle and Dorigo 2002). When a lost ant that moves at random detects a laid pheromone, it is likely that it will follow the path to reinforce the pheromone trails. Thus, ants make probabilistic decisions by updating their pheromone trail and local heuristic information (Al-Ani 2007) to explore larger search areas.

Dorigo and Cambardella (1997) define a trail as the formation and maintenance a line. Ants use trails or pheromone trails both to trace a path to a food source and to prevent themselves from getting trapped in a single food source (Agbehadji 2011). Each time an ant searches, trails are drawn and pheromone substances are deposited in the trail. This substance helps ants to communicate with each other about the location of food sources. Therefore, other ants continuously follow this path and also deposit substances for the trail to remain fresh. Computational systems that depict ant pheromone behavior creates local and global trail-updating formulations. This updating strategy constitutes the property of a meta-heuristic method (Blum and Roli 2003), thus an ant colony is regarded as a meta-heuristic algorithm.

The local trail updating formula is motivated by trail evaporation or trail decay in real ants (Dorigo and Cambardella 1997), and this formula is expressed in Equation 2.55 as

$$\tau(r,s) \leftarrow (1-\alpha) * \tau(r,s) + \alpha * \tau_0 \qquad \textit{Equation 2.55}$$

where τ_0 is a parameter, $r(r,s)$ is the edge on a line, and α is a control parameter.

The following probabilities formula was applied to find s in Equation 2.56:

$$s = \begin{cases} \underset{u \notin M}{arg\,max}\left\{[\tau(r,u) * \eta(r,u)]^{\beta}\right\} & if\ q < q_0 \\ S & otherwise \end{cases} \qquad \textit{Equation 2.56}$$

where $\tau(r, s)$ is the amount of pheromone trail on the edge (*r*,*u*), $\eta(r, u)$ is a heuristic function that is the inverse of the distance between two edges *r* and *u*, β is a parameter that represents the relative importance of the pheromone trail and of closeness, *q* is the random probability between 0 and 1 and *S* is a random variable selected according to the following probability distribution, which favors edges that are shorter and have a higher level of pheromone trail (Dorigo and Cambardella 1997). This is expressed in Equation 2.57 as

$$p_k(r, s) = \begin{cases} \frac{[\tau(r,s)]*[\eta(r,s)]^{\beta}}{\sum_{u \notin M}[\tau(r,u)]*[\eta(r,s)]^{\beta}} & if\ s \notin M \\ 0 \quad otherwise & \end{cases} \qquad \textit{Equation 2.57}$$

where $p_k(r, s)$ is the probability with which ant *k* chooses to move from one edge *r* to another edge *s*.

Global trail updating is an approach used to emulate the property of differential pheromone trail accumulation, which is an interplay between length of path and continuity of time (Dorigo and Cambardella 1997). Global trail updating is expressed in Equation 2.58 as

$$\varphi(r, s) \leftarrow (1 - \alpha) * \varphi(r, s) + \alpha * \Delta\varphi(r, s) \qquad \textit{Equation 2.58}$$

where $\Delta\varphi(r, s)$ is the amount of pheromone deposited on edge φ (r, s), and α is a control parameter.

Trail evaporation is used in ant colony optimization as a strategy to avoid getting stuck in local optima. (Agbehadji 2011). This trail evaporation is an update process where trails seen are expressed in Equation 2.59 as:

$$\tau = \tau + \Delta\tau \qquad \textit{Equation 2.59}$$

Trails that leave the surrounding are expressed in Equation 2.60 as:

$$\tau = \tau - \Delta\tau \qquad \textit{Equation 2.60}$$

The value Δ is a constant determined by input parameters such as the size of the population *q*, minimum or initial pheromone level *τinit* and maximum pheromone level *τmax* (Guntsch and Middendorf 2002). Thus,

$$\Delta = (\tau max - \tau init)/q \qquad \text{Equation 2.61}$$

Given that ants move at approximately the same speed and deposit their pheromone trail at the same rate (Dorigo and Cambardella 1997), it is possible that these trails emit substances into the environment. The intensity of substances emitted can serve as a point of attraction to other animals within a habitat. Therefore, the rate of decay of a trail, or trail evaporation, plays an important role in determining newness or oldness of a trail. The disadvantages of ACO is, firstly, that the time to convergence to optimality is uncertain even though convergence is guaranteed. Secondly, the probability distribution changes by iteration, which leads to sequences of random decisions (not independent). Although the random decision could be seen as a challenge, the advantage is that it can be used in dynamic applications (e.g. adapting to changes such as new distances) (Shekhawat, Poddar and Boswal 2009).

The behavior of ants has been applied to solve many optimization-related problems, including data mining, where it was shown to be efficient in finding best possible solutions. When applied to feature selection ACO improves on the performance of feature selection by finding the best possible path in terms of the path with the least amount of error.

c. Wolf Search Algorithm

The WSA is a meta-heuristic optimization algorithm based on wolf preying behavior (Tang *et al*. 2012). The behavior of wolves, as described earlier, includes the ability to hunt independently by remembering their own trait (meaning wolves have memory); the ability to only merge with their peer when the peer is in a better position (meaning there is trust among wolves to never prey on each other); the ability to escape randomly upon the appearance of a hunter (Tang *et al*. 2012); and the use of scent marks as a way of demarcating their territory and communicating with other wolves (Agbehadji *et al*. 2016). This behavior enables wolves to randomly adapt to their environment when hunting. If a wolf finds a new, better position, it deposits a scent mark that is sensed by other wolves, and this helps communicate the best position already inhabited by a companion wolf.

Grey wolves is another kind of social animal that belongs to the family of wolves. Grey wolves are regarded as apex predators because it is found on top of its food chain whiles other wolves are found below the food chain. This forms a social hierarchy in its social structure of hunting. Mostly, grey wolves found on top of the food chain (referred to as leaders, male or female, are called alphas) are able to identify location of a prey and encircle them. The alphas are responsible to make decision about hunting. Other wolves below the food chain are referred to as subordinate that is beta, or omega. In this regard, omega wolves have to submit to betas and alphas in the hierarchy. This social structure enables grey wolves to search for prey according to the position of the alpha, beta, and omega. Thus, grey wolves can diverge from each other to search for fitter prey and converge to attack prey (Mirjalili, Mirjalili and Lewis 2014).

The wolf search is an iterative search process that starts by the setting of initial parameters, random initialization of the population, evaluation and updating of a current population using a fitness test, and continuing to create new generations/iterations until some stopping criterion is met. Unlike GA, which uses operators such as mutation, crossover and selection methods, the WSA uses attractiveness of prey within its visual range, instinctively flocking together in a pack that is collective, randomly escaping from its enemy, and organizing individual searches. Therefore, the swarming behavior of the WSA is delegated to each individual wolf, and this could form multiple leaders swarming from multiple directions towards the best solution rather than a single flock searching for an optimum in one direction at a time (Tang *et al*. 2012). However, the performance depends heavily on the manually chosen parameters values (Song, Fong and Tang 2016). This could be resolved through self-adaptive methods in parameter value selection.

d. Particle Swarm Optimization

A particle swarm is a bio-inspired method based on swarm behavior in nature, such as fish and bird schooling (Kennedy and Eberhart 1995). The swarm behavior is expressed in terms of how particles adapt, exchange information and make decisions on change of velocity and position within a space, based on the position of other neighboring particles. The search characteristics of a particle swarm involves the initialization of particles, and several iterations are performed to update the position of each particle, depending on the value assigned to its velocity, and they are combined to the best previous own position and the position of the best element among the global population of particles (Aboudi and Benhlima 2016). The advantage of swarm behavior is that as an individual particle makes a decision, it leads to an emergent behavior (Krause *et al*. 2013). This emergent behavior is as a result of local interaction among individual particles in a population of particles. Particle swarm methods are computationally less expensive, which makes them more attractive and effective for feature selection. Again, each particle discovers the best feature combination as it moves in a population.

When applying particle swarm methods to any feature selection problem, it is important to define threshold value during initialization so as to decide which feature is selected or discarded. Often, it is difficult for a user to explicitly set a threshold since it might influence the performance of the algorithm (Aboudi and Benhlima 2016). Xue, Bing and Zhang (2014) suggest an initialization strategy that adopts a sequential selection algorithm that guarantees accuracy of classification and provides the number of feature subsets that are selected (Aboudi and Benhlima 2016). The advantage of particle swarm is that during the initialization/generation of particles, only the most optimist particle can transmit information onto the other particles, and the speed of researching a search space is very fast (Bai 2010). However, the method easily suffers from partial optimism, which makes it less exact in regulating the speed and the direction of each particle in the search space (Bai 2010).

e. Multiple Species Flocking

The MSF model is a bio-inspired approach that mimics the social behavior of birds (Yang 2010). The social behavior is exhibited through continuous exchange of information within a flock to align its position in the same direction as nearby birds. The birds might make the decision to move from one position to another without communicating with each other (Eberhart, Shi and Kennedy 2001). The decision to move is steered by three basic rules, namely alignment, separation and cohesion. These rules are mathematically expressed and described as follows:

The *alignment rule (or velocity matching rule)* is when a boid moves in the same direction of the nearby boids where the velocity vector is aligned with the average velocity vector of the neighboring local flocks. The alignment rule is expressed in Equation 2.62 as

$$d(F_i, A_c) \leq r_1 \wedge d(F_i, A_c) \geq r_2 \Rightarrow \vec{v}_{ar} = \frac{1}{n}\sum_i^n \vec{v}_i \qquad \textit{Equation 2.62}$$

where r_1 and r_2, with $r_1 > r_2$, represents the radius r of the visibility range of the boids and minimum distance among them respectively, and $d(F_i, A_c)$ represents distance between current boid A_c and its flockmate. Further, F_i. $\vec{v}_i$ is the velocity of the boid F_i, and n represents the number of neighbors.

The *separation rule (or collision avoidance)* avoids closeness between the boids, and is formulated in Equation 2.63 as

$$d(F_i, A_c) \leq 2r_2 \Rightarrow \vec{v}_{sp} = \sum_i^n \frac{\overrightarrow{\vec{v}_i + \vec{v}_c}}{d(F_i, A_c)} \qquad \textit{Equation 2.63}$$

where $\vec{v}_{sp}$ represents the separation velocity, and $\vec{v}_c$ and $\vec{v}_i$ represent the velocities of the current boid and *i*th flockmate.

The *cohesion rule (or flock centering)* moves a boid towards the center of the flock or towards other nearby boids. The rule is formulated in Equation 2.64 as

$$d(F_i, A_c) \leq r_1 \wedge d(F_i, A_c) \geq r_2 \Rightarrow \vec{v}_{cr} = \sum_i^n (P_i - P_c) \qquad \textit{Equation 2.64}$$

where $\vec{v}_{cr}$ represents the cohesion velocity, P_{i} and P_{c} represent the position of current boid A_{c} and a neighbor boid F_{i}, and $(P_i - P_c)$ calculates the directional vector point.

The *feature similarity rule* checks the closeness of each boid using the strength of closeness to ensure that similar boids are close to each other. This feature similarity is expressed in Equation 2.65 as

$$v_{sim} \; = \; \sum_i^n (Sim\ (F_i, A_c) * d(P_i, P_c)) \qquad \textit{Equation 2.65}$$

where v_{sim} represents the velocity as a result of similarity of features, *Sim(F*$_i$*,A*$_c$*)* represents the similarity value between features of boids F_i and A_c, and *d(P*$_i$*,P*$_c$*)* represents the distance between their position.

The *feature dissimilarity rule* applies when boids do not have similar features and thus stay away from each other. Therefore, dissimilarity is inversely proportional to the similarity features formulated in Equation 2.66 as:

$$v_{dsim} \; = \; \sum_i^n \frac{1}{Sim(F_i,A_c)*d(P_i,P_c)} \qquad \textit{Equation 2.66}$$

where v_{dsim} is the velocity as a result of dissimilarity of features. The final flocking behavior is expressed as the sum of calculated velocities and weighted action of rules to represent the net velocity vector v of a boid in space. The net velocity $\vec{v}$ is expressed in Equation 2.67 as:

$$v = w_{ar} * \vec{v}_{ar} + w_{sp} * \vec{v}_{sp} + w_{cr} * \vec{v}_{cr} + w_{sim} * v_{sim} - w_{dsim} * v_{dsim} \qquad \textit{Equation 2.67}$$

where v is the boid's velocity in the virtual space and $w_{ar,} w_{sp,} w_{cr,} w_{dsim}, w_{sim}$ are pre-defined weight values per boid in respect of the alignment rule, separation rule, cohesion rule, dissimilarity rule and similarity rule respectively.

The feature similarity rule allows a boid to stay close to boids with similar features and stay away from other boids that have dissimilar features (Cui and Potok 2006). This feature similarity rule is expressed in Equation 2.68 as

$$v_{fr} = \sum_{i}^{n} \frac{(S(B,X)-T)*\overrightarrow{(P_{\imath}-P_{c})}}{d(F_{i},A_{c})} \qquad \textit{Equation 2.68}$$

where $d(F_i, A_c)$ is the distance between boid B and its neighbor X, n is the total number of the boid B's local neighbors, $\overrightarrow{(P_{\imath} - P_c)}$ calculates a directional vector point, *Sim(B,X)* is the similarity value between the features of boid B and X, and T is the threshold for separating similarity and dissimilarity boids.

The strength of the attracting force for similar boids and the repulsion force for dissimilar boids is inversely proportional to the distance between the boids and the similarity value between the boids' features. The flocking behavior of multiple species is determined by weighting actions of all four rules and summing to give the net velocity v for an active boid, as expressed in Equation 2.69:

$$v = w_{ar} * \vec{v}_{ar} + w_{sp} * \vec{v}_{sp} + w_{cr} * \vec{v}_{cr} + w_{fr} * v_{fr} \qquad \textit{Equation 2.69}$$

where v is the boid's velocity in the virtual space and $w_{ar}, w_{sp}, w_{cr}, w_{fr}$ are pre-defined weight values in respect of the alignment rule, separation rule, cohesion rule and feature similarity rule respectively. These four simplified rules help build classifiers on different problem domains that require random search.

f. Bat algorithm

The Bat algorithm (Yang 2010) is a bio-inspired method based on the behavior of micro-bats in their natural environment. The unique behavior that characterizes bats is their echolocation mechanism. This mechanism helps bats orient themselves and find prey within their environment. The search strategy of bats is controlled by the pulse rate and loudness of their echolocation mechanism. While the pulse rate changes to improve on the position that was previously found, the loudness indicates to each other bat that the best position is accepted/found (Fister *et al*. 2014). The bat behavior has been applied in several optimization problems to find the best optimal solution. The Bat algorithm search process starts with random initialization of the population, evaluation of the new population using a fitness function, and finding the best population.

The advantage of the Bat algorithm is that the parameter control, which can vary the values of parameters as the iterations proceed. This provides a way to automatically switch from exploration to exploitation when the optimal solution approaches (Yang 2013). However, the challenge is how to speed up the convergence of the Bat algorithm to optimal solutions (Yang 2013). Secondly, there is no best control strategy that enables the Bat algorithm to switch from exploration to exploitation of a search space within a right/specified time (Yang 2013).

2.2.2. Phase 2: Data mining algorithms

Data mining is the application of an algorithm to a dataset to extract patterns or to construct a model to represent a higher level of knowledge about the data (Hand, Mannila and Smyth 2001; Kantardzic 2003). A model gives a general description of an original dataset and reflects the important characteristics of the data (Hand *et al*. 2001). The importance of a model is that it gives a clear understanding of data and helps predict new data patterns. One of the ways to predict new data patterns is by using basic mathematical formulation and translation into algorithm.

A pattern is defined as an event or combination of events that occur more or less often than expected, that is, representing a significant difference from what would be expected of random variation or representing a significant variation from a trend (Iglesia and Reynolds 2005). A pattern may be a relatively small part of the data (Hand *et al*. 2001) that could be mined. Thus, a pattern, in its simplest form, may show a relationship between two variables (Hand *et al*. 2001), which might have interesting (that is, non-trivial, implicit, previously unknown and potentially useful) information that is relevant. A model may contain patterns as well as other structures within the data. Therefore, boundaries between models and patterns are sometimes intertwined (Iglesia and Reynolds 2005).

An example of a pattern is a frequent pattern, which includes frequent itemsets, frequent subsequences and frequent substructures. A frequent itemset refers to a set of items that frequently appear together in a relational dataset, such as sugar and coffee. A frequently occurring subsequence refers to the pattern where users tend to obtain an item first, followed by another item, and then a series of items. This is a (frequent) sequential pattern (Han and Kamber 2006). A frequent substructure can refer to different structural forms, such as graphs, trees or lattices, which may be combined with itemsets or subsequences (Han and Kamber 2006). If a substructure occurs frequently, it is called a (frequent) structured pattern. Thus, mining frequent patterns results in the discovery of interesting associations in data.

The data mining algorithms reviewed include sequential pattern mining and closed sequential patterns.

i. Sequential pattern mining

A sequential pattern is a sequence with support (that is, proportion of occurrence of a sequence) not less than the minimum support threshold (Zhenxin and Jiaguo 2009). Aggarwal and Han (2014) consider sequential pattern mining as an association rule mining over a temporal relational dataset, as emphasis is placed on ordering of items. In the process of ordering items, some general principles are applied in respect to the property of sequential patterns. The property of a sequential pattern algorithm is that every non-empty subsequence of a sequential pattern must be frequent to show the anti-monotonic (or downward closure) property of the algorithm (Aggarwal and Han 2014). Thus, a pattern that is considered frequent has subsequences that are also frequent (that is, anti-monotonic). The drawback of sequential pattern mining is that it mines the complete set of frequent subsequences that satisfy a minimum support threshold (Raju and Varma

2015). Since mining frequent long sequences may contain several frequent subsequences, it leads to an explosive number of frequent subsequences for long patterns, which is computationally expensive in both time and memory space (Yan, Han and Afshar 2003).

The algorithms that have been developed based on the concept of sequential pattern mining (Agrawal and Srikant 1995) include apriori-based methods, pattern-growth methods and vertical format based methods (Raju and Varma 2015). These algorithms are described in the following subsections.

a. Apriori-based method

The apriori-based method is a level-wise approach for generating frequent itemsets in data. Basically, the principle of Apriori is that every subset of a frequent pattern is also frequent, that is, referred at downward closure. Later, all patterns are put together through the use of "joins" (Aggarwal and Han 2014). The joins enable the union of all patterns into a holistic pattern. During the implementation of the apriori algorithm, a set of patterns are generated as a candidate representation of frequent patterns and tested so as to prune unnecessary candidates or non-frequent patterns, which is often referred to as the candidate-generation-and-test strategy. Hence, apriori-based methods could be referred to as candidate-generation-and-test strategies. However, a candidate-generation-and-test strategy produces a large number of candidate sequences and requires more database scan when there are long patterns (Tu and Koh 2010). The consequence is that counting and pruning a large set involves high computational cost in terms of computational resource usage and high computational time (Raju and Varma 2015). Therefore, the major challenge with the candidate-generation-and-test strategy is high computational time and cost involved. Moreover, once frequent itemsets are obtained as output results, association rules with confidence (that is, ratio of number of occurrences that are classified and occurrences in dataset) larger than or equal to a user-specified minimum threshold (Kumar *et al*. 2007) are generated. The challenge is finding the optimal threshold, which must be set by a user (Yin *et al*. 2013), because too small a support value may produce thousands of patterns that need further filtering, whereas too big a one may lead to no findings (Yin *et al*. 2013). Although pattern compression approaches such as RPglobal and RPlocal (Han, Cheng, Xin and Yan 2007) have been used to address this problem, performing filtering also requires the use of filtering algorithms which could be costly in terms of computation cost involved. The candidate-generation-and-test strategy and user setting of minimum support and minimum confidence are the major challenge of the Apriori algorithm.

The Apriori algorithm can be summarized into the following steps:

Step 1: Find all frequent itemsets.

Step 2: Get frequent items (items whose occurrence in the dataset is greater than or equal to the minimum support threshold).

Step 3: Get frequent itemsets.

Step 4: Generate candidates from frequent items.

Step 5: Prune the results to find frequent itemsets.

Step 6: Generate strong association rules from frequent itemsets (that is, rules which satisfy both the minimum support threshold value and minimum confidence threshold value).

b. Pattern-growth methods

The pattern-growth method is based on the concept of depth-first search to generate frequent patterns from a search space or dataset. During the process of growing patterns, a frequent-pattern tree (FP-tree) is constructed based on the concept of divide and conquer (Rajasekaran and Song 2006). Thus, a pattern tree is divided into two and one is selected as the best branch. The selected best branch is further grown by mining other frequent patterns. The frequent pattern growth approach, FP-growth (Han, Pei and Yu 2000), mines frequent patterns without generating candidates of FP-trees (Yongmei and Yong 2008). The challenge is that when a user fails to specify a minimum support threshold value, it takes a longer time to mine frequent patterns. The two advantages of FP-trees are, first, that the FP-tree is a highly compressed data structure making the dataset much smaller than original dataset, thereby reducing costly database scans (Tu and Koh 2010). Secondly, it avoids candidate generation and test by combining frequent items and using a frequent pattern tree to remove unnecessary candidates (Tu and Koh 2010). The algorithms that are based on pattern growth include FreeSpan (Han, Pei, Mortazavi-Asl, Chen, Dayal and Hsu 2000) and PrefixSpan (Pei *et al*. 2001).

ii. Closed sequential pattern mining

Closed sequential pattern mining is an extension of sequential pattern mining (Lin, Hsueh, and Chang 2008). The advantage of closed sequential pattern mining is threefold. Firstly, there is an efficient use of the search space pruning technique, and it significantly reduces the number of patterns produced (Huang *et al*. 2006). Secondly, more interesting patterns are found, thus reducing the burden of a user having to explore too many patterns (Raju and Varma 2015) within the same minimum support threshold. Thirdly, it retains all information on the complete pattern set in a more compact form (Cong, Han and Padua 2005). A closed sequential pattern is a frequent sequence that has no frequent super-sequence (that is, no larger itemset) with the same minimum support threshold value (that is, occurrence frequency) (Yan *et al*. 2003). Examples of algorithms based on closed sequential pattern mining include Bi-Directional Extension (BIDE) (Wang, Han and Li 2007), Closed sequential pattern mining using a Bi-phase Reduction Approach (COBRA) (Huang *et al*. 2006) and ClaSP (Raju and Verma 2015).

Current research focuses on frequent pattern mining because data analysis problems are important in finding hidden relationships. It may be established that the data mining algorithms have been designed for a parallel computing based platform (Tsai *et al*. 2015). Furthermore, these platforms rely on machine learning-based methods to reduce computational cost in data mining algorithms. It is also established that traditional methods (e.g. neural networks, etc.) are applied to emerging problems/platforms/environments to reduce computation cost when a very large amount of data is required. This indicates that the performance in terms of computational cost needs to be improved for big data analytics frameworks to help guarantee accurate and useful information.

When mathematical models are formulated, it is possible to extract and disclose interesting patterns from frequently changed datasets to avoid user setting minimum support threshold value during the mining of patterns/rules and reduce the pruning space using a meta-heuristic search method. The following subsection reviews the association rules and meta-heuristics algorithms:

i. Association rule

Association rules are rules that steer a mining algorithm to disclose patterns from data to a user (Han and Kamber 2006). The rules whose support and confidence values are below a user-specified threshold are considered uninteresting (Han and Kamber 2006), while rules whose value are above a user-specified threshold are considered interesting.

The challenge of association rule mining could be categorized into two parts. The first is finding frequent itemsets with a support above the minimum support threshold. The second is using frequent itemsets found in the first step to generate association rules that have a confidence level above the minimum confidence threshold (Shih and Kuo 2007). Therefore, many studies on association rule mining concentrate on designing an efficient algorithm on frequent itemset discovery.

Generally, a rule is defined as a conditional statement that specifies an action for a certain set of conditions (Iglesia and Reynolds 2005). The association rule is an implication of the form $K \rightarrow P$, where precondition *K* is referred to as antecedent and the action *P* is called consequent where both *K* and *P* are frequent itemsets. The discovered rule is expressed in an “If …Then…” statement. Thus, If *K* then *P*.

Two different methods may be used to measure association rules. Firstly, the support of a rule measure is defined as the proportion of appearance in the dataset (Gupta and Sikka 2013; Agbehadji *et al*. 2016). In order words, it is the number of transactions which contains itemsets (K and P) over the number of transactions in the database. This is expressed in Equation 2.70 as

$$support\ (K \rightarrow P) = \frac{\sigma(K \cup P)}{\sigma(N)} \qquad \textit{Equation 2.70}$$

where *(N)* is the total number of transactions in a database and *(K ∪P)* is the number of transactions which contains both K and P. Frequent itemsets that are found using the support measure to generate association rules have a confidence value above the minimum confidence threshold specified by a user.

Secondly, confidence of a rule measure is defined as a ratio of the number of occurrences in K that are classified as a decision class of P over the number of occurrences in K. In other words, it is a conditional probability of the consequent given the antecedent (Gupta and Sikka 2013). This is expressed in Equation 2.71 as

$$confidence\ (K \rightarrow P) = \frac{\sigma(K \cup P)}{\sigma(K)} \qquad \textit{Equation 2.71}$$

where *σ(K)* is the number of transactions that contain *K*, while *(K ∪P)* is the number of transactions that contain *K* and *P*. A higher confidence value suggests a strong association between the items *K* and *P*.

These rules are used as quality measures to remove non-interesting rules. A rule is only considered interesting if its value is greater than or equal to the minimum support and minimum confidence criteria (Han and Kamber 2006) set by a user to reveal interesting patterns. According to Silberschatz and Tuzhilin (1995), a pattern is interesting if it is unexpected (that is, surprising to the user) and/or actionable (that is, the user can perform some action with the results to obtain value). This interestingness measure *Intm* is expressed (Ghosh and Nath 2004) in Equation 2.72 as:

$$Intm\ (K \rightarrow P) = \frac{Support(K \cup P)}{Support\ (K)} * \frac{Support(K \cup p)}{Support(P)} * \left(1 - \frac{Support(K \cup P)}{\sigma(N)}\right) \qquad \textit{Equation 2.72}$$

where *σ(N)* represents the total number in the dataset. Although this expression for the interestingness measure of rules provides patterns as interesting, it does not take into consideration the time dimension.

According to Piatetsky-Shapiro (1991), a rule is considered as interesting if its measure satisfies three basic properties. Firstly, the measure value should equal to 0 if *K* and *P* are statistically independent, that is, when *P(K and P) = P(K) *P(P)*. Secondly, the measure should monotonically increase with *P(K and P)* when *P(K)* and *P(P)* remains the same. Thirdly, a measure should monotonically decrease with *P(K)* or *P(P)* when *P(K and P)* and *P(P)* or *P(K)* remains the same.

Association rules are used for different categories of data attributes, namely categorical attributes and numerical attributes (Luna *et al*. 2011). A categorical attribute *A* of an association rule is defined as an attribute with discrete unordered domain *D*, such that *A* is a categorical attribute and *u* is a numeric value in the domain *D* of *A*. The difficulty with the categorical attribute is in terms of working with all possible numerical values. This difficulty is avoided in numerical attributes by defining a wide range of numerical values into different discrete intervals (Luna *et al*. 2011). Hong and Lee (2008) suggest partitioning numerical attributes into intervals rather than a single value on the antecedent rule. This is also supported by Luna *et al*. (2011). Srikant and Agrawal (1996) suggest an approach that partitions attributes and later maps each possible value into consecutive integers. Afterwards, a partial completeness factor, which is defined by the user, determines the completeness of each partition (Hong and Lee 2008). It is important to know the number of partitions and the interval that constitutes each partition. However, the challenge is how to minimize the level of information lost during partitioning.

The partial completeness factor measures the level of information lost (Srikant and Agrawal 1996) and the level of deviation that is tolerated with respect to finest discretization (Adamo 2001). In order words, partial completeness measures the maximum distance between rules obtained prior to partition and the closest generalization in rules obtained after partition (Srikant and Agrawal 1996). Srikant and Agrawal's (1996) approach uses equi-depth (frequency) partitioning, which minimizes the number of intervals required to satisfy the partial completeness level. The measure of partial completeness is expressed taking into consideration the support of the rule as a measure on how far apart rules are. Given the level of partial completeness desired by the user, and the minimum support, the number of partitions required (assuming equi-depth partitioning) is calculated. In order to compute a partial completeness level Q, the support of any partition with more than one value should be less than

$$minsup * \ (Q-1)/(2*n) \qquad \textit{Equation 2.73}$$

where *n* is the number of quantitative attributes. Assuming that equi-depth partitioning splits the support identically, then there should be 1/ maximum support *(s)* partitions in order to get the support of each partition to less than *s*. Thus, the number of intervals is expressed in Equation 2.74 as

$$Number\ of\ intervals = \frac{2*n}{minsup*(Q-1)} \qquad \textit{Equation 2.74}$$

where *minsup* is the minimum support, and *Q* is the partial completeness level. Partial completeness *Q* is expressed in Equation 2.75 as:

$$Q = 1 + \frac{2*n*s}{minsup} \qquad \textit{Equation 2.75}$$

where *n* is the number of quantitative attributes, and *s* is the maximum support for a partition with more than one value among all the quantitative attributes.

Although the partial completeness factor measures the level of deviation between rules, it does not take into consideration the time differences of deviation.

Railean *et al*. (2013) indicate that it is possible for rules to take into account the time difference between the antecedent and consequent of sequential rules. When this happens, an item can either occur at most once in an itemset of a sequence or occur in several itemsets of a sequence. The occurrence of rules (either complex or simple rules) in sequence is then counted, and the average value of time-closeness between the antecedent and consequent over the entire sequence is calculated (Railean *et al*. 2013). Thus, a greater average value indicates a stronger rule. Rules can be grouped into simple rules and complex rules (Railean *et al*. 2013). Simple rules are of the form $V_i \rightarrow V_n$, and complex rules are of the form $V_1V_2...V_{n-1} \rightarrow V_n$ (Railean *et al*. 2013). In this instance, all rules of the simple form $V_i \rightarrow V_n$ were combined between all V_i itemsets. For example, given the rules $A \rightarrow Y$, $B \rightarrow Y$, and $C \rightarrow Y$, complex rules are derived as $AB \rightarrow Y$, $AC \rightarrow Y$, $BC \rightarrow Y$, $ABC \rightarrow Y$. In this context, two or more items (such as *A*, *B*, *C*) are combined together on the antecedent part of the rule to produce one or more items in the consequence part of the rule.

A sequential rule $A \rightarrow B$ is defined as a relationship between two itemsets A and B that belong to the same sequence and where *B* occurs at the same time stamp $t_B \geq t_A$ (Railean *et al*. 2013). This expression helps combine the time-difference between the antecedent and consequent of a rule. Railean *et al*. (2013) are of the view that it is desirable to have rules ranked at the same level if the time difference between the items is not greater than a certain time so that rules' importance is decreased with time. This leads to the definition of closeness in time for sequential rules, as when given a time-interval ω_t and W time-window of size ω_t. Itemset A and B with time-stamps t_A and t_B are $\omega_t - close$ if $|t_A - t_B| \leq \omega_t$. When a sequential rule $A \rightarrow B$, thus $t_B \geq t_A$ where *A* and *B* are ω_t-close if $t_A - t_B \leq \omega_t$. Also, the closeness measure for the ω_t-close rule for $A \rightarrow B$ is defined as $t_B - t_A$ and $1/\sigma_t$, such that if $t_B - t_A \leq \sigma_t$ then the measure should decrease slowly, while if $t_B - t_A > \sigma_t$ then the closeness measure decreases rapidly. The parameter σ_t is used to differentiate the time in the window ω_t. Thus, the closer a rule between antecedent and consequent within [0, σ_t], the more advantageous the rule is and it is ranked at same level. Consequently, if the time stamp [σ_t , ω_t] is larger, then the rule's interest is decreased rapidly (Railean *et al*. 2013).

The closeness measure not only is used as an approach to find rules in cases where the antecedent and consequent are frequent and random in databases (Railean *et al*. 2013), but also used to penalize very frequent itemsets. Thus, the higher the antecedent and consequent value, the lower the closeness measure. The Closeness Preference interestingness measure *CP* between $A \rightarrow B$ is defined in Equation 2.76 by:

$$CP(A \rightarrow B) = \frac{C_{\omega_t,\sigma_t}(B/A)}{R(A)*R(B)} \qquad \textit{Equation 2.76}$$

where the strength of the closeness is denoted by $C_{\omega_t,\sigma_t}(B/A)$ between *A* and *B*. Thus, the closeness preference interestingness measure *CP* selects strong rules with respect to frequencies of antecedent *A* and consequent *B* in respect to the temporal closeness between itemsets of rules in databases. This is expressed in Equation 2.77 as

$$CP(A \rightarrow B) = \frac{\frac{1}{|DB|}\sum_{m=1}^{nAB\omega}\{\frac{1}{n_{\omega_t|m}}\sum_{k=1}^{n_{\omega_t|m}}[\frac{1}{n_{t_{A|k}}}\sum_{i=1}^{n_{t_{A|k}}}(\frac{1}{n_{t_{B|A_i}}}\sum_{j=1}^{n_{t_{B|A_i}}}\frac{1}{1+s^{t_j-\omega_t}})]\}}{R(A)*R(B)} \qquad \textit{Equation 2.77}$$

where *R*(AB) $=\frac{n_{AB}}{|DB|}$, n_{AB} is the number of sequences where the itemset $A \rightarrow B$ appears, and |*DB*| refers to the number of sequences in the database. The higher *R(A)*R(B)*, the lower the closeness measure's value. The parameter ω_t and *s* are defined according to user preferences. The parameter ω_t represents the maximum allowed time-distance between two itemsets to consider the rule $A \rightarrow B$ as a ω_t-close candidate. Further, *s* is expressed in Equation 2.78 by

$$s = \sqrt[(\sigma_t-\omega_t)]{\frac{1}{f(\sigma_t)} - 1} \qquad \textit{Equation 2.78}$$

where $f(\sigma_t)$is a user preference time interval.

If there is only one itemset *A* in the time-interval [0, ω_t], then $n_{t_{B|A_i}}$ is the number of *B* inside the window, which is computed in Equation 2.79 as

$$CP_{B|A_i} = \frac{1}{n_{t_{\boldsymbol{B}|A}}} \sum_{j=1}^{n_{t_{B|A}}} \frac{1}{1+s^{t_{Bj}-\omega_t}} \qquad \textit{Equation 2.79}$$

where $n_{t_{B|A_i}}$is the number of *B* between two consecutive *A* inside the same window, *s* is the slope of the plot (*s* > 1) and is directly proportional with the user-preference time-interval σ_t. However, a larger σ_t implies larger *s*, so a user may either set a threshold or use an expression to compute the threshold. The time distance of each *B* from the beginning of the window (starting from the first *A*) is represented by t_{Bj}.

Additionally, when there is only one itemset *A*, then the number of *A* inside a window ω_t is computed in Equation 2.80 as

$$CP_{\omega} = \frac{1}{n_{t_A}} \sum_{i=1}^{n_{t_A}} CP_{B|A_i} \qquad \textit{Equation 2.80}$$

Thus, the strength of $A \rightarrow B$ in a sequence is expressed as the average value of all time-windows of size ω_t, which is expressed in Equation 2.81 as

$$CP_S = \frac{1}{n_{\omega_t}} \sum_{k=1}^{n_{\omega_t}} CP_{\omega_k} \qquad \textit{Equation 2.81}$$

where n_{ω_t} is the number of time-windows of size ω_t in a single sequence containing the rule $A \rightarrow B$, and CP_{ω_k} is the closeness preference value for each window *k*.

Thus, the average strength of the rule $A \rightarrow B$ is calculated from all the sequences containing $A \rightarrow B$ to the entire database and the closeness index $C_{\omega_t,\sigma_t}(B|A)$, which shows the frequency of closeness. This is expressed in Equation 2.82 as:

$$C_{\omega_t,\sigma_t}(B|A) \;= \frac{1}{|DB|} \sum_{m=1}^{n_{BA_\omega}} CP_{S_m} \qquad \textit{Equation 2.82}$$

where $n_{BA_\omega}$is the total number of sequences where the rule $A \rightarrow B$ holds at least once in the interval [0, ω_t], and $CP_{S_m}$is the expression for each sequence *m*.

A sequential pattern in which each pattern of length *n* consists of *n - 1* rules is extracted using Equation 2.83 by Railean *et al*. (2013):

$$F_{pattern} = \frac{1}{nr_{rules\ in\ pattern}} * \sum_{k=1}^{nr_{rules\ in\ pattern}} [\frac{1}{n_{t_{A|k}}} \sum_{i=1}^{n_{t_{A|k}}} (\frac{1}{n_{t_{B|A_i}}} \sum_{j=1}^{n_{t_{B|A_i}}} \frac{1}{1+s^{t_j - \omega_t}})]$$

Equation 2.83

where $nr_{rules\ in\ pattern}$ represents the number of rules in a pattern. In order to avoid the user setting a threshold value for strong rules to be selected using trial and error, the threshold value θ^* to select rules is expressed as the average value of the weighted function *f*(t, s, ω_k) over ω_t to show the area under the curve of the function and the *x*-axis divided by ω_t. Thus, the threshold value is expressed in Equation 2.84 as

$$\theta^* = \frac{\omega_t + \frac{ln\left(\frac{s^{-\omega_t + 1}}{2}\right)}{ln(s)}}{\omega_t} \qquad \textit{Equation 2.84}$$

where θ^* represents threshold value, ω_t represents the time windows of size, and *s* represents the slope of preference. The final threshold value is used to select the rules with very close itemsets. Conversely, this threshold value may be set by the user instead of using the expression on the threshold value.

Although the approach of Railean *et al*. (2013) to finding rules that are optimal considers three parameters (time-interval, slope of preference function and threshold value), it is possible for a frequent pattern/item to also have discrete/continuous values that are frequently changed within a time interval that was not considered. The challenge is how to discover rules from frequently changed items within time-interval and numeric dimensions.

ii. Meta-heuristic search method

An aspect that has attracted the attention of researchers is the application of animal behavior (that is, bio-inspired/meta-heuristic approaches) to association rule mining or mining of frequent patterns. This was a result of the unique behavior exhibited by animals in steering movements from one location to another. A bio-inspired or meta-heuristic search strategy is based on the behavior of animals in their natural habitat. This behavior is simplified into mathematical expressions and used to form simplified rules that are applied to association rule mining. Aggarwal and Rani (2013) indicate that having good quality rules is significant in making better decisions. The rule quality is viewed in terms of results accuracy and if it is understood by the user (Mangat 2010). Meta-heuristic methods can improve the quality of rules in association rule mining (Aggarwal and Rani 2013).

Meta-heuristic algorithms may be defined as an iterative process that steers subordinate heuristic processes by combining different search strategies to explore and exploit a large search space and to develop learning strategies to structure information in order to find efficiently optimal or near-optimal solutions (Osman and Laporte 1996).

Blum and Roli (2003) outline the following fundamental properties of meta-heuristic algorithms:

- They range from the use of simplified local search procedures to complex learning processes. Although local search procedures get trapped in a search domain, meta-heuristic search methods are well adapted to avoid getting trapped, by using its random search mechanism.
- The algorithm is approximate and usually non-deterministic.
- The search methods are both in-breadth and in-depth searches to enable adequate exploration and exploitation of the search space.
- The algorithm is not problem specific. It may make use of domain-specific knowledge in the form of heuristics that are controlled by the upper-level strategy (where “upper level” refers to higher or meta-heuristics).

These fundamental properties enable meta-heuristic algorithms to efficiently explore search spaces to find optimal or near-optimal solutions within different problem domains, such as the discovery of association rules. The technique that helps with efficient exploration of search space and adaptability to different problems is the parameter tuning technique. The advantage of parameter tuning is that it helps assign different weighting parameters to search problems in order to find the best parameter that fits a problem.

Meta-heuristic algorithms are generally divided into three categories: evolutionary, swarm intelligence and differential evolution algorithms. The first is evolutionary algorithms, which are inspired by evolutionary theory and biological evolutionary processes such as selection, recombination, mutation and reproduction. Evolutionary algorithms include GAs (Goldberg and Holland 1988); evolutionary strategy algorithms (Hansen and Ostermeier 2001); swarm intelligence algorithms, which are inspired by the movements of large groups of animals; and differential evolution algorithms (Storn and Price 1997; Al-Ani, Alsukker and Khushaba 2012). Among these categories, evolutionary computation was shown to enhance accuracy and reduce computation time (Abdullah *et al*. 2013).

The best-known swarm intelligence algorithm is the particle swarm optimization (PSO) algorithm (Kennedy 2011), which simulates the movement of animals to improve the capability of global searches. Other swarm optimization algorithms are the ant colony algorithm (Bonabeau, Dorigo and Theraulez 2000), bee algorithm (Karaboga 2005), Firefly algorithm (Yang 2008) and WSA (Tang *et al*. 2012). Meta-heuristic methods are able to extract more concise and accurate information on association rules than the conventional Apriori-based algorithm.

Among the most popular meta-heuristic approaches to association rule mining are GAs (Goldberg and Holland 1988) and the ant colony algorithm (Dorigo and Cambardella 1997). Other meta-heuristic approaches include Particle Swarm Optimization (Kennedy and Eberhart 1995) and the WSA (Agbehadji *et al*. 2016). Agbehadji *et al*. (2016) propose the WSA for numeric association rule mining by using simplified rules from wolves' behavior to discretize numeric values into intervals without considering the aspect of time dimension. Agbehadji *et al*. (2016) also propose a bio-inspired algorithm (based on kestrels' behavior) for mining frequently changed patterns in large datasets. Moslehi *et al*. (2011) apply ant colony optimization on continuous data based on Gaussian functions to build numeric intervals for numeric attributes without indicating the minimum support and minimum confidence. Aggarwal and Rani's (2013) approach used probabilistic methods to scan a database for frequent datasets and produce optimized results of the Apriori algorithm using the ACO algorithm so as to enhance the quality of rules and remove unnecessary rules.

A GA, as explained previously, is an adaptive search procedure (Holland 1975 cited Agbehadji 2011). The search procedure helps find approximate missing values (Abdella and Marwala 2006) by optimizing an objective function/fitness function in any given search problem. The fitness function value of each generation is calculated in order to find the leading factor that determines the ability of a GA to find the optimal solution (Priya and Kuppuswami 2012). The adaptive search process has been applied to solve problems of association rules without setting minimum support and minimum confidence values. Qodmanan, Nasiri and Minaei-Bidgoli (2010) take a multi-stage approach that first finds frequent itemsets and then extracts association rules from those itemsets. The approach combines the Frequent Pattern (FP) tree algorithm and GA to form a multi-objective fitness function with support, confidence thresholds and the ability to obtain interesting rules. This approach enables a user to change the fitness function so that the order of items is considered in the importance of rules.

Another meta-heuristic algorithm, called PSO, was proposed by Kennedy and Eberhart (1995). PSO is a bio-inspired method based on swarm behavior, such as fish and bird schooling in nature (Kennedy and Eberhart 1995). The swarm behavior is expressed in terms of how particles adapt and make a decision about changing position within a space based on the position of other neighboring particles. The advantage of swarm behavior is that as individual particles make decisions, it leads to an emergent behavior (Krause *et al*.

2013). This emergent behavior is a result of local interactions among particles in a problem space.

Among the particle swarm algorithms for finding best possible solutions in a problem space are the Firefly algorithm (Yang 2008), Bats (Yang 2009) and the bee algorithm (Karaboga 2005). The successful characteristic of the firefly is the short and rhythmic flashes it produces (Yang 2008). This flashing light is used as a mechanism to attract mating partners, attract potential prey and serve as a warning to other fireflies. The signaling system of this flashing light mechanism is controlled by simplified basic rules underlying the behavior of fireflies. Unlike a GA, which uses operators such as mutation, crossover and selection, the Firefly algorithm uses attractiveness and brightness to improve certain individuals in its population. The similarity between GAs and the Firefly algorithm is that both generate initial populations and continue to update their initial population using fitness functions. The brighter fireflies attract those closest around them, and the fireflies whose flashes fall below a given threshold are removed from the population. The brightest fireflies form the next generation, and the generations/iterations continue until a select criterion is reached or a maximum number of generations is reached. The behavior where a bright firefly attracts another firefly with a weaker brightness has been applied in missing data imputation by finding estimates of values closest to known values and then replacing these missing values with these estimates (Agbehadji *et al*. 2018).

The behavior of the bee is that it waits in the dance area in order to make the decision to choose its food source (Karaboga 2005). There are different kinds of bees, including onlooker bees, scout bees and employed bees. The bees carrying out random searches are scout bees. The onlooker bees and scouts are also called unemployed bees, while employed bees are bees that re-visit their food source for further exploration (Karaboga 2005). Bees are adapted to self-organize to enrich their food source (which could result in positive feedback) and to discard poor sources by scouting (causing negative feedback)

(Karaboga 2005). However, the process of discarding poor sources leads to premature convergence to optimality (Yan and Li 2011).

Similarly, the behavior of particle swarms has also been applied to discrete optimization problems using binary PSO to find association rules. Sarath and Ravi (2013) have formulated a discrete/combinatorial global optimization approach that uses a binary PSO to mine association rules without specifying the minimum support and minimum confidence of items, unlike the Apriori algorithm. The quality of the rules is evaluated in term of fitness function, expressed as the product between the support and the confidence. The fitness function ensures the support and confidence are binary between 0 and 1. The proposed binary PSO algorithm consists of two parts: the preprocessing and the mining. The preprocessing part calculates the fitness values of the particle swarm so as to transform the data into binary format. The mining part of the algorithm uses the PSO algorithm to mine association rules. Sarath and Ravi (2013) indicate that binary PSO can be used as an alternative to the Apriori algorithm and the FP-growth algorithm as it allows the selection of rules that satisfy the minimum support threshold. However, if time is significant in finding how close the rules are, then the binary PSO is disadvantaged.

The PSO has also been applied in high utility itemset mining to identify a high profit itemset, especially when that itemset rarely appears but has high profit value (Lin *et al*. 2016). The concept of high utility was designed to discover the “useful” and “profitable” itemsets from quantitative databases. Thus, an item is of high utility if its utility value is no less than the user-specified minimum threshold for items to be mined (Lin *et al*. 2016). The concept of bio-inspired methods can be used to explore the search space for an optimal solution.

Kuo, Chao and Chiu (2011) apply the PSO algorithm and binary data transformation technique (Wur and Leu 1998) to search for association rules when they are likely to produce large sets of rules. The advantage of using the binary transformation is that it improves the computational efficiency of the search algorithm. However, in cases where a large range of rules is generated, the itemset range (IR) method is applied to search within that range of rules in order to decide on the length of rule generated. The itemset range is mathematically expressed in Equation 2.85 by

$$IR = log\big(mTransNum(m)\big) + log\big(nTransNum(m)\big)\frac{Trans(m,n)}{TotalTrans} \qquad \textit{Equation 2.85}$$

where *m*$\neq$*n*, *m*<*n*, *m* represents the length of the itemset, *Trans(m)* represents the number of transaction records containing *m* items, *n* is the length of itemsets, *Trans(n)* represents the number of transaction records containing *n*, *Trans(m, n)* represents the number of transactions containing *m* and *n*, and *TotalTrans* represents the total transaction.

Dorigo and Cambardella (1997), propose another meta-heuristic algorithm called ant colony optimization (ACO). Ant colony optimization (ACO) (Dorigo and Cambardella 1997) is a meta-heuristic inspired by the foraging behavior of real ants in their search for the shortest paths to food sources. When a source of food is found, ants deposit a pheromone to mark their path for other ants to traverse. A pheromone is an odorous substance that is used as a medium of indirect communication between ants. The quantity of pheromone depends on the distance, quantity and quality of the food source (Al-Ani 2007). However, the pheromone substance tends to decay or evaporate with time, which prevents ants from converging to sub-optimal positions (Stützle and Dorigo 2002). When a lost ant, moving at random, detects a laid pheromone, it is likely that it will follow the path to reinforce the pheromone trails. Thus, ants make probabilistic decisions on updating their pheromone trail and local heuristic information (Al-Ani 2007) to explore larger search areas.

ACO has been applied to solve many optimization-related problems, including data mining problems, where it was shown to be efficient in finding best possible solutions. In the field of data mining, an important issue for association rule generation is frequent itemset discovery, which is an important factor in implementing association rule mining. Kuo and Shih (2007) propose a model that uses the ant colony system to find the best global pheromone before the generation of association rules. The model proposed by Kuo and Shih was applied to a health database where a user specifies more than one attribute and defines two or more search constraints on an attribute for association rule generation. Constraint-based mining enables users to extract rules of interest to their needs, and computational speed is faster, thus improving the efficiency of mining tasks. Kuo and Shih (2007) indicate that the constraint-based mining provides condensed rules, contrary to the Apriori method. Additionally, the computational time was reduced since the database was scanned only once to disclose the mined association results. The use of constraint conditions reduces search time during the mining stage. However, the challenge is how to merge many similar rules that are generated in the mining results.

Kuo and Shih (2007) apply ACO to find association rules (between potential disease and early prevention in a health database) from a multi-dimensional constraint problem. In their study, multi-dimensional items were classified into two constraints, namely a single constraint against two constraints (such as, $max(X, cost) \leqslant (X, price)$) and a conjunction or disjunction of multiple sub-constraints $(C1{:}\ X, cost \leqslant v1) \wedge (C2{:}\ X, price \leqslant v2)$, where $v1$ and $v2$ are constant values respectively (Kuo and Shih 2007). Besides these two multi-dimensional constraints, a frequency constraint is considered to find useful/interesting rules. The advantage of ACO over the Apriori method is that ACO condenses more rules and uses less computational time to generate association rules.

Tang *et al*. (2012) propose another meta-heuristic algorithm called the WSA. The WSA is a bio-inspired heuristic optimization algorithm that is based on wolf preying behavior (Tang *et al*. 2012). The behavior of wolves includes the ability to hunt independently by remembering their own traits (meaning wolves have memory); the ability to only merge with their peer when the peer is in a better position (meaning there is trust among wolves to never prey on each other); only being attracted to prey within their visual range; the ability to escape randomly upon the appearance of a hunter (Tang *et al*. 2012); and the use of scent marks as a way of demarcating their territory and communicating with any other wolf (Agbehadji *et al*. 2016).

This behavior expressed by the wolf enables it to randomly adapt to its environment when hunting. If a wolf finds a new, better position, the incentive is stronger to assume this new position, provided that the position is not already inhabited by a companion wolf. Furthermore, each wolf instinctively flocks together in a pack that is a collective, and organizes individual searches of an individual wolf. Therefore, the swarming behavior of the WSA is delegated to each individual wolf, and this behavior could form multiple leaders swarming from multiple directions towards the best solution rather than a single flock searching for an optimum in one direction at a time (Tang *et al*. 2012). The WSA is an iterative search process that starts with the setting of initial parameters, random initialization of the population, evaluation and updating a current population using a fitness test, and continuing to create new generations/iterations until some stopping criteria are met. A variant of WSA is the Wolf Search Algorithm with Minus Step Previous (WSA-MP) (Tang *et al*. 2012). The WSA-MP allows a wolf to remember a previous best position and avoid the old positions that do not produce a best solution.

The Bat algorithm (Yang 2010) is another kind of meta-heuristic algorithm and is based on the behavior of micro-bats in their natural environment. The unique behavior that characterizes bats is their echolocation mechanism. This mechanism helps bats orient themselves and find prey within their environment. The search strategy of bats is controlled by the pulse rate and loudness of their echolocation mechanism. While the pulse rate changes to improve on the position that was previously found, the loudness indicates to each other bat that a best position has been accepted/found (Fister *et al*. 2014). Bat behavior has been applied in several optimization problems to find the best optimal solution.

The Bat algorithm search process starts with random initialization of the population, evaluation of the new population using a fitness function, and finding the best population. Unlike the Wolf algorithm, which uses attractiveness of prey to govern its search, the Bat algorithm uses the pulse rate and loudness to control the search for an optimal solution.

These experiments on meta-heuristic algorithms were conducted using pre-fixed parameters that were pre-tuned or controlled by the bio-inspired behavior of the respective algorithm at each iteration process (Wei-yong *et al.* 2015). Although Kuo and Shih (2007) and Kuo *et al*. (2011) apply meta-heuristic algorithms to different problem domains, neither study considers the time and numeric dimension of frequently changed itemsets. It is possible to apply a meta-heuristic algorithm as a search strategy to perform randomization of search space to select optimal results when large volumes of data and velocity of frequently changed items are important for pattern discovery. When this is achieved, it is possible to improve on the runtime performance (Oweis *et al*. 2016) of searching for association rules in large datasets. Yang, Lin and Jin (2015) indicate that randomization is faster and reduces data size when large volumes of data are involved. Furthermore, Yang, Lin and Jin (2015) observe that having optimal results improves the convergence rate of data when speed (velocity) is significant. The combination of the benefits of randomization and approach to optimal solutions provides an efficient search algorithm (Yang, Lin and Jin 2015). It is possible to relate these properties of data attributes to the frequently changed or frequently used aspects of data. Thus, exploring the frequently changed or frequently used attributes through randomization and optimal solutions may provide interesting patterns. These interesting patterns may be provided through animal behavior, as animals can learn from their environment and adapt to different situations (such as in association rule mining). In the next subsection, the learning behavior of animals is discussed.

iii. Learning behavior of animals

Learning is essential for social animals as it enables them to adapt to different environments in their search for food. During the process of adapting, social animals tend to imitate best behavior that enables it to survive. Social imitation learning is a skill used to achieve social adaptation, and the more an individual animal observes a behavior, the more likely it is to imitate that behavior and later engage in similar action. This suggests that each animal uses the social learning concept (Bandura 1971), which states that learning may be achieved by observing behavior, encoding best features and later imitating previously observed behavior. The key factor that could possibly lead to imitation is distance, as the closer the distance, the higher the chances of accurate imitation. Animals that are cooperative in hunts or that hunt in groups, are possibly adapted with natural abilities that allow them to adopt the social learning concept.

Sakato, Ozeki and Oka’s (2012) method of imitating actions of well-adapted individuals within an environment uses the reinforcement learning method, which not only allows individuals to imitate best actions but also to behave autonomously in their environment (Sakato, Ozeki and Oka 2013). In this method, as an individual (referred to as an agent) imitates similar actions of another individual, a similarity value is computed to show the importance of features that were imitated in terms of the type of feature and location. Imitated actions are then selected using an action selection module. Agents learn through trial and error in a given environment what to imitate and how to evaluate the imitation until the correct value is determined and then maps it to a corresponding feature. Function approximates are used to calculate approximate value, which represents the similarity value of an action and the observed action.

Although this approach uses a probability in calculating the similarity between an action and the observed action, it is possible to use a meta-heuristic search method that is adapted with randomization and optimization search strategies to imitate optimal or near-optimal solutions.

2.2.3. Phase 3: Data visualization

Data visualization presents data in pictorial or graphical format and enables the display of interesting patterns for decision-making (Bikakis 2018). Han and Kamber (2006) provide the following as reasons for data visualization: Gain insight into information by mapping data onto graphical form; provide a qualitative overview of large datasets; search for patterns, trends and relationships among data; and help find interesting patterns for decision-making. Additionally, data visualization avoids data distortion by showing actual data using pictures for easy understanding (Ward, Grinstein and Keim 2010). In view of these reasons, data visualization faces the challenge that billions of records are condensed into a million pixels (Shneiderman 2008), and this results in distortion of data. Although distortion is a challenge in data visualization, the focus of the present thesis is on the representation of data in a systematic form with data attributes and variables, which helps display patterns in graphical format. The conventional approaches that are used to represent data in systematic format include the use of bar charts, scatter diagrams and maps (L. Wang, G. Wang and Alexander 2015).

The conventional techniques of visualization that tie in with data visualization could consider visual performance scalability and response time during the visual analytics process (L. Wang, G. Wang and Alexander 2015). Among the techniques of visualization are dense pixel display, the stacked display technique (Keim 2000; Keim 2002; Leung, Kononov, Pazdor and Jiang 2016) and cellular ant-based methods (Moere *et al*. 2006).

According to Keim (2000), the dense pixel technique maps the dimensions of values of both text and numeric data to colored pixels and then groups the pixels belonging to each dimension into adjacent areas using the circle segments technique. The principle of the circle segments technique is that, close to a center, all attributes close to each other enhance the visual comparison of values. The stacked display technique (Keim 2002; Leung, Kononov, Pazdor and Jiang 2016), meanwhile, displays sequential actions in a hierarchical fashion. The basic idea of the stacked display technique is to integrate one coordinate system with another, that is, two attributes form the outer coordinate system and two other attributes are integrated into the outer coordinate system.

When coordinate systems are integrated, it is possible to update these coordinate systems to be dynamic. Mittelstaedt and Mittelstaedt (1982) express this integration as summing up or adding coordinates together. Thus, positions on the coordinate systems are updated using the Cartesian system, where each rotation is categorized into two orthogonal (sine and cosine) components, and then the components are integrated (summed up) over the total path (Mittelstaedt and Mittelstaedt 1982). Thus, it is possible to apply path integration as a method of updating coordinate systems to be dynamic. Path integration is an incremental recursive process in which changes in current estimates of the position are added to position vectors of previous steps (Etienne, Maurer and Saucy 1988) through the use of rules for processing information (Etienne and Jeffery 2004). However, the major drawback is that computational performance degrades due to cumulative error after each update process.

2.2.3.1 Bio-inspired approach to data visualization

The animal behavior/bio-inspired approach to visualization includes the use of cellular ants based on any colony system (Moere *et al*. 2006) and flocking behavior of animals. Flocking behavior for data visualization (Moere 2004) focuses on simplified rules that model the dynamic behavior of objects in *n*-dimensional space. The spatial clustering technique helps group each dynamic behavior or similar features of data as a cluster that is viewed in *n*-dimensional space on a grid. In order to assist users of the visual data to understand patterns, a blob shape is used to represent groups of spatial clusters. This blob shape represents data plotted on grids.

Moere *et al*. (2006) combine characteristics of ants and cellular automata to represent datasets in visual clusters. The cellular ants use the concept of self-organization to autonomously detect data similarity patterns in multi-dimensional datasets and then determine the visual cues, such as position, color and shape size, of the visual objects. A cellular ant determines its visual cues autonomously, as it can move around or stay put, swap its position with a neighbor, and adapt a color or shape size where each color and shape size represents data objects. Data objects are denoted as individual ants placed within a fixed grid that creates visual attributes through a continuous iterative process of pairwise localized negotiations with neighboring ants in order to form a pattern that can be visualized on a data grid. When ants perform continuously pairwise localized negotiation, the position of one ant is swapped with another ant, which relates to swapping one color with another in a single cluster (Moere *et al*. 2006). In this instance, the swap in positions relates to interchange between data values that are plotted on a grid for visualization by users.

Generally, there is no direct predefined mapping rule that interconnects data values with visual cues to create the visual format for users (Moere *et al*. 2006). Hence, the shape size scale adjustments are automatically adapted to data scale in an autonomous and self-organizing manner (Moere *et al*. 2006). In view of this, instead of mapping a data value to a specific shape size, each ant in the ant colony system maps one of its data attributes onto its size by negotiating with its neighbors. During the shape size negotiation process, each ant compares randomly the similarity of its data value and circular radius size that is measured in screen pixels. It is possible that each ant can grow large or become small, therefore simplified rules from ant behavior are expressed and applied to check how ants can grow in their neighboring environment. These rules are significant in determining the scalability of visualized data, whereas the randomization process is significant in determining the adaptability of data value. The process of shape size scale negotiation may require extensive computational time in coordinating each ant into a cluster or single completed action.

2.2.3.2 Data visualization evaluation techniques

Keim, Bergeron and Pickett (1994) indicate that data visualization evaluation techniques give an idea that could lead to improvement on data visualization methods. Although there is a lack of quantitative evidence measuring the effectiveness of data visualization techniques, Keim *et al*.'s (1994) approach to quantitatively measuring the effectiveness of visualization techniques was by generating arbitrary/artificial test datasets with similar characteristics such as different structures or semantics. When similar characteristics are grouped, statistical methods are used as a way of evaluation, and include the use of mean and variance of some dimensions such as location, size and shape of clusters. When some parameters (such as statistical parameters) that define the data characteristics are varied at a time within an experiment in a controlled manner, it helps in evaluating different visualization techniques to find where the point data characteristics are perceived for the first time or to find the point where characteristics are no longer perceived, in order to build more dynamic and realistic test data

Another approach to evaluating data visualization was proposed by Keim *et al*. (1994). They indicate that data for visualization can be evaluated when the same test data is used in comparing different visualization techniques so as to determine the strengths and weaknesses of each technique. However, the limitation of the visualization approaches is that the evaluation is based only on users' experience in the use of the visualization techniques. Marghescu (2008) notes that the effectiveness of a visualization technique is based on the user's ability to read, understand and interpret the visual display easily, accurately, quickly, et cetera. Thus, effectiveness depends not only on the graphical design but also on the users' visual capabilities (Marghescu 2008). Card, Mackinlay and Shneiderman (1999) define effectiveness as the capability of a human to view a display well, interpret the results faster and convey distinctions in the display with fewer errors. Mostly, effectiveness is measured in terms of time to complete a task or quality of the tasks' solutions (Dull and Tegarden 1999; Risden and Czerwinski 2000).

Some visualization evaluation techniques include observation by users, the use of questionnaires, and graphic designers who critique visualized results (Santos 2008) and give an opinion. Although these visualization evaluation techniques are significant, they are subjective and qualitative, so a quantitative approach could provide an objective approach to measure visualization evaluation techniques.

2.3 Summary

This chapter provided a review of methods/concepts relating to data cleansing, extrapolating missing values from data sources, and data mining algorithms/big data mining algorithms and their challenges. Challenges with current data mining algorithms are the high computational time and cost, the user setting a minimum support threshold value, and not considering the time and numeric value dimension in frequent pattern/item discovery. Appendix 1 is a summary of the data mining algorithms, advantages and limitations.

These challenges (as shown in Appendix 1) are the gaps that were identified, and although different search strategies have been applied to different problem-specific domains to find frequent patterns/items in respect of data mining algorithms, these applications have failed to disclose frequently changed patterns/items that might have interesting patterns based on which a user can take a sequence of actions. Additionally, challenges identified with duplicate detection techniques include the inability to perform global alignment of words (that is, using the Smith-Waterman algorithm) and the computational complexity of duplicate detection algorithms (that is, the using Smith-Waterman algorithm). Also, while the BLAST algorithm for duplicate detection takes less time, it may not guarantee accurate results as compared with the Smith-Waterman algorithm. Although, Naumann’s (2013) framework presents a process to be followed for duplicate detection, when the algorithm used in the process is unable to find accurate duplicates from any data source it influences the performance results. Hence, the algorithm used aspect will be the focus of this thesis.

Additionally, Appendix 2 shows a summary of the advantages and disadvantages of meta-heuristic search methods.

After reviewing *what* the key issues of concern are in respect of the challenges, it is important to know *how* to address these challenges, which still remains a gap. This thesis seeks to fill the gaps (Appendix 1 and Appendix 2) and propose a method on *how* to address the challenges by proposing a search algorithm for cleansing data and a data mining algorithm for frequently changed patterns/items that are characterized as having volume, velocity and value. Finally, the study also proposes an approach for data visualization in order to address scalability issues with data visualization when data is characterized as having velocity (that is, moving with speed).

CHAPTER 3: DEVELOPING METHODOLOGICAL FRAMEWORK

3.1 Introduction

This chapter illustrates how the research gaps identified in the previous chapter will be addressed in this thesis. In this chapter, the researcher applies mathematical formulation as a method to model certain behaviors of selected animals and then translate this model into algorithms. The modelled behaviors are, from the researcher's perspective, the dominant features that represent unique characteristics that distinguish each animal. The unique characteristics that are mathematically expressed were translated into search algorithms (referred to as agent-based search algorithms) to find best possible solutions in different problem domains that are characterized as having large volumes and velocity. This agent-based search algorithm can be adapted to different problem dimensions and domains by parameter tuning to observe the results of each parameter. The next subsection discusses the methodological framework that helped guide this study and helped address the gaps identified in literature.

3.2 Methodological framework for big data analytics

Research methodology is defined as a way to collect data and to analyze and interpret results to either confirm or reject a claim (Creswell 2013). Creswell (2013) notes that there are four questions that help a researcher clearly understand a problem in a research design. The four questions are: What theory of knowledge is embedded in the theoretical perspective (referred to as epistemology) that informs the research (either objectivism or subjectivism)? What is the theoretical perspective (that is, what lies behind the methodology in question, e.g. positivism and post-positivism, interpretivism or critical theory)? What is the methodology (that is, strategy or plan of action that links methods to outcomes) that governs the choice and use of methods (e.g. experimental research, survey research, ethnography, etc.)? And what methods (that is, techniques and procedures) are used (e.g. questionnaire, interview, focus group, etc.)?

This philosophical process of knowledge claims are linked to four different schools of thought, namely positivism, constructivism, advocacy/participatory and pragmatism. The positivist knowledge claim helps make underlying knowledge claims that form a basis to understand the nature of a problem in the real world (Creswell 2013). This enables the researcher to formulate a strategy and appropriate method to solve the problem (Creswell 2013). Knowledge claims use existing theory; strategy is the use of experimental design; and method is a way of measuring output (Creswell 2013). Given the nature of the study, the researcher adopted objectivism as epistemology (as the researcher deals with objective reality), positivism as theoretical perspective (as ideas are reduced into small subsets that can be tested, such as variables in hypotheses and theories, and the effects that these variables cause are identified), and experimental design as methodology (because with this design, it is possible to control numeric attributes and measure their causal effect) (Creswell, 2013).

Using this methodology, the researcher followed the premise of this study, which is that data is frequently generated from different sources that are characterized as having velocity, volume, value and variety. As data is generated, its characteristics might frequently change and, based on this possibility, the thesis focuses on the aspect of disclosing interesting patterns from frequently changed data and on the visualization of these patterns. The proposed methodological framework consists of three phases: data preprocessing/data cleansing, data mining and data visualization. Table 3.1 shows the various phases, stages, proposed algorithms and the comparative algorithm that constitute the methodological framework.

Table 3.1: Phases, stages and algorithms

Phases	**Stages**	**Proposed algorithm**	**Comparative algorithms**
Phase 1: Data cleansing/preprocessing	Stage 1: Identify and eliminate duplicate text	Enhanced Smith-Waterman algorithm	Jaro-Winkler distance metrics
	Stage 2: Extrapolating missing data values	KSA	WSA-MP, BAT and Firefly algorithms
	Stage 3: Feature selection	KSA	WSA-MP, BAT, ACO and PSO algorithms
Phase 2: Data mining	-	KSA	ACO, BAT, PSO and WSA-MP algorithms
Phase 3: Data visualization (using linear graph)	-	DBA	ACO for data visualization, Bee algorithm

Source: Researcher

Table 3.1 consists of three phases. Phase 1 is data preprocessing/data cleaning, which consists of three stages, namely: extrapolating missing data; identifying and eliminating duplicates; and feature selection. The extrapolating stage estimates missing values at random in the observed data sample, as this may create inaccurate results in analysis. The method to extrapolate missing data was mathematically formulated from the random search characteristics of a bird in order to build a search algorithm that was used to extrapolate missing data. Duplicate text data may create inaccurate results, which is identified and eliminated using the Naumann's (2013) framework and the enhanced Smith-Waterman algorithm is applied. Finally, the feature selection stage allows large volumes of data to be narrowed/reduced into smaller sets of relevant frequently changed data for a quick and easy feature selection process. The formulated mathematical expression for feature selection was combined with a deep learning network to form a classifier for feature subset selection. Although there are several traditional search methods (such as ANN, SVM, etc.) for building classifiers for feature selection, the formulated search strategy was adopted as it related to how the bird being modelled shows a dominant behavior trait, such as maintaining a still position with eyes fixed in a forward-looking direction to search through a series of overlapping areas (Shrubb 1982).

The distinction between the proposed methodological framework and other framework (e.g. Srivastrva 2014) differs in terms of the approach adopted in each stage, such as data preprocessing or data mining. Even the various data mining frameworks, devoted to one phase of the data management process, lack uniformity (Khana, Mohamudally and Babajee 2013). Consequently, a comparison of the two frameworks or of any existing framework are not considered and, hence, benchmark testing proposed against existing frameworks are not focus of this thesis. The proposed methodological framework largely focuses on a bio-inspired behaviour modelling and implementation of KSA but to a minor extent, of the dung beetle as well in the stages of preprocessing, data mining, and visualisation. Other frameworks do not include these phases and, hence, this proposed framework forms an original contribution in this thesis.

Phase two, data mining, uses the KSA to mine association rules to disclose interesting frequently changed patterns within time and numeric dimensions. The interesting patterns are measured in terms of their closeness to a user-specified time interval (Railean *et al*. 2013), which is discussed in section 3.3.2.

The third phase is the data visualization. This phase is used to view the data mining results using a simple linear graph with low computational cost. The DBA, which is based on ball rolling, dance and path integration behavior of dung beetles, was used as an approach to data visualization. The approach was mathematically formulated into simplified mathematical expressions (as indicated in section 3.4) for visualization of relevant and interesting frequently changed items with numeric value from the data mining phase.

In addition, the selected comparative algorithms that were described and discussed in literature review, as indicated in Table 3.1, were compared with the proposed bio-inspired algorithm to test the robustness of the algorithm proposed in this thesis. Moreover, the appropriate comparative meta-heuristic algorithms were selected for each phase of the proposed computational model.

In the next sub-section, the behavior of the bird (the kestrel) is discussed, as well as some of its unique characteristics, the mathematical formulation depicting these characteristics, the assumptions underlining the mathematical formulation and the formulation of basic rules from each characteristic identified, as these constitute the steps in formulating the KSA algorithm. Similarly, these steps are applied in the formulation of a search algorithm for the proposed DBA for data visualization.

3.2.1 Description of kestrel behavior

The kestrel is a bird that hunts by hovering (that is, flight-hunting) or from a perch. This bird can defend its territory from other kinds of birds and can change its hunting technique based on type of prey, prevailing weather conditions (such as wind) and energy requirements (for gliding). Kestrels are well adapted to use their eyesight to watch small and agile prey on the ground. When kestrels are hovering, they can maintain a still position with forward-looking eyesight to encircle prey beneath. Frequently bobbing their head gives them a degree of magnified or binocular vision that helps in judging distance to locate prey from a remarkably far distance. Kestrels have an ultraviolet-sensitive characteristic to visually locate trails of prey such as voles, because trails of urine and feces reflect ultraviolet light, making them visible to kestrels (Viitala *et al*. 1995).

Most frequently, an individual kestrel hunts a portion of an area and shifts entirely to another portion at regular time intervals to look for prey (Shrubb 1982). When kestrels live as social birds in a flock, it improves their chances of finding food sources, the risk of predation reduces and their roost serves as a place for communicating with others.

Although a group of kestrels can be seen moving together in a similar direction and staying as a group, this behavior is only temporary, since each will disperse to hunt individually (Zyl 2013).

Varland (1991) indicates that during a hunt, kestrels are imitative rather than cooperative due to their solitary hunting in close proximity to each other. This suggests that kestrels prefer not to communicate with each other but rather imitate the behavior of other kestrels with better hunting techniques and improve on their hunting technique, even though the hunting technique can change based on type of prey, prevailing weather conditions and energy requirements (for gliding or diving) (Vlachos *et al*. 2003).

During flight-hunting, the kestrel performs a random search, either by a series of hovers in a close pattern around a small area and then moving forward, or by a series of more widely spaced single hovers (Shrubb 1982). Village (1990: 66) indicates that "kestrels have incredible co-ordination required to maintain position in a constantly changing airstream. While the winds of kestrel and body are buffeted about like a flapping rag, its head stays fixed, as if pinned by invisible clamps." This suggests that kestrels are able to maintain a still position with eyes fixed pointing in a specific direction. Kestrels are able to flap their wings and adjust their long tails to stay in place, referred to as a still position in changing airstreams. This enables kestrels to search for prey in wide circles centered beneath them (Shrubb 1982). Ákos *et al*. (2010) observe that birds can adjust their flight to weather conditions using speed-to-fly theory – a theory proposed by Paul MacCready. It may be inferred that when speed is approximately zero, then a stable position can be observed.

Specifically, during perched hunting, kestrels are seen on high fixed structures constantly scanning the ground with their eyes. The benefit of a fixed high perch area is that it enables fairly large scans or use of a larger search area than low perch areas. The frequent bobbing of the head characterizes the behavior in perch. Shrubb (1982) observes that kestrels in perch tend to look forward and search series of overlapping bands. When prey is found, kestrels either glide down to strike directly or fly out and hover over the spot area for a closer look and determination of the measurement of distance to travel. Mostly, the strike from kestrels, while in perch mode, is directed at small mammals closer to a perch than avian prey. This suggests that in perch mode, kestrels conserve some energy and direct their ultraviolet-sensitive characteristics at slowly moving prey on the ground and lift the prey. During active hunting, that is, in either hovering and perch mode, kestrels maintain upright posture, continually scan the ground and bob their head several times to better judge distance to prey.

Kestrels are mostly known to flight-hunt if they are looking for small mammals and if the wind conditions are favorable. Thus, an increase in flight-hunting leads to a corresponding decline in perch-hunting, which reflects the capture rate in that the perch-hunt capture rate is lower. Flight-hunting is necessary if the yield from perch-hunting is too low, but this is usually done when the wind conditions make hovering most efficient. Perch-hunting is therefore a good way to meet daily food needs, as it requires little effort. Flight-hunting yields more strikes than perch-hunting, however, because flying kestrels are higher (searching directly overhead) and can move more rapidly, thereby scanning more ground in a given time (Village 2010). The characteristics of the kestrel are simplified as follows:

1) Soaring: Gives a larger search space (global exploration) within the visual coverage area.
    a. separates from group assembly of kestrels in order to hunt individually
    b. still (motionless) position with eyesight fixed on prey
    c. encircles prey beneath with keen eyesight
    d. stepped descent to capture prey by surprise

2) Perching: Each kestrel does a thorough search (local exploitation) within the visual coverage area.

a. positioned on high fixed structures
b. frequent bobbing of head
c. attracted to prey using seen trail, then glides to capture
d. imitation of behavior

3.2.1.1 Kestrel-based search algorithm

The KSA is a meta-heuristic algorithm based on the hunting characteristics of the kestrel. The KSA is a meta-heuristic algorithm because the hunting behavior, as described, performs local exploration of domain-specific knowledge in the form of heuristics, which are controlled or guided by the upper-level search strategy (referred to as meta), global exploitation. The learning strategy of kestrels is mathematically expressed and used to structure information to find efficient near-optimal solutions. The behavior of the KSA relates to the fundamental properties of meta-heuristic algorithms as defined by Blum and Roli (2003).

The hunting technique of kestrels changes based on type of prey, prevailing weather conditions and energy requirements (for gliding or diving) (Vlachos *et al*. 2003). In order to perform local exploration, the solution algorithm generates initial solutions and then tries to update with a better solution from a population. The challenge of local exploration is avoiding getting stuck in a local optimum (Iglesia and Reynolds 2005), which may be avoided by a random encircling formulation and the mechanism of trail evaporation strategy that is random.

When hovering, kestrels perform a wider search (global exploration) across territories within a visual circling radius, maintain a motionless position with forward-looking eyes fixed on prey, and detect minute air disturbances from flying prey (particularly flying insects), which gives an indication to capture prey, and mostly move with precision through changing airstreams. While in perch, mostly from high fixed structures, kestrels perform a thorough search (local exploitation) of their local territory with fewer energy requirements, and they use their ultraviolet-sensitive capabilities on mammals, such as voles, closer to perch area. Mostly, strikes are directed at mammals closer to the perch. This suggests that in perch, kestrels conserve some energy and direct their ultraviolet-sensitive capabilities at slowly moving prey on the ground. It is significant to combine this search behavior because perch-hunting is thorough and directed at a variety of ground-moving prey, while flight-hunting is rapid and depends on speed to dive. Again, flight-hunting explores hunt areas that are beyond the scanning range of perches that would otherwise be unexploited (Agbehadji *et al*. 2016).

Assumptions:

- Kestrels hunt individually.
- Kestrels are imitative rather than cooperative. Perhaps kestrels can imitate birds from far off.
- The still position gives a near-perfect circle, thus frequent changes in circle direction depend on the position of a prey in shifting the center of circling direction.
- Frequent bobbing of the head gives a degree of magnified or binocular vision that helps in judging distance to a prey and moving with speed to strike.
- Perch is not likely to give a perfect circle. Thus, ultraviolet-sensitive capabilities are used to visually determine attractiveness of trails closer to the perch area and then glide to capture the prey. Attractiveness is proportional to light reflection, so the higher a distance, the less bright a trail. Additionally, new trails are more attractive than old trails.
- An increase in flight-hunting leads to a corresponding decline in perch-hunting.

3.2.1.2 Preliminary formulation of basic rules

- **Random encircling**

  The encircling mechanism defines a circle-shaped neighborhood solution around a position with different random radii (Mirjalili, Mirjalili and Lewis 2014). The assumption is that as the prey moves at random to its current position, the kestrel randomly changes the center of circling direction randomly in order to recognize the current position of prey to be encircled. This movement of prey determines the best possible position assumed by the kestrel. The mathematical model to depict encircling behavior $\vec{D}$ (Muro *et al*. 2011; Kumar 2015) is expressed as follows:

$$\vec{D} = \left|\vec{C} * \overrightarrow{x_p}(t) - \vec{A} * \vec{x}(t)\right| \qquad \textit{Equation 3.1}$$

$$\vec{x}(t+1) = \overrightarrow{x_p}(t) - \vec{A} * \vec{D} \qquad \textit{Equation 3.2}$$

where

$$\vec{A} = 2 * \vec{z} * \overrightarrow{r2} - \vec{z} \qquad \textit{Equation 3.3}$$

$$\vec{C} = 2 * \overrightarrow{r1} \qquad \textit{Equation 3.4}$$

where $\vec{A}$ is the coefficient vector, $\overrightarrow{x_p}(t)$ is the position vector of the prey, and $\vec{x}(t)$ indicates the position vector of a kestrel. Further, *r1* and *r2* are learning rates that assume a random value between 0 and 1, $\vec{x}(t+1)$represents the current position of a kestrel, $\vec{z}$ represents a parameter to control the active mode, with $\vec{z}_{hi}$ as the parameter for flight mode and $\vec{z}_{low}$ as the parameter for perched mode, which linearly decreases from 2 (high active mode value) to 0 (low active mode value) respectively during the iteration process. This is expressed in Equation 3.5 as

$$\vec{z} = \vec{z}_{hi} - (\vec{z}_{hi} - \vec{z}_{low})\frac{itr}{Max\_itr} \qquad \textit{Equation 3.5}$$

where *itr* is the current iteration and *Max_itr* is the total number of iterations that are performed during the search.

The encircling prey formulation, as adopted from grey wolves’ behavior (Muro *et al*. 2011), which is a variant of wolf behavior, shows group hunting behaviors such as: tracking, chasing and approaching prey, as well as pursuing, encircling, attacking and harassing prey until it stops moving. Although grey wolves’ encircling behaviour is similar to kestrels, kestrels perform encircling of prey individually rather than in groups. This individual hunting behavior suggests that kestrels have the natural ability to track the position of prey and to shift their center direction randomly to adjust their location and identify their prey. Hence, random encircling was adopted to depict the hunting characteristics of kestrels for a successful hunt.

- **Change in position**

  The change in position of a kestrel helps it move toward its prey. The change in position of a kestrel depends on attractiveness and frequency of bobbing, which are explained below. A kestrel's position is updated using Equation 3.6:

$$x_{t+1}^{k} = x_t^k + \beta_o e^{-\gamma r^2}\left(x_j - x_i\right) * f_t^k \qquad \textit{Equation 3.6}$$

where x_{t+1}^{k} is the current best position of the kestrel, which represents candidate solution, x_i is the previous position of the kestrel, $\beta_o e^{-\gamma r^2}$represents the attractiveness of prey, x_j represents a kestrel with a better position, and f_i^k is the frequency of bobbing.

The best candidate represents a candidate that has better knowledge about the potential position of a prey. The best candidate with better position is saved to oblige the other search agents (that is, kestrels) to change their positions according to the position of the best search agent. As kestrels hunt individually, search agents tend to imitate the best position of the other search agents and then update their velocity accordingly.

- **Velocity**

  The velocity of a kestrel moving from its position is expressed in Equation 3.7 as

$$v_{t+1}^{k} = v_t^k + x_t^k \qquad \textit{Equation 3.7}$$

where $\mathrm{v}_{t+1}^{\mathrm{k}}$ represents the current velocity of the kestrel, $\mathrm{v}_{\mathrm{t}}^{\mathrm{k}}$ is the initial velocity, and $x_{\mathrm{t}}^{\mathrm{k}}$ is the position of the kestrel. The change in velocity is controlled by the inertia weight ω (which is also referred to as the convergent parameter) (Ahmed and Glasgow 2012). This inertia weight has a linearly decreasing value, as explained in the random encircling formulation in Equation 3.9. Thus, velocity is expressed in Equation 3.8 as

$$v_{t+1}^{k} = \omega v_{t}^{k} + x_{t}^{k} \qquad \textit{Equation 3.8}$$

where ω is expressed in Equation 3.9 as

$$\omega \quad = \omega_{hi} - (\omega_{hi} - \omega_{low})\frac{t}{T_{max}} \qquad \textit{Equation 3.9}$$

where *t* refers to the iteration counter, ω_{hi} and ω_{low} are the parameters on flight mode (higher bound) and perched mode (lower bound) respectively, and T_{max} is the allowable number of iterations to terminate the search.

- **Frequency of bobbing**

  Frequency of bobbing depends on the kestrel's position on a fixed structure. This is expressed in Equation 3.10 as

$$f_{t+1}^{k} \; = \; f_{min} + \; (f_{max} - f_{min})\alpha \qquad \textit{Equation 3.10}$$

where, $\alpha \; \in \; [0,1]$ is a random parameter between 0 and 1 to control the frequency. The maximum frequency f_{max} is set to 1, and the minimum frequency f_{min} is set to 0. The f_{t+1}^{k} is the current frequency that indicates the frequency of bobbing at each step on a high fixed structure.

When a kestrel is located at a current position, each bob of the head calculates all sight distances to pick the best possible sight distance. This bobbing of the head gives a degree of magnified or binocular vision for judging distance to a prey in hyper-space from different positions before it moves with a velocity to strike. An awareness of changing airstreams and signs of nearby enemies are attributes that dominate in hyperspace. Each kestrel has a visual coverage area with a circling radius defined by *V* for *x*, that is, radius *V* consisting of *x* sets of points representing potential solutions. Sight distance $s(x_i, x_c)$ is expressed using Minkowshi distance in Equation 3.11 as

$$s(x_i, x_c) = (\sum_{k=1}^{n} |x_{i,k} - x_{c,k}|^{\lambda})^{\frac{1}{\lambda}} \qquad \textit{Equation 3.11}$$

Thus,

$$V \leq s(x_i, x_c) \qquad \textit{Equation 3.12}$$

where x_i is the current sight measurement, x_c is all potential neighboring sight measurements near x_i, n is the total number of neighboring sights, and λ is the order (1 or 2) that is assumed in this study to represent the maximum of two "degrees of freedom" to change its position.

- **Attractiveness**

Ultraviolet sensitivity is used to visually locate trails of ground-moving prey. Thus, a kestrel's attraction (β_o) to prey using the ultraviolet sensitivity is proportional to light reflection (or light intensity γ). Thus, the greater the distance r (which equals vision measurement), the less bright the trail. Kestrels are most attracted to new trails. The intensity of the trail γ varies with distance r, since light intensity decreases with distance from source, and light is absorbed in the medium on which the kestrel is found. For a given medium with a fixed light absorption coefficient γ, the light intensity I varies with the distance r. This is expressed in Equation 3.13 as

$$I = I_o e^{-\gamma r} \qquad \textit{Equation 3.13}$$

where I_0 is the original light intensity, γ is the absorption coefficient, and r is distance. The absorption is therefore approximated using the Gaussian equation, hence, attractiveness β is expressed in Equation 3.14 by

$$\beta(r) = \beta_o e^{-\gamma r^2} \qquad \textit{Equation 3.14}$$

where β_o , which equals l_o, is the attractiveness, and γ represents the variation of light intensity between [0, 1].

- **Trail decay or trail evaporation**

A trail is described as formation and maintenance of a line (Dorigo and Cambardella 1997). Usually, in the ant meta-heuristic algorithms, ants use trails to trace the path to a food source and to prevent themselves from getting stuck relying on a single food source. Thus, using these trails, ants can search many food sources in their habitat (Agbehadji 2011). As ants continue to search, trails are drawn and biological evaporative substances are deposited in trails to enable ants to communicate with each other about the location of food sources. Therefore, other ants continuously follow this path and also lay more biological substances for a trail to remain fresh. Similar to ants, kestrels use trails in search of food sources. However, these trails are rather deposited by prey, which provide an indication to kestrels on the availability of food sources. The assumption is that the biological substances deposited by this prey are similar to substances deposited on ants' trail. Additionally, when the source of food depletes, kestrels no longer follow this path. Consequently, the trail substance begins to diminish with time at an exponential rate, causing trails to become old (Agbehadji *et al*. 2018). This diminishment denotes the unstable nature of the trail substances, which can be theoretically stated as: If there are *N* unstable elements with an exponential decay rate γ, then an equation can be formulated to describe how *N* substances decrease in time *t* (Spencer 2002). This equation is expressed in Equation 3.15 as

$$\frac{dN}{dt} = -\gamma N \qquad \textit{Equation 3.15}$$

In other words, since the substances are unstable, they introduce randomness in the decay process. Thus, decay rate γ with time *t* is re-expressed in Equation 3.16 as

$$\gamma_t = \gamma_o e^{-\lambda t} \qquad \textit{Equation 3.16}$$

where γ_o is a random initial value of substance that is decreased at each iteration, and where *t* is the number of iterations or time steps. Further, *t* ∈ [0, *Max_itr*], where *Max_itr* is the maximum number of iterations. The decay rate γ_t at time *t* to indicate a new trail or old trail is expressed in Equation 3.17 as

$$if\ \gamma_t \rightarrow \begin{cases} \gamma_t > 1, & trail\ is\ new \\ 0, & otherwise \end{cases} \qquad \textit{Equation 3.17}$$

Again, the decay constant λ is expressed in Equation 3.18 by:

$$\lambda = \frac{\phi_{max} - \phi_{min}}{t_{\frac{1}{2}}}$$ *Equation 3.18*

where λ is the decay constant, ϕ_{max} is the maximum number of substances in the trail, ϕ_{min} is the minimum number of substances in the trail, and $t_{\frac{1}{2}}$ is the half-life period of a trail, which shows that a trail is old and unattractive if less than ½ is not worth exploring (Agbehadji *et al*. 2018).

- **Imitative behavior**

Kestrels are territorial and hunt individually rather than collectively. As a consequence, a model by Cui and Potok (2006), which depicts the collective behavior of birds for feature similarity selection, could not be applied. However, individual hunting can be used for feature selection based on imitative behavior. Since kestrels are imitative, it implies that a well-adapted kestrel would perform actions appropriate to its environment, while other kestrels that are not well adapted imitate and remember the successful actions of the well-adapted kestrel. The imitation behavior reduces learning and improves the skills of less adapted kestrels. A kestrel is most likely to take a random step that imitates a successful action for a global optimum rather than not imitating and, as a result, can become stuck in a local optimum that it alone discovered.

Imitation learning is an approach to skill acquisition (Englert *et al*. 2013) where a function is expressed to transfer skills to lesser-adapted kestrels. This suggests that lesser-adapted kestrels feel more drawn to imitate while observing from a close distance (Penaloza *et al*. 2012). Therefore, the short distance results in higher imitation. In the present approach, the position at which a kestrel can copy an action in a large search domain was imitated. The imitation behavior is mathematically expressed and applied to select similar features into a subset. A similarity value $Sim_{value\ (O,T)}$ that helps with the selection of similar features is expressed in Equation 3.19 by

$$Sim_{value\ (O,T)} = e^{\left(-\frac{\sum |O_i - E_i|^2}{n}\right)} \qquad \textit{Equation 3.19}$$

where n is the total number of features, and $|(O_i - E_i)|$ represents the deviation between two features where O is the observed, and E_i is the estimated velocity of the kestrel. Since the deviation is calculated for each feature dimension, at each time step only the minimum deviation is selected (the rest of the dimension is discarded), thus enabling the kestrel to allow the handling of different problem dimensions of data (Englert *et al*. 2013). Speed of convergence of imitation of similar features is regulated by a control parameter where the higher the value of the control parameter, the lower the convergence speed of imitation, and hence a lower learning rate. Moreover, cases where features that were imitated are not similar (that is, dissimilarity) are expressed in Equation 3.20 as

$$dis\_sim_{value\ (O,T)} = 1 - Sim_{value\ (O,T)} \qquad \textit{Equation 3.20}$$

The fitness function, which is similar to the fitness function formulation used by (Mafarja and Mirjalili 2018) and which evaluates each solution, is expressed in terms of classification error of the RNN and the similar value obtained from each solution. The fitness function is formulated using Equation 3.21:

$$fitness = \rho * Sim_{value\ (O,T)} + dis\_sim_{value\ (O,T)} * \rho \qquad \textit{Equation 3.21}$$

where $\rho \in (0,1)$ is a parameter that controls the chances of imitating features that are dissimilar, C_{error} is the classification error of an RNN classifier, and $Sim_{value\ (O,T)}$ refers to the feature similarity value obtained in feature imitation.

- **Slope of glide to capture**

The slope of glide to capture differentiates the time distance that each kestrel moves in time width ω_t. When the time σ_t at which a kestrel moves is specified and a minimum value $f(\sigma_t)$ in an interval [0, σ_t] is determined, then the slope of glide s to the prey within a time width ω_t is computed using Equation 3.22:

$$s = \sqrt[(\sigma_t - \omega_t)]{\frac{1}{f(\sigma_t)} - 1} \qquad \textit{Equation 3.22}$$

where $f(\sigma_t)$ represents a function of the final velocity of the kestrel obtained within a time window width ω_t and specified time σ_t to move. Thus, a larger σ_t implies a larger slope *s*. The basis of the slope to glide is to help determine the time differences between the points at which a kestrel starts a glide to another intermediate point. The significance of slope to glide is that it helps in discovering association rules within each time interval.

3.2.2 Description of dung beetle behavior

The behavior of dung beetles for data visualization will be described in detail in section 3.4 of this thesis. The characteristics and simplified rule formulation on the characteristics of dung beetles will also be discussed.

3.3 Kestrel behavior that relates to big data characteristics

Kestrel behaviors that relate to big data environment characteristics are *velocity* (that is, the speed to dive on prey with a time dimension), *variety* (different kinds of prey that require different attack methods), *veracity* (attractiveness, that is, accuracy of results from the processing system), *value* (what the user will gain (usefulness) from the analysis) and *volume* (larger search space referring to size of data). Prey represents a frequent item while movement (e.g. change in numeric data points, such as price data) among prey indicates frequently changed items among frequent items. Hence, frequent items can be classified by a kestrel's view based on size of data, speed, accuracy and usefulness of pattern in respect of time. This behavior of kestrels is mathematically modelled to handle size, speed, accuracy and usefulness in order to handle the challenge of data cleansing and data mining.

In respect of visualization of patterns, the present study looked at the role of dung beetles in data visualization without relating their behavior to aspects of big data characteristics. The role is to ensure large volumes of data can be viewed with limited computational cost and in short time (i.e. velocity) by data analytics platforms. In order to fulfil this role, the navigation and orientation behavior of dung beetles is applied to model an algorithm for data visualization (see section 3.4).

The following sub-sections contain a discussion of how to address the challenges of data cleansing and data mining in big data frameworks using the behavior of kestrels.

3.3.1 Phase 1: Data cleansing

3.2.1.3 Stage 1: Extrapolating missing data using the Kestrel-Based Search Algorithm

The search is a random search strategy that is derived from the mathematical formulations on the characteristics of the kestrel. The mathematical modeling of the kestrel behavior (in section 3.2.1) was used to extrapolate missing data (that is, missing values at random).

3.2.1.4 Stage 2: Identifying and eliminating duplicate texts

During the second stage, the Smith-Waterman algorithm, as discussed in the literature review, was applied to identify, match and eliminate duplicate text from a single dataset. The importance of duplicate detection is that it avoids duplicate tuples from a dataset. Consequently, this algorithm was used to remove duplicate text that had been identified. An approach to removing duplicates is the use of the pairwise comparison method. Pairwise comparison of a sequence of characters of a text was applied to find the optimal local alignment of characters as follows. The local alignment is performed in the following steps:

Let *A* and *B* represent the sequence of two texts by $A=\{a_1,a_2,\ldots a_n\}$ and $B=\{b_1,b_2,\ldots,b_m\}$, where *n* and *m* are the length of the two texts respectively.

Step 1: Find the substitution matrix and compute the gap penalty in terms of the cost of starting *s* and ending a gap as:

$$s(a_i, b_j) = \sum_{i,j=1}^{k} s(a_i, b_j) \qquad \textit{Equation 3.23}$$

where s: $(\Sigma \cup \{-\})^2 \rightarrow \mathbb{R}$ represents the similarity function, and s (a_i, b_j) >0, s(a_i, -) < 0, s(-, b_j) < 0, where the symbol "-" represents a gap that appears either after a character or before a character. If there is a match, +1 is assigned, and if a mismatch, -1 is assigned. Thus, the substitution matrix is described in Equation 3.24 as:

$$s(a_i, b_j) = \begin{cases} +1, a_i = b_j \\ -1, a_i \neq b_j \end{cases} \qquad \textit{Equation 3.24}$$

Step 2: Initialize scoring matrix *M[i][j]* such that *M[i][0]=0* and *M[0][j]=0* and the size of the score matrix is *(1+length(A))*(1+length(B))*.

Step 3: Scoring matrix. Score each element of the *M[i][j]* from left to right, top to bottom matrix using the matrix equation in Equation 3.25:

$$M[i][j] = max \begin{cases} M[i-1][j-1] + \ s(a_i, b_j) \\ M[i-1][j] - c; \ \ if \ (a_i, -) \\ M[i][j-1] - c; \ \ if \ (-, b_j) \end{cases} \qquad \textit{Equation 3.25}$$

where an entry into the matrix produces the best possible score for match or mismatch *s*(a_i, b_j) on a prefix of two strings, and *c* is the cost of a single gap that is expressed in a linear gap penalty as C_k=*k*C_1, where *k* is the gap length.

Each element in the scoring matrix in Equation 3.25 can be represented in a matrix table as follows:

Table 3.2: Element representation in tabular form

	b_{j-1}	b_j
a_{i-1}	$M[i-1][j-1]$ $+ s(a_i, b_j)$ 1	$M[i-1][j]$-c_1 2

a_i	$M[i][j-1]$-c_1 3 →	🟥

The best local alignment score is computed by:

$$Score = Max_{i,j=1}^{n}(M[i][j]) \qquad \textit{Equation 3.26}$$

Step 4: Traceback. Start with elements with the highest score and end at a matrix cell with score equal to 0. The traceback starts at the bottom right and ends at the top left for best alignment. Traceback helps find best local alignment.

The challenge with the Smith-Waterman algorithm is the large amount of time required to perform a similarity check on each character in a text (CLC bio 2007). In order to demonstrate how the Smith-Waterman algorithm compares two sequences using local alignment and identifies best alignments of high importance to both the reliability and relevance of the data obtained as duplicate, the following example was adapted from CLC bio (2007) to explain how the Smith-Waterman is used:

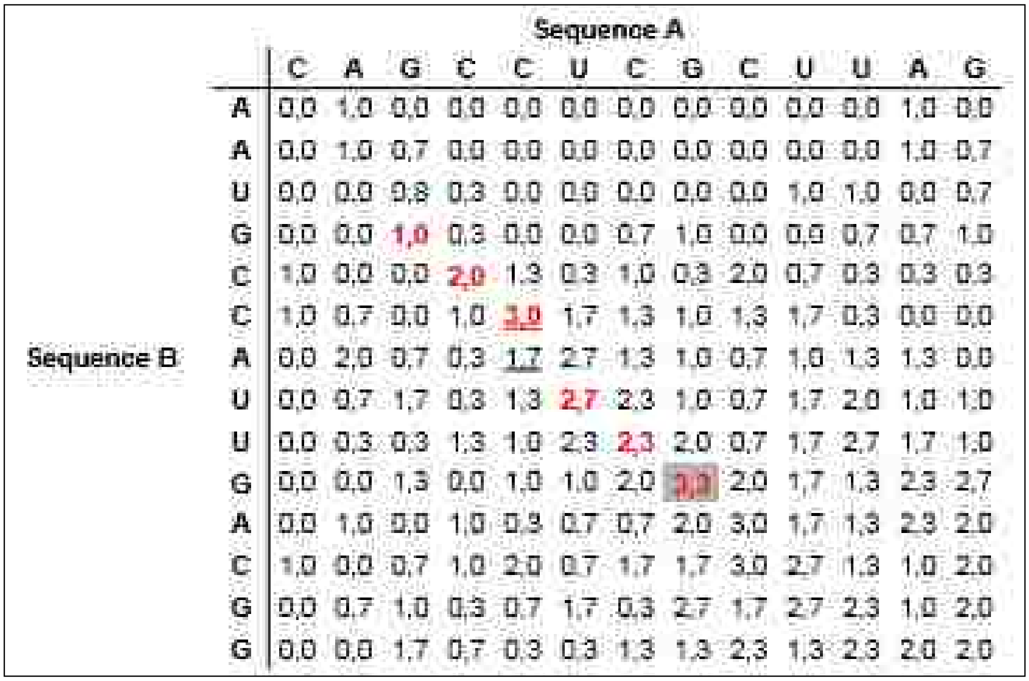
Sequence in text A: CAGCCUCGCUUAG

Sequence in text B: AAUGCCAUUGACGG

parameters for the scoring matrix being:

- match = 1
- mismatch = $-\frac{1}{3}$
- gap = ch =$\frac{1}{3} * k$), where k represents the gap extension number.

Using Equation 3.26, the similarity matrix is filled as shown in Figure 3.1:



Sequence A

Sequence B

	C	A	G	C	C	U	C	G	C	U	U	A	G
A	0,0	1,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	1,0	0,0
A	0,0	1,0	0,7	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	1,0	0,7
U	0,0	0,0	0,3	0,3	0,0	0,0	0,0	0,0	0,0	1,0	1,0	0,0	0,7
G	0,0	0,0	1,0	0,3	0,0	0,0	0,7	1,0	0,0	0,0	0,7	0,7	1,0
C	1,0	0,0	0,0	2,0	1,3	0,3	1,0	0,3	2,0	0,7	0,3	0,3	0,3
C	1,0	0,7	0,0	1,0	3,0	1,7	1,3	1,0	1,3	1,7	0,3	0,0	0,0
A	0,0	2,0	0,7	0,3	1,7	2,7	1,3	1,0	0,7	1,0	1,3	1,3	0,0
U	0,0	0,7	1,7	0,3	1,3	2,7	2,3	1,0	0,7	1,7	2,0	1,0	1,0
U	0,0	0,3	0,3	1,3	1,0	2,3	2,3	2,0	0,7	1,7	2,7	1,7	1,0
G	0,0	0,0	1,3	0,0	1,0	1,0	2,0	3,3	2,0	1,7	1,3	2,3	2,7
A	0,0	1,0	0,0	1,0	0,3	0,7	0,7	2,0	3,0	1,7	1,3	2,3	2,0
C	1,0	0,0	0,7	1,0	2,0	0,7	1,7	1,7	3,0	2,7	1,3	1,0	2,0
G	0,0	0,7	1,0	0,3	0,7	1,7	0,3	2,7	1,7	2,7	2,3	1,0	2,0
G	0,0	0,0	1,7	0,7	0,3	0,3	1,3	1,3	2,3	1,3	2,3	2,0	2,0

Figure 3.1: Scoring matrix

Figure 3.1 represents the scoring matrix, where cell value represents the score of the optimal alignment ending at the cell coordinates, and the highest scoring position in the matrix reports the ending point of the highest scoring and thereby the optimal alignment between the two sequences compared (CLC bio 2007).

To construct the optimal alignment, the starting point is the cell with the highest scoring value representing the last residue in this alignment. The complete alignment is identified by tracing back through the array from the highest scoring matrix cell until a cell scoring zero is reached.

In Figure 3.1, the highest scoring cell in the diagonal is **3.3** and is traced back six steps. The search for local alignments allowing any position to be a starting point and any position to be an ending point means that the optimal alignment can be of any possible length and is thereby identified as the optimal local alignment. Finally, the alignment represented by the path shown in red in the similarity matrix is expressed as

Sequence in text B: G C C A U U G

Sequence in text A: G C C - U C G

Time cost may be important but, in some contexts, accuracy is more significant in a situation where the risk of missing very sensitive information could result in loss of life (CLC bio 2007). It is possible to perform global alignment by considering each string that has been locally aligned as a token.

- **Tokenization**

Generally, the concept of tokenization is used for lexical analysis, where different elements represent individual words. These different elements represent a segment on a sequence of strings, often called tokens (Thakker, Osman and Lakin 2009), which indicate words. Tian *et al*. (2002) suggest that the way to identify erroneous words is by comparing words, and words that are of similar value are considered identical. Originally, tokenization works best when words are transposed (Elmagarmid *et al*. 2006). Kannan and Gurusamy (2014) indicate that text data could be represented as blocks of letters, that is, words, that are stored in machine-readable formats. In the process of tokenization, tokenizers (identifiers) are used to identify meaningful keywords (Vijayarani and Janani 2016) within documents so as to find the consistency of documents. Although document analysis is not the focus of this thesis, it is worth mentioning the concept of tokenization, which forms the basis of the present approach to duplicate detection. The advantage of the token-based method is that while it compensates for the character-based and *n*-gram similarity measures, it fails to capture the similarity of the strings when the order of words is changed (e.g. Forbit Café versus Café Forbit) (Gali *et al*. 2016). An example of an algorithm based on token-based methods is the *n*-gram (Cohen *et al*. 2003). The *n*-gram is an algorithm that enables the detection of error in words to suggest misspelt words, where *n* represents consecutive letters in a word (Tian *et al*. 2002). The *n* consecutive letters may be taken in twos (di-grams), threes (tri-grams) or fours (quad-grams). For instance, using the word “cardiogram”, di-grams can be formed as (ca-ar-rd-di-io-og-gr-ra-am); tri-grams are formed as (car-ard-rdi-dio-iog-ogr-gra-ram); and quad-grams is formed as (card-ardi-rdio-diog-iogr-ogra-gram) respectively. The example indicates the structural information of the word (Tian *et al*. 2002). Thus, *n*-grams of a word represent some structural information of the word.

Each token is compared with other tokens to find the transitive, reflexive or symmetry property of equality. For a given binary relation *R* on *A* (records), the transitive, reflexive and symmetry properties that identify records as duplicate are expressed by the following definitions:

Definition 1: Transitive property of equality of record

The significance of the transitivity property is that it reduces the number of total comparisons that need to be made on words (Dănăilă *et al*. 2012). Transitivity is expressed as a binary relation *R*. For example, the relation *R* on {1,2,3} is given by *R* = {(1,1), (1,2), (2,1), (2,2), (2,3), (1,3)} under the assumption that the "is a duplicate of" (Hernandez and Stolfo 1995) relation *R* is transitive by constructing a similarity graph where duplicates amount to finding the connected components of the resulting graph, thereby avoiding unnecessary comparisons between already connected nodes and reducing computation cost (Dănăilă *et al*. 2012). This means that, for each word *x, y, z* in *A*, if *x* word relates to *y* word, and *y* word relates to *z* word, that implies *x* word relates to *z* word in an ordered pair. Thus, *x, y, z* are the same word within a record. In this case *x, y, z* are considered nodes in a connected graph. The transitive closure of *R* is defined as the binary relation R^t on *A* satisfying the following three properties:

1. R^t is transitive.
2. *R* is a subset of R^t.
3. If *S* is any other transitive relation that contains *R*, then *S* contains R^t.

In other words, the transitive closure of *R* is the smallest transitive relation containing *R*.

Definition 2: Reflexive property of equality of record

A record is considered reflexive if every word x in the record A is related to itself (Borschev and Partee 2001). The reflexive property can be expressed as a binary relation R if for each $x \ \in A$, $(x, x) \ \in R$, $(x, x) \in R$, where the reflexive relation is the use of the "is-equal-to" relation. This means that, for every word x in A, x word relates to another x word. Similarly, a word in a record relates to the same word in another record. For example, the relation R on {x1, x2, x3} is given by R = {(x1, x1), (x2, x2), (x2, x3), (x3, x3)} and is reflexive. This means x1 is equal to another x1, and both x1 and x1 are therefore reflexive.

Definition 3: Symmetry property of equality of record

A binary R is symmetric if for each $x, y \ \in \ A$, $(x, y) \in R$ implies $(y, x) \ \in R$. Meaning for all $x, y \ \in \ A$, x word relates to y word implies that y relates to x. For instance, the relation R on {1,2,3} is given by R = {(1,1), (1,2), (2,1), (1,3), (3,1)} and is symmetric. These definitions expressed adhere to the property of equality in the sense that a word in each record is transitive for each, thus if x=y, y=z, then z=x. The reflexive property is expressed as x=x while, the symmetry property is expressed as if x=y then y=x. The transitive, reflexive and symmetry properties' adherence to the equality (or equivalence) property forms the basis for pairwise alignment between two health records. This equivalence is measured in terms of the degree between 0 and 1 (Monge 2000) where 1 is assigned if there is an equivalence, while 0 is assigned if there is no equivalence.

- **Pairwise alignment**

Pairwise alignment finds the best-matching sequence of strings such as words. Given two strings Y (y_1, y_2...y_n) and X (x_1x_2...x_m), a pairwise alignment between Y and X is defined as an ordered set of pairings of each (y_i, x_j) and of gaps (y_i, −) and (−, x_j), where i $\leq$ n., with the constraint that an alignment is reduced to two original strings when all gaps in the alignment are deleted. The symbol "–" represents a gap that appears either after a character/letter or before a character/letter. The gap penalty function penalizes alignment of words and then either accepts a gap or inserts alignment to achieve a good/approximate alignment of words. The gap penalty function is expressed as *w(k)*, where *w* is the cost of a gap and *k* is the length. The cost of a gap penalty, when carefully selected, avoids a high gap penalty that may lead to unmatched positions of letters. This can result in unequal intervals on the position of characters in a text, thus leading to inaccurate matches. Hence, the gap penalty gives an acceptable interval for good alignment. The matched letters are selected and optimized over the sum of both matched and unmatched words (Vingron and Waterman 1994).

Agbehadji, Millham, Fong and Yang (2018a) method on duplicate word/text detection applies two concepts, namely word tokenization and a character-based method to detect duplicate words. In this context, tokens are referred to as segments of words that are grouped together as useful semantic units for processing by the Jaro-Winker and/or Smith-Waterman algorithm. During the tokenization process, it is possible to have large volumes of datasets that can increase the computational time. To avoid this situation, the transitivity closure on words using the Union/Find technique, which has the property to reduce the number of total comparisons of words, is applied. Afterwards, the property of equality (that is, the symmetry property) is applied to perform pairwise word comparison. Finally, a character-based method, such as the Jaro-Winkler and/or the Smith-Waterman algorithm, is applied to perform pairwise comparison of characters in each word.

3.2.1.5 Stage 3: Data reduction

This is the process of reducing a large volume of data into a smaller set for an efficient data mining process (Rehmana *et al*. 2016). Data reduction is important because it helps build a model to classify datasets by selecting best features, avoids unbiased classification and avoids imbalances in the dataset so as to guarantee the same results. Clustering is one of the methods that helps put unlabeled data into similar subsets, and it is regarded as unsupervised classification (that is, no predefined classes) (Ullman *et al*. 2014). Although data clustering is important for data reduction, it is outside the scope of this thesis. This thesis focuses on classification to help reduce data into relevant subsets.

The application of an efficient search strategy to select optimal feature subsets in the classification of data into subsets is significant in reducing the computational cost when the volume of data is larger. In the present approach, to achieve optimal feature subset selection, the random encircling formulation and imitation behavior of kestrels was combined with an RNN (which was explained earlier) with a long short-term memory network to build a classifier for feature selection in big datasets. The KSA was applied to learning an optimized parameter from the original dataset, and the learned parameter was used as input into the RNN with LSTM network so as to select relevant features. Additionally, it helped evaluate the performance (in terms of accuracy) of the KSA as a feature selection algorithm.

3.3.2 Phase 2: Data mining

Data mining/big data mining is an approach used to find hidden and complex relationships present in data (Sumathi and Sivanandam 2006) with the objective to extract comprehensible, useful and non-trivial knowledge from large datasets (Luna *et al*. 2011). Although there are many hidden relationships to be discovered in data, this thesis focuses on association rule relationships, which are explored using association rule mining.

- **Association rule mining**

  Association rules are rules that help disclose frequently changed patterns from a dataset. These changed patterns are measured in terms of support and confidence values set by a user as expressed in Equation 3.27 and 3.28 respectively. Rules with minimum support and minimum confidence value below a user-specified threshold are considered uninteresting (Han and Kamber 2006), while rules with minimum support and minimum confidence value above a user-specified threshold are considered interesting.

Generally, an association rule is measured in terms of support and confidence. The support of a rule is defined as the proportion of appearance in the dataset (Gupta and Sikka 2013; Agbehadji *et al*. 2016). In other words, it is the frequency of the rule in the dataset. A high value shows that the rule involves a great part of the dataset. Support is expressed in Equation 3.27 as

$$support\ (K \rightarrow P) = \frac{\sigma(K \cup P)}{\sigma(N)} \qquad \textit{Equation 3.27}$$

where *N* is the total number of items in a dataset and $K \cup P$ is the number of attributes containing both *K* and *P*.

Generally, confidence is used to measure the number of times an item in *P* appears in transactions that contain *K*. Confidence of a rule is when rules are applied to find the ratio of the number of occurrences in *K* and *P* over the number of occurrences in *K*. In other words, confidence of a rule is expressed as a conditional probability of the consequent given the antecedent (Gupta and Sikka 2013). This is expressed in Equation 3.28 as

$$confidence\ (K \rightarrow P) = \frac{\sigma(K \cup P)}{\sigma(K)} \qquad \textit{Equation 3.28}$$

where σ *(K)* is the number of occurrences that contain *K*. Typically, a higher confidence value suggests a strong association between *K* and *P*.

The use of support and confidence measures is not, however, sufficient in selecting actionable sequences (Tseng, Liang and Chu 2006; Yin *et al*. 2013). This is because the actionable sequences form patterns that may be interesting to a user, but the challenge is when the user has to search exponentially many potentially interesting patterns that meet a defined criterion (Vreeken and Tatti 2014). Additionally, evaluating each pattern to disclose interestingness is quite tedious and infeasible, and mostly considered non-trivial (Vreeken and Tatti 2014).

The present approach to measuring interestingness of rules on frequently changed patterns uses closeness preference interestingness measure (Railean *et al*. 2013) that finds rules in cases where the antecedent and consequent are not just frequent or random (Railean *et al*. 2013) but has a time dimension at which an actionable sequence is recorded. The advantage of the time dimension is that it could help determine the time to take an action when numeric items are frequently changed. The Closeness (user) Preference CP interestingness measure (Railean *et al*. 2013) was used to select strong rules with respect to frequencies of antecedent *A* and consequent *B*. Rules are said to be close based on three parameters, namely close time interval, slope of preferences and threshold value (Railean *et al*. 2013). The time closeness refers to the user-defined time difference in which items occur. Slope of preferences refers to the size between two itemsets or the size between two slides (window). Finally, threshold value is the user-specified value within which rules are extracted. These three parameters are used to select strong rules for patterns to be extracted.

Railean *et al*. (2013) notes that time closeness plays a significant role in finding how close antecedents are to consequents of items. The time closeness shows the time difference between items over an entire sequence of changing items. The smaller the time differences, the closer the occurrence of frequently changed items. The support of a rule on an item is expressed in Equation 3.29 by

$$sup_{pattern} = \frac{nr_{p\ o}}{|\ DB\ |} \qquad \textit{Equation 3.29}$$

where |*DB*| is the total size of the dataset, and $nr_{p\ o}$ is the number of occurrences in a pattern of frequently changed numeric items. The Modified Closeness Preference with Support (MCPs) (Railean *et al*. 2013) function that fulfils the anti-monotone property (that is, the support of an itemset never exceeds the support of its subsets) of an item is expressed as the product of a function as

$$MCP_{s\ func} = \prod_{s=1}^{nr_{\ Freq\ with\ p}} [\frac{1}{nr_{\ p\ in\ Freq}} \sum_{t=1}^{nr_{p\ in\ Freq}} f_{pattern_t}] \qquad \textit{Equation 3.30}$$

where $nr_{p\ in\ Freq}$ is the number of how many times the pattern *p* is found in a given sequence to be frequently changed. Further, $f_{pattern}$ is expressed in Equation 3.31 by:

$$f_{pattern} = \frac{1}{nr_{2-length\ subpattern}} * \sum_{k=1}^{nr_{2-length\ subpattern}} \left[\frac{1}{n_{t_{P_{i,k}}}} \sum_{m=1}^{n_{t_{P_{i,k}}}} \frac{1}{n_{P_{i+1|P_{i,k,m}}}} \sum_{j=1}^{n_{P_{i+1|P_{i,k,m}}}} \frac{1}{1+s^{t_{P_{i+1,j}}-\omega t \cdot}} \right] \qquad \textit{Equation 3.31}$$

where $n_{t_{P_i}}$ is the number of P_i between the previous and next itemsets; $t_{P_{i+1,j}}$ is the time distance of each P_{i+1} from the beginning of the window (starting from the considered P_i); *nr* is the number of items; and *s* represents the slope of glide to capture, which was expressed in Equation 3.22 (in section 3.2). In items that are frequently changed with time, taking into consideration time differences between two consecutive frequently changed items, each pattern of length *n* is made of *n*-1 sub-patterns of length 2. In order to avoid patterns that are not frequently changed (i.e. that only occur a few times in the entire dataset), the support measure was included within the mathematical formulation such that the MCPs function with the support measure is expressed in Equation 3.32 as

$$MCP_{s\ func} = \frac{nr_{p\ o}}{|\ DB\ |} * [0.9^{nr} + (0.9^{nr-1} - 0.9^{nr}) * \prod_{s=1}^{nr_{p\ o}} [\frac{1}{nr_{p\ in\ seq}} \sum_{t=1}^{nr_{p\ in\ seq}} f_{pattern}]] \qquad \textit{Equation 3.32}$$

Finally, the MCPs function is expressed in Equation 3.33 by:

$$MCP_{s\ function} = \frac{nr_{p\ o}}{|\ DB\ |} * f(nr, \Delta t_{average}) \qquad \textit{Equation 3.33}$$

where $f(nr,\ \Delta t_{average})$ is a weighting function that is defined to take into consideration the size of the pattern *nr* (that is, number of frequently changed numeric items) and the medium time interval $\Delta t_{average}$ in a pattern of frequently changed numeric items; $nr_{\text{pattern occurrences}}$ ($nr_{p\ o}$) represents the number of occurrences of frequently changed numeric items; and |*DB*| is the total number of items in the dataset. The weighting function that shows the anti-monotone property is expressed by Railean *et al*. (2013) as in Equation 3.34:

$$f(nr, \Delta t_{average}) = 0.9^{nr} + (0.9^{nr-1} - 0.9^{nr}) * \ \Delta t_{average} \qquad \textit{Equation 3.34}$$

where $\Delta t_{\text{average}}$ is replaced by Equation 3.31. Where the co-efficient of 0.9 represents 90 percent of rules that were selected. In addition, if $\Delta t_{\text{average}}$ is higher, it takes the upper value of the ceiling, otherwise, it takes the bottom value, but it never passes to the previous coefficient the values of its sub-patterns, which guarantees the Apriori principle (in this instance, if an itemset is frequently changed, then all of its subsets may also be frequently changed).

The Modified Closeness Preferences with confidence (MCPc) are expressed in Equation 3.35 by

$$MCPc = \frac{nr_{p\ o}}{nr_{p\ o} + nr_{n\ p}} \qquad \textit{Equation 3.35}$$

where $nr_{\text{pattern occurrences}}$ ($nr_{p\ o}$) represents pattern occurrence of frequently changed items, and $nr_{\text{not freq. patterns}}$ ($nr_{n\ p}$) represents patterns that are not frequently changed. The Modified Closeness Preferences with support and confidence (MCP_{sc}) for a pattern are defined as the interestingness measure of frequently changed numeric items, as expressed in Equation 3.36:

$$MCP_{sc\ pattern} = \phi * \frac{nr_{p\ o}}{nr_{p\ o} + nr_{n\ p}} * \frac{nr_{p\ o}}{|\ DB\ |} * f(nr, \Delta t_{average}) \qquad \textit{Equation 3.36}$$

where ϕ represents a control parameter between 0 and 1; $nr_{\text{pattern occurrences}}$ ($nr_{p\ o}$) represents frequently changed items; $nr_{\text{not freq. patterns}}$ ($nr_{n\ p}$) represents patterns that are not frequently changed; $|DB|$ is the total size of the dataset; and $f(nr, \Delta t_{average})$ is a function that is defined to take into consideration the size of the pattern *nr* and the medium time-interval $\Delta t_{average}$ in a pattern.

3.4 Data visualization

In section 3.3, the kestrel's behavior was described as it relates to big data, which forms Phase 1 and Phase 2 of the proposed methodological framework. In this section, the behavior of the dung beetle is described in detail as it relates to data visualization. This forms Phase 3.

3.4.1 Description of dung beetle behavior

Data visualization presents data in pictorial or graphical format (e.g. two dimensionally) and enables the display of interesting patterns for decision-making activities (Bikakis 2018). The present approach to data visualization uses the navigation and orientation behavior of dung beetles to present data in a graphical format. The dung beetle is an animal with a tiny brain (similar to a grain of rice) that feeds on the dung of herbivorous animals. The dung beetle is known to use minimal computation for navigation and orientation, which is the reason for its selection. Furthermore, during navigation, it uses an external reference point (referred to as a celestial polarization pattern) (Wits University 2013), which serves as a source of illumination to avoid being stuck. They also navigate by combining internal cues of direction and distance with external reference from their environment and then orient themselves using the celestial polarized pattern (that is, moon/skies) (Wits University 2013; Dell'Amore 2013). In other words, if a source of light is removed completely, the dung beetle stops moving and stays in a stable position (or unknown state) until the source of light is restored before it climbs on top of its dung ball to perform orientation (referred to as a dance), during which it locates the source of light and then begins to move toward its home. Thus, the beetle returns to its home using an internal sense of direction (derived from sensory sources, including vision).

There are different forms of dung beetles, and these can be grouped into three types, namely rollers, tunnelers and dwellers. Rollers form dung into a ball and roll it to a safe location. In contrast, tunnelers land on a pile of dung and simply dig down to bury the dung, while dwellers stay on top of a dung pile to lay their eggs (Dell'Amore 2013). Although there are different behaviors that categorize each group of dung beetle, the present study focused on the category of ball roller behavior for data visualization purposes.

In addition to the different categories of behavior of dung beetles, their feeding process is also a unique behavior that is worth mentioning. During the feeding process, each beetle (of the type ball rollers) carries dung in the form of a ball and rolls it from the source of food to a remote destination (referred to as home). Dung beetles use the sun, moon and skies as a direction guide in carrying rolled ball (that is, its food) along a straight path from the dung pile (Dacke *et al*. 2011). Given that celestial bodies always remain constant relative to the dung beetle, the beetle keeps moving in a straight line (Wits University 2013) until it reaches the final destination.

Another interesting behavior is that given a burrow (home) and forage (food), the dung beetle is able to move in search of forage by counting its number of steps, and when returning home, the motion cues are used to integrate its path (that is, combine paths) in order to reduce the distance in moving. The path integration technique is significant in reducing the time and distance of moving to its home. Path integration leads to a home-based global vector that repeatedly inform animals (such as dung beetles) about their current position relative to their starting point (Andel and Wehner 2004). When the starting points are determined, path integration provides a "signal" to reinforce the learning of visual landmarks within the environment of animals (Schatz *et al.* 1999). The dung beetle ignores landmarks (Smolka and Dacke 2017) because navigation using landmarks is very complex and may require complex perceptual and learning processes that may not always be available to animals such as dung beetles (Etienne *et al*. 1988). The reason for choosing the dung beetle is because of light-weight computational requirements. In view of this, animals that use the landmark navigation technique require extensive computational time, which may not be suitable within the context of this thesis.

Golani, Benjamini and Eilam (1993) indicate that animals in a new environment center their exploration base on a reference point in order to integrate their path. The path integration (Mittelstaedt and Mittelstaedt 1982) is based on the assumption that movement from one position to another may be achieved by adding successive small changes in position incrementally, and by continuously updating the direction and distance from the initial point (Etienne and Jeffery 2004) using the motion cues. In other words, it allows beetles to calculate a route from their initial position. Adding these successive small movements on a route creates a stack of moves in a hierarchical fashion. The basic steps of the path integration process are the continuous estimation of self-motion cues to compute changes in location (distance) and orientation (head direction) (Etienne and Jeffery 2004). In this regard, every displacement of forage that leads to path integration from a reference point creates an imaginary home, and this subsequently creates a stack of neighboring imaginary homes close to each other. In this context, these real or imaginary homes are circular holes (representing a data grid) where the rolled balls (that is, data values) are placed as pixels.

A. *Characteristics of dung beetles*

The dynamic behavior of dung beetles is characterized as follows:

i. Ball rolling on a straight line

ii. Dance: Combining internal cues of direction and distance with external reference from their environment and then orienting themselves using the celestial polarized pattern

iii. Path integration: Sum of sequential change in position in hierarchical fashion and continuously updating direction and distance from the initial point to return home.

B. *Simplified rule formulation on characteristics of dung beetles*

i. Ball rolling: The distance *d* between two positions (x_i, x_{i+1}) on a plane is calculated using the straight line equation in Equation 3.37:

$$d(x_i, x_{i+1}\,) = \sqrt{\sum_{i=1}^{n}(x_{i+1} - x_i\,)^2} \qquad \textit{Equation 3.37}$$

where x_i represents the initial position, x_{i+1} represents the current position of a dung beetle on a straight line, and *n* is the number of discrete points on the line.

ii. Path integration: Change in position is expressed in Equation 3.38 as

$$x_{t+1}^{k} = x_t^k + \beta_m(\, x_{i+1} - x_i\,)_t^k + \varepsilon \qquad \textit{Equation 3.38}$$

where x_{t+1}^{k} represents the current position of a dung beetle, and β_m represents motion cues. Since path integration is an incremental recursive process, error ε is introduced as a random parameter in the formulation to account for cumulative error. Each frequent return to a home resets the path integrator to a zero state, so that each trip starts with an error-free path integrator (Etienne and Jeffery 2004). Thus, total path is expressed as the sum of all paths, as expressed in Equation 3.39:

$$Path = [\sum_{i=1}^{n} x_{t+1}^{k}] \qquad \textit{Equation 3.39}$$

where current position is x_{t+1}^{k} and *n* represents the number of paths. In order to control the movement *v* between a “real home” and “imaginary home”, to ensure the current position x_{t+1}^{k} converges to the “real home” of a dung beetle during path integration, the following expression was applied, as shown in Equation 3.40:

$$v = v_o + path - (\mu_1 P + \mu_2 A) \qquad \textit{Equation 3.40}$$

where v_o represents the initial movement; μ_1 is a factoring co-efficient of repulsion *P* between each dung beetle; and μ_2 is a factoring co-efficient of attraction *A* between each dung beetle when a trace is detected on its path by another dung beetle. Furthermore, *P* and *A* are expressed by Mamduh *et al*. (2014) using Equation 3.41 and Equation 3.42:

$$P = 1 - d(x_i, x_{i+1})\theta/(d(x_i, x_{i+1})_{max}\, \pi) \qquad \textit{Equation 3.41}$$

$$A = \theta/\pi \qquad \textit{Equation 3.42}$$

where P is the repulsion between each dung beetle, θ is the angle of the dung beetle, $d(x_i, x_{i+1})$ is the distance between two dung beetles, $d(x_i, x_{i+1})_{max}$ is the maximum distance recorded between two dung beetles, and π represents the ratio of circumference to a diameter.

iii. Dance: The internal cue (*I*$_q$) of distance and direction is less than the external reference point (*E*$_r$) (that is, a random number). Thus, orientation (δ) after the dance is expressed as

$$\delta = \left[I_q(d, M) \leq E_r\right] \qquad \textit{Equation 3.43}$$

$$\delta = \alpha * \left[E_r - I_q(d, M))\right] \qquad \textit{Equation 3.44}$$

where α is a random parameter to control the dance, E_r is a specified point of reference, d represents the distance of internal cues, and M represents the magnitude of direction expressed as a random number (between 0 and 1).

3.5 General outline of procedure for bio-inspired/meta-heuristic model

The outline procedure gives a general overview of the steps for the proposed bio-inspired model. As indicated in Table 3.1 on the proposed methodological framework of the bio-inspired model, the steps outlined below were followed during the experiment at each phase. The significance of the experiment is that after analyzing the dataset, the best algorithm was selected as the algorithm with the optimal results. Although each bio-inspired/meta-heuristic algorithm has initial parameters (that is, set by a user), the steps provide a guide on how the best algorithm was selected.

- Step 1: Mathematical modeling as expressed in section 3.2.
- Step 2: Translate the mathematical model into the proposed agent-based search algorithms (Chapters Four, Five and Six)
- Step 3: Implementation of the proposed algorithm using the selected programming language (that is, MATLAB). The reason for the choice of programming language is explained in section 3.7.
- Step 4: Conduct an experiment by testing the implemented algorithm against selected comparative algorithms (indicated in Table 3.1) on a selected dataset or set of datasets, as well as with various initial parameters that are fine-tuned by the algorithm at each iteration. The results of the experiment are presented in subsequent chapters.
- Step 5: Analyze the performance results (in terms of accuracy using MAE, etc.) and use various statistical procedures (such as the Wilxocon-Signed rank test and Friedman test) in order to select the best bio-inspired algorithm. The

analysis results are presented in subsequent chapters.

- Step 6: Analyze the internal implementation code of each algorithm through profiling (that is, extraction of function calls using built-in tools in MATLAB) to understand the underlying behavior of the comparative bio-inspired algorithms in terms of time to call function. Perform the analysis using simple statistical methods such as mean to select the best algorithm.

3.6 Reason for choice of dataset

The dataset used to test the proposed models was obtained from benchmarked datasets from the "classic UCI machine learning repository" (Lichman 2013) (such as stock market data and health-related data) and Arizona State University data repository. The basis for using these datasets is because they represents a standard benchmark dataset for experimental research and they also do have large volume of continuous data, which is suitable for this research work.

The data is downloaded/collected from the online data repositories for the experiment in this thesis. In view of the standard nature of the benchmark dataset, the preparation of data collected for the experiments were not considered. However, the attributes that define these data were taken into consideration. For instance, data attributes includes whether data is continuous or binary, the number of instance in data, the number of features, etc. In chapter 4, 5 and 6, the data attributes suitable for each experiment is indicated.

3.7 Reasons for choice of preferred programming language

The basic mathematical expressions that were formulated from the behavior of animals require the use of a software package that is suitable for the experiment in that it can translate the mathematical expressions into an algorithm that can be executed. The software package chosen for this study is MATLAB. MATLAB has built-in functions and toolboxes to help solve a diverse range of tasks from mathematical operations to graphical display (in either two or three dimensions), which makes it easy to visualize information and gain meaningful insight (Attaway 2009). These built-in functions help perform basic numerical operations and matrix-based computations, encourages simple interactivity (that is, expressions entered by the user are immediately computed and results are displayed) and has complete set of programming constructs that help easily customize a program for a particular problem domain (Attaway 2009). The customized program (in the form of an algorithm) can be deployed within production systems of businesses. Additionally, it has the capability to analyze larger datasets and scale up data to clusters (MathWorks 2017). In view of MATLAB's built-in functions and capabilities, it was thus chosen as an appropriate software package to express the computational mathematics of the proposed bio-inspired/meta-heuristic algorithm for big data analytics and for the display of insight on interesting patterns. Additionally, MATLAB has the built-in capability to support profiling of internal implementation codes of each algorithm to understand the underlying behavior of the algorithms.

3.8 Comparative algorithms

The proposed algorithms were evaluated against comparative algorithms suited for each step of each stage of big data management. Because the nature of different bio-inspired algorithms may make them better suited for certain tasks than other algorithms, these algorithms were included or excluded in the set of comparative algorithms. Consequently, due to this fact, the type and number of comparative algorithms at a specific step of a particular big data management stage may vary slightly.

The experimental results, using a suitable selected set of metrics, generated by the comparative algorithms implemented in Matlab were output into a text file and/or on matlab output screen. These results were put into a table for easy evaluation and comparison (e.g. using MAE etc). In addition to the experiment, further statistical analysis were conducted on the experimental results to find the most suitable algorithm as explained in subsequent chapters.

3.9 Summary

In this chapter, the knowledge claim used was based on positivist approach as it helps understand the theory behind the behavior of animals and how different mathematical expressions are used to model the behavior. The mathematical techniques enable dynamic formulation to depict the different behavioral aspect of the animals being considered. This aspect helped model the three-phase framework. The different animals and their behaviors were related to aspects of big data frameworks, namely velocity and volume characteristics. Basically, the velocity aspect relates to how animals move with speed to capture their prey aided by visual ability, while volume relates to how animals exploit a wide search area for prey.

The ability of animals to capture their prey and exploit a search area for food are some dominant visible behaviors of animals such as the kestrel. Similarly, other animals, such as the dung beetle, show navigation and orientation behavior, and these were explored in this chapter to demonstrate how such behavior can be used to visualize patterns. The general outline provided a general guide in the form of steps to translate the mathematical formulation based on the bio-inspired behavior into the respective algorithms during each phase of the largely bio-inspired framework, as indicated in the methodological framework. MATLAB was chosen as an appropriate software package to implement the computational model because of the robust built-in functions and mathematical capabilities.

In the next chapter, the general outline procedure (see section 3.5) will be followed to empirically test each algorithm using appropriate and diverse datasets characterized as having both high volume and frequently changed items. The results will be tabulated and graphs will also be used to aid visual display and understanding by users.

CHAPTER 4: DEVELOPING, TESTING AND EVALUATING DATA CLEANSING

4.1 Introduction

The chapter validates the first phase of the proposed model, data cleansing, using actual data. The results from the validation process of the algorithms of the proposed model are presented using tables and graphs to aid understanding. During the process of validating the algorithmic structure, various parameters were used to observe the behavior of the model on challenging issues such as missing value estimation, duplicate text detection of health-related records and feature selection in classification of high-dimensional bioinformatics datasets.

The general outline in section 3.5 was followed to model an algorithm for missing value estimation, duplicate data detection and feature selection, as indicated in the methodological framework in Table 3.1. The sub-sections in this chapter discuss missing value estimation, duplicate data detection and feature selection.

4.2 Bio-inspired computational approach to missing value estimation

The kestrel formulation also adopts aspects of swarm behavior in terms of individual searching, moving to better positions and fitness evaluation. However, what makes the kestrel distinctive is the individual hunt through its random encircling of prey and its imitation of the best individual kestrel. Since the kestrel hunts individually and imitates the best features of successful individual kestrels, it suggests that the kestrels is able to remember the best solution from a particular search space and continue to improve upon the initial solution until the near-best solution is reached.

When comparing the unique characteristics of the KSA with the Firefly, Wolf and Bat algorithms, the following can be stated: The Firefly algorithm is based on attractiveness, collective behavior and brightness; the Wolf algorithm is based on attractiveness, collective behavior and escape; the Bat algorithm is based on pulse rate and loudness; and the KSA is based on attractiveness, encircling and brightness of trail, which is dependent on its half-life period. This encircling behavior allows kestrels to be adaptable in searching multiple missing values within a particular search space. The basis for the comparison of algorithms is to assess the interesting behavior of the newly developed algorithm (that is, KSA) and show how different the newly developed algorithm is from previous algorithms in terms of accuracy of results on missing values from datasets.

4.2.1 Fitness function evaluation of algorithms

The fitness function is used to evaluate how well the meta-heuristic algorithm performs in terms of the quality of estimation. This performance is measured in terms of minimizing the deviation of data points from the estimated value without considering the direction (negative or positive) of the fitness value. Thus, the performance evaluation method used the mean of absolute error (MAE) as fitness function evaluation because it allows the model to fine-tune absolute values and improve on performance of values, leading to much finer positive values without consideration of negative values. The MAE is expressed in Equation 4.1 as follows:

$$MAE \;= \frac{1}{n}\sum_{i=1}^{n} |o_i - x_i\,| \qquad \textit{Equation 4.1}$$

Thus, the fitness function is expressed in Equation 4.2 as

$$fitness\ (x) = \frac{1}{n}\sum_{i=1}^{n} |o_i - x_i| \qquad \textit{Equation 4.2}$$

where o_i is the observed data point at the *i*th position in the sampled dataset, x_i is the estimated value at the *i*th position in the dataset, and *n* is the number of data points in the sampled dataset.

4.2.2 Experimental results

An experiment was conducted to evaluate the meta-heuristic algorithms, namely the WSA-MP, BAT and Firefly algorithms as well as the proposed KSA. During the experimental setup, different parameters were defined for each meta-heuristic algorithm, as proposed by the authors of the algorithms, to guarantee optimal solution in any search problem. The experimental results are presented using figures, where the *x*-coordinate represents iteration and the *y*-coordinate represents the fitness value using MAE.

Experimental setup A (WSA-MP to estimate missing values)

The solution algorithm for the proposed meta-heuristic algorithm was implemented in MATLAB 2012A and the quality of estimation was evaluated with the MAE method. The WSA-MP algorithm was compared with the proposed meta-heuristic algorithm (that is, KSA). The basis for the comparison is to obtain results that can demonstrate how well each meta-heuristic algorithm produces the smallest error between the actual and estimated values.

The initial parameters for KSA were set as β_o=1; *visual range*=1. The following arbitrary parameters were set for the lower and higher bound: *zmin*=0.2 and *zmax*=0.8 respectively. A representative dataset was used to test the proposed meta-heuristic algorithm, and a maximum of 500 iterations/generations were done to have a greater chance to further refine the best value in each run. A sample set of data (46 by 9 matrix) with multiple missing values in the row matrix was used in order to provide a thorough test of missing values in each row of a matrix. In WSA-MP, the randomness (σ) parameter was set to 0.2, while escape from local minimum was also set to 0.25.

The technique that is being proposed assumes that only the best estimate obtained by either the WSA-MP or KSA will give the smallest absolute error between the input and the output.

Results on experimental setup A

Figure 4.1 shows the comparison of fitness evaluation of KSA and the WSA-MP algorithm, both using MAE as fitness function.

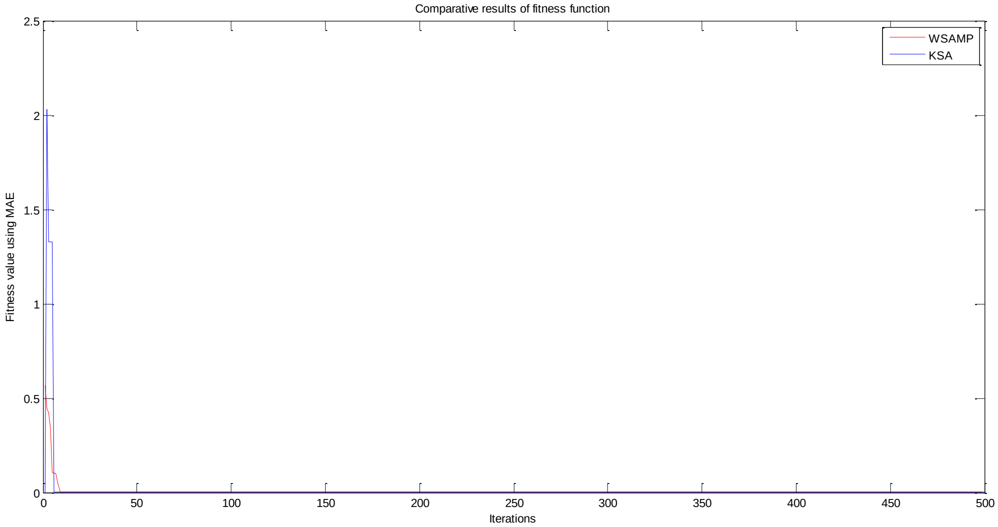


Figure 4.1: Comparison of KSA with WSA-MP algorithm

Figure 4.1 shows the curve on comparison of the fitness evaluation of KSA with WSA-MP in 500 iterations/generations. The fitness curve gradually slopes down on the *x*-axis and maintains a constant fitness, indicating quick convergence at the start of the iteration along the *x*-axis. Table 4.1 indicates the fitness values of the curve of both KSA and WSA-MP as follows:

Table 4.1: Comparison results of KSA and WSA-MP

Algorithm	**Fitness using MAE**
KSA	**7.9912e-05**
WSA-MP	5.6978e-07

Table 4.1 shows the comparative results of KSA with WSA-MP. The resultant fitness values show that WSA-MP has a minimum fitness value of 5.6978e-07, as compared with KSA, which has a fitness value of 7.9912e-05.

4.2.2.1 Conclusions: Experimental setup A

In several iterations that were performed, WSA-MP maintained minimum fitness values compared to KSA. As demonstrated from the results, WSA-MP performed better than KSA, and this performance of WSA-MP suggests that the WSA-MP algorithm utilizes its characteristic good memory on previous best positions, which gives it an edge over random encircling for extrapolating missing values.

Experimental Setup B (Firefly algorithm to estimate missing values)

In the Firefly algorithm, the randomness (σ) and absorption coefficients (γ) were set to 0.2 and 1.0 respectively. This setting allowed a small interval between the random numbers being generated. Meanwhile, randomness reduction was set to 0.97 (similar to an annealing schedule).

Results on experimental setup B

Figure 4.2 shows the comparison of fitness evaluation of KSA and the Firefly algorithm, both using MAE as fitness function.

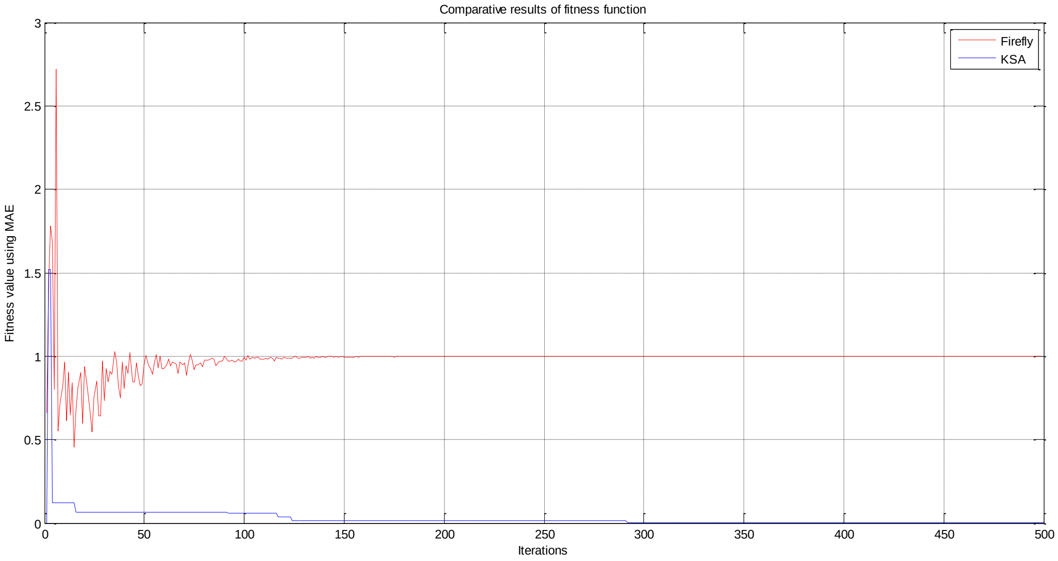


Figure 4.2: Comparison of KSA with Firefly algorithm

Figure 4.2 shows the comparison of KSA with the Firefly algorithm. The curve indicates that KSA converges to a global minimum after the end of the iterations along the *x*-axis. In contrast, the curve of the Firefly algorithm shows several local minimum curves during the start of the iterations, then the curve smoothes until the final iteration on the *x*-axis, suggesting that the curve moves from a local minimum and then gradually lessens to a global minimum. Table 4.2 indicates the fitness values on the *y*-axis of both KSA and the Firefly algorithm:

Table 4.2: Comparison results of KSA and Firefly algorithm

Algorithm	**Fitness using MAE**
KSA	**0.0054204**
Firefly	1.0000

Table 4.2 shows the comparative results of KSA with the Firefly algorithm. The fitness value of KSA converges along the *x*-axis to a value of 0.0054204, while the Firefly algorithm results in a fitness value of 1.0000. This suggests that KSA produces minimum error when estimating missing values.

4.2.2.2 Conclusions: Experimental setup B

The results indicate that KSA has a better performance than the Firefly algorithm. These results suggest that the KSA is best for random encircling and that this algorithm is one of the best in estimating missing values in any big data analysis environment.

Experimental Setup C (Bat algorithm to estimate missing values)

In the Bat algorithm, both the loudness and the pulse rate were set to 0.5 without fine-tuning these parameters. Also, the arbitrary frequency range was set to a minimum of 0.2 and maximum of 0.9. This frequency range determines the frequency scaling of a bat. The Bat algorithm was compared with KSA and the comparative curve of the fitness value on the *y*-axis is illustrated in Figure 4.3:

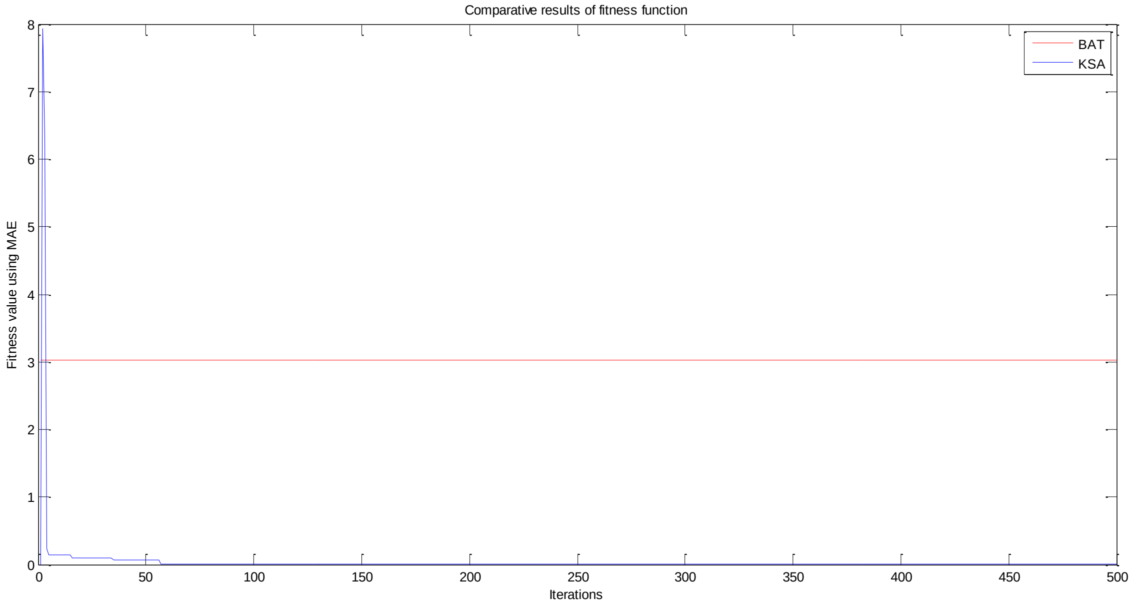


Figure 4.3: Comparison of KSA with Bat algorithm

Figure 4.3 shows the curve on the comparison of the fitness evaluation of KSA with the Bat algorithm in 500 iterations/generations. The fitness curve for KSA peaked at the initial iteration, gradually sloped down along the *x*-axis and maintained a constant fitness value to convergence at 0.002916 along the *x*-axis. Table 4.3 indicates the fitness values and comparative results of KSA and the Bat algorithm:

Table 4.3: Comparison results of KSA and Bat algorithm

Algorithm	**Fitness using MAE**
KSA	**0.0029716**
BAT	3.0326

Results: Experimental setup C

The resultant fitness values show that KSA has a fitness value of 0.0029716, while the Bat algorithm has a fitness value of 3.0326. The Bat algorithm, however, showed a horizontal line from the initial iteration to the end of the iterations.

4.2.2.3 Conclusions: C

The results suggest that the Bat algorithm was unable to converge to a global minimum. Thus, using the Bat algorithm for estimating missing values at random results in a high error of estimation. In contrast, KSA showed minimum error in estimating missing values at random and it also converged to a global minimum.

Different Problem dimensions/scales

Different problem dimensions/scales of the dataset were applied to each algorithm, and the corresponding fitness value (that is, MAE) was computed. The dimensions that were selected helped observe the behavior of each meta-heuristic algorithm on a different problem scale. Table 4.4 show the results on MAE values obtained from the comparative meta-heuristic algorithms on different problem dimensions as follows:

Table 4.4: MAE results from comparative algorithms on different problem scales

Problem dimension	KSA	BAT	Firefly	WSA-MP
	MAE	**MAE**	**MAE**	**MAE**
40x9	7.09E-05	3.0326	0.90723	8.16E-07
30x9	0.012553	3.0843	1	2.51E-07
20x9	0.04752	3.0655	0.15362	9.22E-06
25x9	0.023865	3.3836	1	1.34E-07
10x9	0.39469	3.536	0.6943	1.73E-05
Mean	7.98E-02	3.19E+00	7.93E-01	4.72E-06

Table 4.4 further indicates that irrespective of problem scale, KSA obtained optimal results compared to the bat and Firefly algorithms. However, WSA-MP has the most optimal result. Secondly, the mean of MAE indicates that WSA-MP has the minimum value of 4.72E-06, compared to KSA (7.98E-02), firefly (7.93E-01) and bat (3.19E+00). The results suggest that in a frequently changed dataset, the WSA-MP algorithm produces

the most optimum values as estimates of missing values while the proposed KSA showed potential for finding optimal values.

4.2.3 Statistical analysis of experimental results

The basis for the statistical analysis of experimental results on the comparative algorithms is to find the significance of results obtained from each algorithm. In order to do this comparison in an accurate manner, a profile was used on all the test functions used in each of the algorithms and the MAE results (that is, the quality of estimation) in Table 4.4. A non-parametric statistical procedure was used to analyze the significance of the results. This statistical procedure was used as it does not make underlying assumptions about parameters, such as mean and variance of the algorithm being assessed. In contrast, parametric statistical procedures make assumptions on parameters that are being assessed. In this section, the profiling of test functions and the non-parametric statistical procedure adopted for the analysis are discussed. This section is divided into two parts: statistical analysis on profiling of test functions and statistical analysis on MAE results (quality of estimation).

4.2.3.1 Statistical analysis on profiling of test functions

Profiling is a technique used to extract functions and measure time spent on aspects of a program, such as a function (Sorensen *et al*. 2012). This technique helps optimize functions and improve on performance of the algorithms. During profiling, the following are considered: function name, the number of times a function was called upon (Calls), the total time spent on each function, including sub-functions (Total_Time), and total time spent on a function excluding the time spent on sub-functions (Self_Time). It is possible for functions that are less time intensive to call other functions that are more time intensive. Profiling technique is important as it determines which functions are responsible to call other functions.

Profiling was applied as a technique to extract functions and to group the functions into two categories, namely major functions and basic functions. While the major functions are functions that were written to implement the behavior of the algorithms, the basic functions are the in-built functions that work alongside the major functions. Table 4.5 indicates how the functions are grouped during profiling on each comparative algorithm:

Table 4.5: Major function names of the comparative algorithms

	Function name	Calls	Total_Time (s)	Self_Time* (seconds (s))
	WSA-MP algorithm			
f2	WSA-MPnew>fnc_fitness_mae	172517	1.298	1.298
f1	WSA-MPnew>main	1	14.426	12.993
	Mean	*86259*		
	Firefly algorithm			
f2	fireflyApproachnew>fnc_fitness_MAE	4522	0.018	0.018
f3	fireflyApproachnew>ffa_move	500	0.415	0.404
f4	fireflyApproachnew>findrange	500	0.011	0.011
f5	fireflyApproachnew>newalpha	500	0.002	0.002
f1	fireflyApproachnew>main	1	47.811	29.291
f6	fireflyApproachnew>init_fireflyalg	1	0.001	0.001
	Mean	*1004*		
	KSA algorithm			
f2	KSAapproachnew>fnc_fitness_MAE	23046	0.094	0.094
f3	KSAapproachnew>fnchalflife	500	0.002	0.002
f4	KSAapproachnew>fncbobbing	500	0.011	0.011
f1	KSAapproachnew>main	1	1.329	1.036
	Mean	*6011.8*		
	BAT algorithm			
f2	BATApproachnew>fnc_fitness_MAE	23046	0.101	0.101
f3	BATApproachnew>simplebounds	23000	0.317	0.317
f1	BATApproachnew>main	1	1.353	0.718
	Mean	*15349*		

Table 4.5 shows the different test functions in each comparative algorithm. The WSA-MP has two major functions, categorized into the main function (represented by f1) and a sub-function (represented by f2). All main functions are represented in each algorithm by f1. The Firefly algorithm has six major functions (one main function f1 and five sub-functions). The Bat algorithm consists of three major functions (one main function f1 and 2 sub-functions) while KSA consists of 4 major functions (one main function f1 and three sub-functions).

In order to obtain a true reflection of the nature of in-built functions that were extracted, all in-built functions were considered for analysis. It was observed that when some in-built functions were called, the total time was zero seconds, thus making those in-built function calls inconsequential in terms of execution time. However, these inconsequential in-built functions were taken into consideration so as not to lose track of any function calls made.

A statistical procedure was applied to analyze the significance of profile results obtained in Table 4.5. A non-parametric statistical test was conducted to assess which of the algorithms have better performance in terms of the behavior of test function call time. The basis for the statistical analysis is to find out the significance of the profiled results from each algorithm. García, Fernández, Benítez and Herrera (2007) state that non-parametric or distribution-free statistical procedures help perform pairwise comparison on related algorithms, even in the case where the sample size of a dataset is small, such as where sample size $n < 30$. In a multiple comparison situation, such as in this thesis, the Wilcoxon signed-rank test could be applied to rank and test how significant algorithms outperform each other in respect of detecting the differences in the mean time to call test functions (García, Fernández, Benítez and Herrera 2007) and to find the probability of an error by suggesting that the medians of two algorithms are equivalent. This probability is called *p*-value (Zar 1999). The advantage of the Wilcoxon test is that there is no need to make assumptions about the population of functions being used for the experiment. Since the

Wilcoxon signed-rank test can guarantee about 95% of efficiency (that is, 0.05 level of significance) if the population is normally distributed. This suggests that if there are 500 observations on test function calls, then Wilcoxon signed-rank is efficient to about 499 observations on test function calls.

García, Molina, Lozano and Herrera (2008) observe that the Wilcoxon signed-rank test is an alternative to the paired-sample *t*-test. The present study used the Wilcoxon test because the researcher did not wish to make the assumptions necessary for the *t*-test for sample distribution. Samples are related if one sample matches the other sample, while the rank is a number assigned to an individual sample according to its order in a list of algorithms. Thus, the Wilcoxon statistical technique helps assign ranks to algorithms in order to identify the best-ranked behavior of evolutionary algorithms (García, Molina, Lozano and Herrera 2008) and to determine the significance of each algorithm. The following steps were applied in computing the Wilcoxon signed-rank test:

Step 1: Compute the difference *D* of paired samples in each algorithm. Any pairs with a difference of 0 are discarded

Step 2: Find the absolute *D*.

Step 3: Compute the rank of signs (*R*+ difference and *R*- difference) from lowest to highest. The sum of ranks is expressed in Equation 4.3 by

$$\sum R_{+} + R_{-} = \frac{n(n+1)}{2} \qquad \textit{Equation 4.3}$$

where *n* is sample size.

Step 4: Compute the test statistic *T*. Thus, *T* = *min*{R+, |R-|}. Thus, the test statistic *T* is the smallest value.

Step 5: Find the critical values based on the sample size *n*. If *T* is less than or equal to the critical value at a level of significance (that is, α =0.05), then a decision is made that algorithms are significantly different (García, Fernández, Benítez and Herrera 2007). In order to accomplish this, the Wilcoxon signed-rank table is consulted, using the critical value (α =0.05) and sample size *n* as parameters, to obtain the value within the table. If this value is less than the calculated value of the algorithmic comparison, this means that the algorithmic difference is significant.

In order to apply the Wilcoxon signed-rank test, an analysis was performed on both the time to call and function name (both in-built functions and major functions) as follows:

i. a. In-built functions calls time analysis

The performance of the comparative algorithms was based on test function call time differences between the self time and total time of built-in functions in each algorithm. Based on the steps in computing the Wilcoxon signed-rank test, the time for the test function call per algorithm is shown in Table 4.6.

Table 4.6: Wilcoxon rank on profile extracts of built-in-in function calls of algorithms

	Algorithms	Sum of calls	Total number of built-in-in functions	Sum of self_Time	Sum of total_Time	D	R+/R-	Rank	Sum of signed rank
1	WSA-MP	123	31	0.132	0.401	0.269	1	1	1
2	Firefly	99780	229	49.228	160.73	111.503	1	4	4
3	KSA	1043	34	0.184	0.471	0.287	1	2	2
4	BAT	105	69	0.215	0.953	0.738	1	3	3

In Table 4.6, *D* represents the difference between the sum of total time and the sum of self time. It is observed that the sum of signed positive ranks $R+$ is 10, while the sum of negative ranks *R*- is 0. Since the sample size *n* (4) is less than 30 and the Wilcoxon signed-rank table shows that there is no critical region on the sub-function at $\alpha = 0.05$, the

Wilcoxon signed-rank table suggests that built-in-in functions are equivalent. In terms of ranking of algorithms, the WSA-MP was ranked first while KSA was ranked second.

Since *n* is small, it is tedious to find a critical value for small values of *n*. Table 4.7 shows the mean and standard deviation on the sum of self time and sum of total time on the sub-function.

Table 4.7: Mean and standard deviation of built-in-in functions

Sum	N	Mean	Std. Deviation	Minimum	Maximum
Self_time	4	12.4398	24.52552	.13	49.23
Total_time	4	40.6387	80.06121	.40	160.73

Table 4.7 shows sample size *N*, the mean of the sum of self time as 12.4398 and the sum of total time as 40.6387, with their corresponding standard deviations. The results show a standard deviation of 80.06121 on the total time, and this suggests a high deviation of total time on the built-in function calls as compared with the low standard deviation of 24.52552 on self time built-in function calls. Therefore, there is a high total time spent on built-in function calls in the algorithms as compared to self time on built-in function calls. This means the algorithms spent an intensive amount of time on calling an average of 40.6387 built-in functions and an average time of 12.4398 on excluding other built-in function calls. Table 4.8 illustrates the Wilcoxon signed-rank test between total time and self time of built-in function calls computed using the Statistical Package for the Social Science (SPSS). Table 4.8 contains the sample size *N*, mean rank and sum of ranks.

Table 4.8: Wilcoxon signed ranks on built-in functions

		N	Mean rank	Sum of ranks
Sum of total_time Sum of self_time	Negative ranks	0[a]	.00	.00
	Positive ranks	4[b]	2.50	10.00
	Ties	0[c]		
	Total	4		

a. Sum of total_time < Sum of self_time

b. Sum of total_time > Sum of self_time

c. Sum of total_time = Sum of self_time

Table 4.8 shows Wilcoxon signed ranks on the comparison of the sum of total time and the sum of self time. There were four samples between total time and self time. The basis was to find out if the differences between total time and self time are significantly different from zero and if the differences that were observed in the mean rank (0.00 versus 2.50) can be located in the population of built-in function calls. In order to locate the value between the mean rank (0.00 versus 2.50), the test of significance of time on performance of built-in functions was computed, as shown in Table 4.9.

Table 4.9: Test statistics on built-in functions

	Sum of total_time - Sum of self_time
Z	-1.826[a]
Asymptotic Sig. (2-tailed)	.068

a. Based on negative ranks

b. Wilcoxon signed-ranks test

Table 4.9 shows the test statistic that was obtained. The Asymptotic Sig. (2-tailed) in the table represents the *p*-value for the test, while the Wilcoxon signed-ranks test was computed using the *z* statistic. Thus, the Wilcoxon signed-rank test, which was used on four samples to find out whether there is a significant change of total time with self time, shows *z*=-1.826 and *p*=0.068. Since p> $\alpha(0.05)$ within the mean rank (0.00 versus 2.50), the value of 0.068 indicates that, statistically, time did not result in a significant change in performance of built-in function calls.

ii. b. Major function calls time analysis

Statistical analysis was also carried out to determine whether time might be significant in enhancing the performance of major function calls of each algorithm. Table 4.10 shows the summary of the Wilcoxon signed-rank test on major functions between self time and total time of each algorithm.

Table 4.10: Wilcoxon rank on profile extracts of major function total time and self time of algorithms

	Algorithm	Sum of calls	Total number of major functions	Sum of self_time	Sum of total time	D	R+/R-	Rank	Signed rank
1	WSA-MP	172518	2	14.291	15.724	1.433	1	1	1
2	Firefly	6024	6	29.727	48.258	18.531	1	1	1
3	KSA	24047	4	1.143	1.436	0.293	1	1	1
4	BAT	46047	3	1.136	1.771	0.635	1	1	1

Table 4.10 shows the difference *D* between the sum of total time and the sum of self time of each algorithm. It is observed that the sum of signed positive ranks $R+$ is 4, while the sum of negative ranks *R-* is 0. The sample size *n* (4) is less than 30, and from the Wilcoxon signed-rank table it is evident that there is no critical region on the significance level of $\alpha = 0.05$ to suggest that the major functions are significantly different in terms of total time and self time of calling the major functions. All the major functions of each algorithm were ranked equally. The Wilcoxon signed-rank test was conducted to test if there was a significant difference. Firstly, Table 4.11 indicates the mean and standard deviation of both the sum of self time and the sum of total time of the major functions.

Table 4.11: Mean and standard deviation on major functions

	N	Mean	Std. Deviation	Minimum	Maximum
Sum of self_time	4	11.5742	13.59744	1.14	29.73
Sum of total_time	4	16.7973	22.00520	1.44	48.26

Table 4.11 shows sample size *N*, the mean of sum of self time as 11.5742 and the sum of total time as 16.7973 with their corresponding standard deviations. The results indicate a standard deviation for total time as 22.00520 and self time as 13.59744. Thus, there is a high variation in total time as compared with low variation in self time of major function calls. Table 4.12 illustrates the Wilcoxon signed-rank test between total time and self time of major function calls.

Table 4.12: Wilcoxon signed ranks of major functions

		N	Mean rank	Sum of ranks
Sum of total_time Sum of self_time	Negative ranks	0[a]	.00	.00
	Positive ranks	4[b]	2.50	10.00
	Ties	0[c]		
	Total	4		

a. Sum of total_time < Sum of self_time

b. Sum of total_time > Sum of self_time

c. Sum of total_time = Sum of self_time

Table 4.12 shows Wilcoxon signed ranks of the comparison of the sum of total time and the sum of self time. There were four samples between total time and self time. The reason for this comparison was to find out if the differences between total time and self time are significantly different and if the differences that were observed in the mean rank (0.00 versus 2.50) can be located in the population of major function calls. In order to locate the value between the mean rank (0.00 versus 2.50), the test of significance of time on performance was computed, as shown in Table 4.13.

Table 4.13: Test statistics on major functions

	Sum of total_time - Sum of self_time
Z	-1.826[a]
Asymptotic Sig. (2-tailed)	.068

a. Based on negative ranks

b. Wilcoxon signed-ranks test

Table 4.13 shows the test statistic that was obtained. The Asymptotic Sign. (2-tailed) in the table represents the *p*-value for the test, while the Wilcoxon signed-rank test was computed using the *z*-statistic. The Wilcoxon signed-rank test was used to find out whether there is significant change in total time and self time at *z*=-1.826 and *p*=0.068. The results indicate that, statistically, time did not result in a significant change in performance of major functions calls.

4.2.3.2 Statistical analysis of output results on quality of estimation

The Wilcoxon signed-rank test was conducted on all the dimensions of results on quality of estimation, as shown in Table 4.14.

Table 4.14: Results on accuracy from comparative algorithms using MAE

Problem dimension	KSA	BAT	Firefly	WSA-MP
	MAE	MAE	MAE	MAE
46x9	7.99E-05	3.0326	1.000	5.70E-07
40x9	7.09E-05	3.0326	0.90723	8.16E-07
30x9	0.012553	3.0843	1.000	2.51E-07
20x9	0.04752	3.0655	0.15362	9.22E-06
25x9	0.023865	3.3836	1.000	1.34E-07
10x9	0.39469	3.536	0.6943	1.73E-05
Mean	**7.98E-02**	**3.19E+00**	**7.93E-01**	**4.72E-06**

Table 4.14 consists of all problem dimensions of each algorithm and the respective MAE. Although the null hypotheses were formulated based on the 46x9 dimension, the hypotheses on all problem dimensions of the MAE value were tested. Earlier, the analysis of the performance of each paired algorithm showed the following:

1. WSA-MP outperformed KSA in terms of the minimum error (MAE)
2. KSA outperformed the Bat algorithm in finding the optimal value.
3. KSA produced minimum error when estimating missing values when compared with the Firefly algorithm.
4. WSA-MP produced minimum error when estimating missing values, compared with the Bat algorithm.
5. The Firefly algorithm outperformed the Bat algorithm in terms of minimum error.
6. WSA-MP outperformed the Firefly algorithm in terms of minimum error.

In order to test the significance of quality of estimation, the Wilcoxon test statistic was computed using SPSS, and the *p*-value is shown in Table 4.15.

Table 4.15: Wilcoxon signed-rank test statistic on accuracy

Comparative algorithm	Asymp. Sig. (2-tailed) *p*-value
KSA vs. WSA-MP	.028
KSA vs. Firefly	.028
KSA vs. Bat	.028
Firefly vs. Bat	.028
WSA-MP vs. Firefly	.028
WSA-MP vs. Bat	.028

Table 4.15 shows the *p*-values that were obtained from each comparing algorithm. Since the *p*-values must be less than or equal to the level of significance of 0.05 in order to be significant, the results in Table 4.15 show that the quality of estimations were significant between the paired algorithms. In this case, the WSA-MP significantly outperformed KSA in terms of the MAE.

4.2.3.3 Multiple comparison of output results on accuracy

Trawiński *et al*. (2012) note that the Wilcoxon signed-ranked test is best used for pairwise comparisons between two algorithms. In a multiple comparison situation, where two or more algorithms are compared, it is possible for errors to accumulate such that performance of algorithms is significant. García, Fernández, Benítez and Herrera (2007) state that performing multiple comparison enables the researcher to correct the Family-Wise Error Rate (FWER), which occurs after multiple algorithms are combined. In order to do this comparison, García, Fernández, Benítez and Herrera (2007) use the results of accuracy obtained by algorithms to perform statistical analysis on algorithms. The statistical significance of combining pair of algorithms is computed using the following equations:

$$p = P\ (Reject\ H_o | H_o\ true) \qquad \textit{Equation 4.4}$$

$$p = 1 - P\ (Accept\ H_o | H_o\ true) \qquad \textit{Equation 4.5}$$

$$p = 1 - P\ (Accept\ A_k = A_i, i = 1, \ldots, k-1 | H_o\ true) \qquad \textit{Equation 4.6}$$

$$p = 1 - \prod_{i=1}^{k-1} P\ (Accept\ A_k = A_i | H_o\ true) \qquad \textit{Equation 4.7}$$

$$p = 1 - \prod_{i=1}^{k-1} [1 - P\ (Reject\ A_k = A_i | H_o\ true)] \qquad \textit{Equation 4.8}$$

$$p = 1 - \prod_{i=1}^{k-1} (1 - pH_i) \qquad \textit{Equation 4.9}$$

Using Equation 4.9, *p*-values of each algorithm are computed to find the final *p*-value. If the *p*-value is less than the critical value (e.g. α =0.05), then it forms the basis for rejection of a hypothesis. However, a final decision cannot be made to fully reject or fail to reject (accept) a hypothesis based on an analysis result without performing a test on the possible errors that could have accumulated when comparing algorithms.

In a multiple comparison situation, testing the differences between more than two evolutionary algorithms and avoiding the accumulation of error can be achieved through the Friedman test (Friedman 1940; 1937). The Friedman test is a two-way analysis of the variations in the ranking of algorithms. The Friedman test is a non-parametric procedure that aims to compare the median of a distribution in order to find out if significant differences have occurred between the behavior of two or more algorithms. The null hypothesis of the Friedman test applies equality of medians (García, Luengo and Herrera 2015), while the alternative hypothesis negates a null hypothesis. The Friedman test procedure can be summarized into the follow steps:

Step 1: Rank algorithms separately for the dataset.

Step 2: The best performing algorithm with least MAE gets the rank of 1, the second best the rank of 2, etc.

Step 3: If there is a tie between ranks, assign the average rank. Let r_i^j represent the rank of the j^{th} of k algorithm on the i^{th} of N dataset.

Step 4: Compare the average ranks of the algorithm using Equation 4.10:

$$R_j = \frac{1}{N} \sum_i r_i^j \qquad \textit{Equation 4.10}$$

where r_i^j represents the rank of the j^{th} of k algorithm on the i^{th} of N dataset. The null hypothesis computes the equivalence and their ranks R_j, which is equal to the Friedman statistic (Friedman 1940; 1937), computed in Equation 4.11 as:

$$X_F^2 = \frac{12}{nk(k+1)} \left[\sum_j R_j^2\right] - 3n(k+1) \qquad \textit{Equation 4.11}$$

where R_j is the rank, X_F^2 is distributed with k-1 degrees of freedom, such that n and k should have a large sample size (n) (as a rule of a thumb, n >10 and k>5) (García, Luengo and Herrera 2015) since large sample sizes are significant in computing the degree of freedom on the rank of algorithms. Finally, k is the number of groups that are being compared.

Step 5: The calculated value of X_F^2 must be larger than or equal to the appropriate critical table value of X^2 or larger than or equal to the value of X_F^2 in the small samples table.

García, Luengo and Herrera (2015) indicate that to perform multiple comparison, two measures are used. The first is to check whether the results obtained from the algorithm have inequality and rank using the Friedman test. The Friedman test states that under a null hypothesis, all the algorithms are equivalent, so a rejection of a hypothesis indicates the existence of significant differences in performance of all the algorithms studied (García, Luengo and Herrera 2015).

In the present approach to identify the best algorithm (deemed to be the algorithm with the lowest ranking value) that can be used as a control algorithm, the results in Table 4.14 were applied and the Friedman test was conducted to identify the best algorithm. In order to rank the algorithms, the mean and standard deviation were computed, as shown in Table 4.16.

Table 4.16: Descriptive statistics

	N	Mean	Std. Deviation	Minimum	Maximum
KSA	6	.0000	.00001	.00	.00
WSA-MP	6	.7925	.33471	.15	1.00
BAT	6	.0798	.15528	.00	.39
Firefly	6	3.1891	.21606	3.03	3.54

The results n Table 4.16 indicate that the KSA has the least standard deviation among the comparative algorithms. The standard deviation measures the amount of variation in a set of data (Gordon and Gordon 1994). Thus, the larger the standard deviation, the greater the variation in the data, while the smaller the standard deviation, the smaller the amount of variation in the data. Since the KSA has minimum standard deviation of 0.00001, there is small variation in the KSA. Based on the results in Table 4.16, the Friedman test ranked the algorithms as shown in Table 4.17.

Table 4.17: Descriptive statistics

	Mean Rank
BAT	2.00
Firefly	4.00
WSA-MP	3.00
KSA	1.00

The ranks in Table 4.17 indicate that the KSA is the best algorithm among the comparative algorithms. The Friedman test statistic, with a sample size *N*, was then computed, as shown in Table 4.18.

Table 4.18: Friedman test statistics

N	6
Chi-Square X^2	18.000
df	3
Asymp. Sig. (*p*-value)	.000

a. Friedman Test

Table 4.18 shows the results of the Friedman test, where X^2 obtained is 18.000, with 3 degrees of freedom and a significance (Asymp Sig) level of 0.0000. Since the significance level is α (0.05), the computed value on X^2 must be larger than or equal to the critical value for significance of 0.05. Since *df* is 3 at 0.05 level of significance, the value that was read from the critical value of the chi-square X^2 distribution table (Hinton, 1995) is 7.82, thus 18 > 7.82 at α (0.05). There is a significant difference in the results on quality of estimation of missing values among the algorithms, meaning the algorithms are not the same.

4.2.4 Conclusion

These results on the performance of accuracy of results from each algorithm were arrived at from the profiling conducted on the built-in function call time and major function call time on comparative algorithms. The reason for using the Wilcoxon signed-rank test was to rank the various evolutionary algorithms in respect of minimum error. The lowest minimum error of estimation is acceptable as the best output results. Statistical analysis (using the Wilcoxon signed-rank test) conducted on accuracy results using different dimensions of the same dataset indicated that the proposed KSA outperformed both the bat and Firefly algorithms. However, WSA-MP outperformed KSA in terms of the minimum error.

Further statistical analysis (Friedman test) was conducted to test the error rate when two or more algorithms are combined. The reason for using the Friedman test was to compare the median of a distribution to find out if significant differences occurred between the behavior of two or more algorithms. Thus, the null hypothesis is that all comparative algorithms are equivalent at a confidence interval value of 0.05. The results (shown in Table 4.18) suggested that there is a significant difference in the quality of estimation among the algorithms, thus the meta-heuristic algorithms are not the same.

4.3 Duplicate data (text data) detection

With the ever-increasing volume of data generated every second, matching records to the name of a correct person becomes increasingly complicated as organizations share records electronically using different systems that may lead to an increased chance of identity error, particularly as electronic information becomes more prevalent. The healthcare sector is one of the sectors with widespread use of electronic information exchange of patient data (Morris *et al.* 2014). As the use of health systems increases exponentially, the accuracy of identifying and matching records has been recognized as a major challenge (Morris *et al.* 2014). In order to resolve this challenge, duplicate detection algorithms are used. This study discusses duplicate detection algorithms and compares the accuracy of two frequently used algorithms, namely the Smith-Waterman and the Jaro-Winkler algorithms. Although either algorithm might be appropriate when a small dataset is used, it leads to a small amount of loss of data. However, large amounts of data can cause an issue because as the dataset grows large, the risk of data loss is increased. Hence, transitive and symmetry properties of both the Jaro-Winkler and Smith-Waterman algorithm are used to handle large amounts of data involved in the duplicate detection process.

4.3.1 Experimental setup

The experimental setup describes how the Jaro-Winkler algorithm and Smith-Waterman algorithm were implemented in MATLAB to demonstrate the transitivity and reflexive property. During the experimental setup, a 5x1 synthetic matrix dataset was created as a representative dataset with multiples of duplicate words. The basis of this experiment was to determine the best threshold for match words in both comparative algorithms. After several preliminary trial tests of the synthetic matrix dataset, as a basis for setting the gap penalty for the Smith-Waterman algorithm, two measures were taken into consideration in order to avoid setting too small or large values. While small values allow a previously accumulated local alignment to continue with an insertion in one of the sequences, large gap values lead to previous alignment scores being removed completely. The score on a matched word was set to 1, mismatched set to -1, and gap set to -1. In the case of Jaro-Winkler distance, a distance value equal to 1 indicates a duplicate string, while a distance value less than 1 indicates non-duplicate strings. Thus, the higher the distance value, the higher the matched score.

The proposed steps on the present approach to duplicate detection are summarized as follows:

Step 1: Load data from repository.

Step 2: Initialize penalty gaps.

Step 3: Perform tokenization of words and pairwise comparison of words.

Step 4: Apply character-based methods to compute the similarity score (using both the Jaro-Winkler and the Smith-Waterman algorithm independently).

Step 5: Check the reflexive property of equality.

Step 6: Output results.

Table 4.19 illustrates the proposed algorithm for duplicate detection.

Table 4.19: Proposed algorithm

```
Step 1: Load data from repository
Step 2: Initialize penalty gap
Step 3: Perform tokenization
Get (current word);
        Get (next word);
      // Apply transitive property through the use of loops
        WHILE each word
         WHILE each word is not empty
             Pick each character in a word
         END WHILE
                Get (next word);
             // pairwise comparison of words using the equality property
current word= next word
        END WHILE
Step 4: Compute similarity score
Step 5: Check symmetry property of equality
Step 6: Output results
```

The proposed algorithm on duplicate text/word starts with input of data from the data source. Initially, the original Smith-Waterman algorithm and Jaro-Winkler algorithm allow the comparison between only two words to detect the similarity score. The current structures of these algorithms were not suitable to be used in this experiment as they could not allow comparison of multiple words in rows of a dataset. Thus, the approach in this study applied Step 3 of the proposed algorithmic structure in Table 4.19. During Step 3, two words from the dataset are selected, where the current word is selected from position *i* in a row and the next word is selected from position *i*+1 in a row of the same dataset for initial comparison. For instance, current word (referred to as *x*): G C C A U U G and next word (referred to as *y*) G C C - U C G, with respective lengths *x* and *y*. After the lengths of *x* and *y* are compared, each character in a word is compared to find the pairwise alignment, as it is possible for words to be of equal length but different in terms of the characters in each word. After the pairwise alignment, Step 4 is applied, as indicated in the walkthrough examples presented earlier, on both the Smith-Waterman and Jaro-Winkler algorithm to compute the similarity score. If words are similar, then both would have the same similarity score and the word is stored as duplicate (represented as *z*). During Step 5, the symmetry property of equality is applied, as illustrated earlier. Afterwards, the duplicate word *z* is assigned to the next comparison process with another word on the *i*+2 position in the row of the dataset. The iteration process continues until the entire search process ends. This iteration process constitutes the workflow of the proposed algorithm to enhance both the Smith-Waterman algorithm and the Jaro-Winkler algorithm.

During the experimental setup, a program was written in MATLAB to implement the iteration process, and a preliminary 5x1 synthetic matrix dataset was created as a representative dataset with multiples of duplicate word to test the robustness of the solution algorithm. In the case of the Smith-Waterman algorithm, parameters for the cost of single gap *c* was set to -1, while the cost of a matched word was set to 1, and the mismatched was set to -1. Thus, a matrix score of 1 means words are duplicate; otherwise words are non-duplicate. In the case of Jaro-Winkler algorithm, a distance value of 1 means words are duplicate, otherwise words are non-duplicate. Thus, the higher the distance value, the higher the chances of words being duplicate or similar.

After the preliminary test of the algorithm, the proposed steps of the algorithm were refined and executed on a 209x1 matrix health disease-related dataset extracted from a typical online data repository called "UCI machine learning data repository" (Lichman 2013). This instance of the dataset was extracted and applied as it consists of different health diseases related to chest pain, with multiple duplicates that are suitable for this study. Although the matrix size may not be considered large, it reflects how the comparative algorithms can perform when a dataset with 209 words is applied.

4.3.2 Experimental results

In order to observe the accuracy of results after applying Step 3 and Step 4 of the proposed algorithmic structure, which forms part of the enhancement of both algorithms, for the purpose of clarity, the initial enhancements (that is, the use of Step 3 and Step 4) are referred to as a partially enhanced Smith-Waterman algorithm and a partially enhanced Jaro-Winkler algorithm. After a thorough test of the matrix dataset, the following observations were made, and the experimental results are displayed as follows:

4.3.2.1 Results on Smith-Waterman algorithm

Table 4.20 illustrates the experimental results obtained on the partially enhanced Smith-Waterman algorithm.

Table 4.20: Results of partially enhanced Smith-Waterman algorithm

Algorithm	Number of duplicate (match) words detected	Number of mismatch words detected
Partially enhanced Smith-Waterman	101	109

The results displayed in Table 4.20 indicate that the number of duplicate (match) words detected in the dataset was 101, and the number of mismatch words detected was 109. The rate of matched and mismatched words over number of words using the same algorithm is illustrated in Table 4.21.

Table 4.21: Results of partially enhanced Smith-Waterman algorithm rate of match and mismatch

Algorithm	Rate of pairwise match of word	Rate of mismatch of word
Partially enhanced Smith-Waterman	0.48325	0.51675

The results displayed in Table 4.21 show the rate of pairwise match as 0.4825 and the rate of mismatch as 0.51675. This shows a high rate of mismatched words compared to matched words. During pairwise comparison of words, the proposed algorithm selects the first word in a row and sequentially compares it with each other word using the transitive property. However, the second word in the row was not sequentially compared in the subsequent iterations, and this resulted in the high number of duplicate words detected. Figure 4.4 shows the nature of the undirected graph.

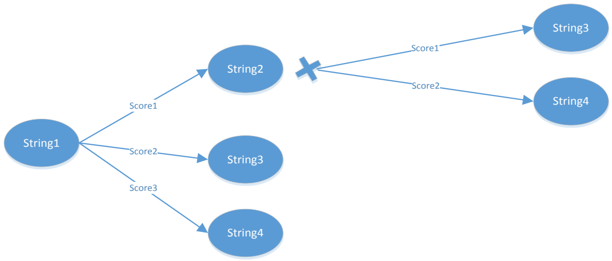


Figure 4.4: Nature of break in transitivity in Smith-Waterman algorithm

Figure 4.4 shows a break “x” in sequential comparison within the next iteration from String 2 to String 3 and String 2 to String 4. The string number represents words where each word is considered a node. The undirected graph indicates that other strings were not compared in subsequent iterations; thus, other duplicate words were not detected. In order to solve this break in the pairwise word comparison, the property of equality (symmetry property), which includes Step 5 of the proposed algorithm structure, was applied at the end of each iteration. For the purpose of clarity of results, the final algorithm is referred to as the fully enhanced Smith-Waterman algorithm. The results of the fully enhanced Smith-Waterman algorithm are displayed in Table 4.22.

Table 4.22: Results of fully enhanced Smith-Waterman algorithm after break in sequential comparison

Algorithm	Number of duplicate (match) words detected	Number of mismatch words detected
Fully enhanced Smith-Waterman	74	135

The results displayed in Table 4.22 show the number of duplicate (match) words detected as 74, and the number of mismatch words detected as 135. Table 4.23 shows the rate of matched and mismatched words over number of words (209), using the same algorithm.

Table 4.23: Results of fully enhanced Smith-Waterman algorithm rate of match and mismatch after break in sequential comparison

Algorithm	Rate of pairwise match of word	Rate of mismatch of word
Fully enhanced Smith-Waterman	0.35407	0.64593

The results displayed in Table 4.23 show the rate of match words as 0.35407 and the rate of mismatch as 0.64593. Thus, 35% of words were detected as duplicate, while 64% of words were detected as mismatch.

These results show that while there was a decrease in the rate of match from 0.48325 (Table 4.21) of the partially enhanced algorithm to 0.35407 (Table 4.23) of the fully enhanced algorithm, there was also an increase in the rate of mismatch from 0.51675 (Table 4.21) of the partially enhanced algorithm to 0.64593 (Table 4.23) of the fully enhanced algorithm. This indicates a reduction in the number of duplicate words detected, from 101 (Table 4.20) of the partially enhanced algorithm to 74 (Table 4.22) of the fully enhanced algorithm, and an increase in the number of mismatch words detected from 109 (Table 4.20) of the partially enhanced algorithm to 135 (Table 4.22) of the fully enhanced algorithm. This suggests that more mismatched words that are not duplicate words were further detected in the fully enhanced Smith-Waterman algorithm.

4.3.2.2 Results of partially enhanced Jaro-Winkler algorithm

The experimental results on the partially enhanced Jaro-Winkler algorithm are presented in Table 4.24.

Table 4.24: Results of partially enhanced Jaro-Winkler algorithm

Algorithm	Number of duplicate (match) words detected	Number of pairwise mismatch words detected
Partially enhanced Jaro-Winkler	101	107

The results displayed in Table 4.24 indicate that the number of duplicate (match) words detected was 101, and the number of pairwise mismatch of words detected was 107. Table 4.25 shows the results on the rate of pairwise match and rate of pairwise mismatch.

Table 4.25: Results of partially enhanced Jaro-Winkler algorithm rate of match and mismatch

Algorithm	Rate of pairwise match	Rate of pairwise mismatch
Partially enhanced Jaro-Winkler	0.48558	0.51442

The results displayed in Table 4.25 show the rate of pairwise match as 0.48558 and the rate of pairwise mismatch as 0.51442. Thus, 48% of words were detected as duplicate while 51% of words were detected as mismatch.

The experimental results demonstrate that the partially enhanced Jaro-Winkler algorithm considers a pair of strings (words/tokens) and performs pairwise comparison of two words. Since the algorithm considers words as paired, it does not perform a comparison on the n^{th}-1 instance (row) if the number of instances is an odd number. This suggests a loss of data in the dataset.

The study tested the pairwise word comparison approach to observe how the partially enhanced Jaro-Winkler algorithm adheres to the property of equality (symmetry property) by assigning each second word to relate to the first word at the end of each iteration. The final algorithm is referred to as the fully enhanced Jaro-Winkler algorithm. The experimental results are displayed in the Table 4.26.

Table 4.26: Results of fully enhanced Jaro-Winkler algorithm

Algorithm	Number of duplicate (match) words detected	Number of mismatch words detected
Fully enhanced Jaro-Winkler	72	136

The results displayed in Table 4.26 indicate that the number of instances in the matrix was 208 instead of the original 209 instances, while the number of duplicate (match) words detected was 72 and the number of mismatches detected was 136. This suggests that there was a high number of mismatch words detected by the algorithm. Table 4.27 shows the results on the rate of pairwise match and rate of pairwise mismatch.

Table 4.27: Results of fully enhanced Jaro-Winkler algorithm rate of match and mismatch

Algorithm	Rate of pairwise match	Rate of pairwise mismatch
Fully enhanced Jaro-Winkler	0.34615	0.65385

The results displayed in Table 4.27 show the rate of pairwise match as 0.34615 and the rate of pairwise mismatched as 0.65385. Thus, 34% of words were detected as duplicate while 65% of words were detected as mismatch.

The experimental results demonstrate that while the number of duplicate (match) words detected decreased from 101 in the partially enhanced Jaro-Winkler algorithm to 72 in the fully enhanced Jaro-Winkler algorithm, the number of pairwise mismatches words detected increased from 107 in the partially enhanced algorithm to 136 in the fully enhanced algorithm. Additionally, the rate of pairwise match of words decreased from 0.48558 in the partially enhanced algorithm to 0.34615 in the fully enhanced algorithm, while there was an increase in the rate of pairwise mismatch words from 0.51442 in the partially enhanced algorithm to 0.65385 in the fully enhanced algorithm.

Comparably, the number of duplicate (match) words detected in the enhanced Smith-Waterman algorithm is 74, while that of the enhanced Jaro-Winkler algorithm is 72. In terms of the rate of pairwise match, the number of duplicate (match) words detected in the enhanced Smith-Waterman algorithm is 0.35407 (that is, 35%), and in the enhanced Jaro-Winkler algorithm it is 0.34615 (that is, 34%). The experimental results suggest that the fully enhanced Smith-Waterman algorithm gives better results compared to the fully enhanced Jaro-Winkler algorithm.

4.3.3 Conclusion

The original Smith-Waterman algorithm and Jaro-Winkler algorithm in their current structure are limited in considering each word as token. The current structure is not suitable to be applied to duplicate detection of words when there are several rows of data to consider. The unique feature of the proposed algorithm is that, by applying the tokenization, transitive closure and property of equality (symmetry property) to a large dataset, duplicate words in a record can be identified, while mismatched words (misspelt words) can be detected and grouped together. The experimental results show that while the fully enhanced Smith-Waterman algorithm is accurate at pairwise word comparison without missing any, the fully enhanced Jaro-Winkler algorithm could not perform pairwise word comparison on the n^{th}-1 instance (row) if the total number of instances (row) is an odd number. This indicates that the fully enhanced Jaro-Winkler algorithm accurately performs a comparison if the number of instances (row) is even, but if the number of instances is odd, then it only computes the n^{th}-1 instance (row), so some duplicate records can be missed. This suggests that the fully enhanced Jaro-Winkler algorithm is challenged in respect of accuracy of comparing each word in a dataset, and in large datasets, it could miss some words, leading to the loss of information. In contrast, the fully enhanced Smith-Waterman algorithm performs accurate comparison without recourse to whether the number of instances is odd or even. Thus, the fully enhanced Smith-Waterman algorithm performs comparison without missing any words, and this suggests that in large datasets, the fully enhanced Smith-Waterman algorithm could perform better.

4.4 Applying the bio-inspired method of learning parameter onto Long Short-Term Memory (LSTM) network for feature selection in classification of high-dimensional bioinformatics datasets.

The approach to feature selection uses a bio-inspired method (KSA) and RNN with Long Short Term Memory network (LSTM) (Agbehadji *et al*. 2018b). The KSA was used to find optimum parameters from the entire dataset, which are then used to pre-train the RNN with LSTM network. In respect of the random nature of the proposed algorithm, each optimum was evaluated using the specified equation to test the fitness of each solution and the level accuracy. The following algorithmic structure was used to implement the feature selection process (Table 4.28).

Table 4.28: Proposed algorithmic structure for feature selection

```
Set parameters
Initialize population of $n$ kestrels and evaluate fitness of population.
```

```
Start iteration (loop until termination criterion is met)
    Generate new population using random encircling
    Evaluate fitness using imitation (Equation 3.12).
    Evaluate selected subset using the classification algorithm
    Update encircling position for each kestrel for all $i$=1 to $n$
    Find the optimal feature subset
End loop
```

4.4.1 Experimental setup

The proposed algorithmic structure was implemented in MATLAB 2018A. For the purpose of ensuring that best solution (in terms of optimized parameters) is selected as best parameter for training the RNN with LSTM network classifier (which is configured with 100 hidden layers), 100 epochs were performed as suggested by Batres-Estrada (2015), as it guarantees optimum results on classification accuracy. At each time step, the LSTM network is updated with the optimal parameter from KSA and the LSTM is reset. Batres-Estrada (2015) observes that choosing a small value as learning rate makes the interactions in weight space smooth, but at the cost of a longer learning rate. Similarly, choosing a large learning rate parameter makes the adjustment too large, which makes the network unstable (that is, the deep learning network). To avoid network instability, all neurons in the input to output layers on a network should learn at the same rate (that is, with smaller learning rate) (Batres-Estrada 2015). The use of a small but optimized learning rate/parameter was ensured by the use of meta-heuristic algorithms such as the KSA.

The optimized results from the meta-heuristic algorithms and the respective results on classification accuracy are the criteria to evaluate each meta-heuristic algorithm used in the experiment for classification of features. The optimized results from each meta-heuristic algorithm are considered as best solution if they have higher classification accuracy of features (Mafarja and Mirjalili 2018).

The initial parameters for each meta-heuristic algorithm are defined in Table 4.29. These were suggested by the creators of the algorithms as best parameters that guarantee optimal solutions.

Table 4.29: Algorithm and initial parameters

Algorithm	**Initial parameter**

KSA	pa=0.97; % Frequency of bobbing
	zmin=0.2; % perched parameter
	zmax=0.8; % flight parameter
	half-life=0.5; % half-life parameter
	dissimilarity=0.2 % dissimilarity parameter
	Similarity=0.8 % similarity parameter

PSO	w=1; % Inertia Weight
	c1=2.5; % Personal/cognitive Learning Coefficient
	c2=2.0; % Global/social Learning Coefficient

ACO	α=1; % Pheromone Exponential Weight
	ρ=0.05; % Evaporation Rate

BAT	α=0.9; % constant parameter
	γ=0.9; % constant parameter
	β=1; % random vector which is drawn from a uniform distribution [0, 1]
	A=1; % Loudness (constant or decreasing)
	r=1; % Pulse rate (constant or decreasing)

WSA-MP	v=1; % radius of the visual range
	pa=0.25; % escape possibility; how frequently an enemy appears
	Tol=1.0e-3; % tolerance
	α=0.2; % velocity factor (α) of wolf

To test the robustness of the proposed algorithm, nine benchmark datasets (that is, biological datasets from Arizona State University) were used. These datasets were chosen because they represent a standard benchmark dataset with continuous data for experimental research that are suitable for this research work. Table 4.30 shows the benchmark dataset and number of features in the original dataset.

Table 4.30: Benchmark datasets and number of features in dataset

	Dataset	**# of instances**	**# of classes**	**# of features in original dataset**

1.	Allaml	72	2	7129
2.	Carcinom	174	11	9182
3.	Gli_85	85	2	22,283
4.	Glioma	50	4	4434
5.	Lung	203	5	3312
6.	Prostate-GE	102	2	5966
7.	SMK_CAN_187	187	2	19,993
8.	Tox_171	171	4	5748
9.	CLL_SUB_111	111	3	11340

4.4.2 Experimental results

The minimum learning parameter from the original dataset and classification accuracy helped evaluate and compare the different meta-heuristic algorithms. One hundred iterations where performed by each algorithm to refine parameters for the LSTM network classifier in each dataset (that is, the biological dataset from Arizona State University). Table 4.31 shows the learning parameters in terms of optimum value of each meta-heuristic algorithm.

Table 4.31: Results obtained on optimum learning parameters of algorithms after running comparative meta-heuristic algorithms

Learning parameter	KSA	BAT	WSA-MP	ACO	PSO
Allaml	**4.0051e-07**	1.232e-07	1.7515e-07	3.3918e-07	1.9675e-06
Carcinom	**1.3557e-07**	1.0401e-07	3.0819e-05	8.7926e-04	0.5123
Gli_85	4.1011	0.032475	3.6925	**0.0053886**	2.2259
Glioma	**2.3177e-06**	3.0567e-05	1.9852e-05	9.9204e-04	0.3797
Lung	**5.1417e-06**	4.4197e-05	3.0857e-05	6.231e-04	0.3373
Prostate-GE	**1.6233e-07**	4.5504e-06	1.0398e-06	3.4663e-05	0.1178
SMK_CAN_187	0.015064	1.338e-05	**4.7188e-05**	2.7294e-05	2.5311
Tox_171	0.16712	**0.0002043**	0.086214	0.0023152	2.2443
CLL_SUB_111	0.82116	0.075597	0.76001	**0.011556**	9.6956

Table 4.31 shows the optimum learning parameters obtained after executing each meta-heuristic algorithm, with the best learning parameter for each meta-heuristic algorithm

highlighted in bold. It is observed from Table 4.31 that out of the nine datasets that were used, the KSA has the best learning parameter in five datasets. The learning parameters were fed into the LSTM network to determine the performance in terms of classification accuracy of each algorithm (that is, a way of knowing which algorithms outperform each other), and the results are shown in Table 4.32.

Table 4.32: Best results on accuracy of classification for each algorithm

Classification accuracy	KSA	BAT	WSA-MP	ACO	PSO
Allaml	0.5633	0.6060	**0.6130**	0.5847	0.4459
Carcinom	**0.7847**	0.7806	0.6908	0.7721	0.7282
Gli_85	0.2000	**0.4353**	0.2004	0.4231	0.3335
Glioma	0.7416	0.7548	0.5063	0.7484	**0.7941**
Lung	0.5754	0.5754	0.5754	0.5754	**0.7318**
Prostate-GE	0.6852	0.6718	0.6147	0.5444	**0.7223**
SMK_CAN_187	**0.6828**	0.6759	0.6585	0.6111	0.2090
Tox_171	**0.7945**	0.6925	0.7880	0.5889	0.2127
CLL_SUB_111	**0.7811**	0.4553	0.7664	0.4259	0.2000
Average	**0.6454**	**0.6275**	**0.6015**	**0.586**	**0.4864**

Table 4.32 shows the classification accuracy using the full dataset and the learning parameter from each algorithm. The classification accuracy for the Allaml dataset using KSA is 0.56 and 0.6130 using WSA-MP. It is observed that the algorithm with the best parameter is not the best choice in some datasets. For instance, the KSA has the best parameter of 1.6233e-07 on the Prostate-GE dataset, but produced a classification accuracy of 0.6852, while the Bat algorithm has a worst parameter of 0.1178, but produced a classification accuracy of 0.7223. Hence, the results (as shown in Table 4.32) suggest that a minimum learning parameter is not always a guarantee of accuracy, as it depends on the dataset, particularly the number of features. It can be observed that KSA provided the highest classification accuracy on four out of nine datasets. This classification accuracy in Table 4.32 demonstrates that the proposed approach explores and exploits search space efficiently, and finds the best results that produce higher classification

accuracy in many types of datasets. In order to select features, Mafarja and Mirjalili (2018) indicate that the higher the classification accuracy, the better the solution and, hence, the smaller the number of features in a subset. Table 4.33 shows the dimensions of features selected by each algorithm.

Table 4.33: Dimensions of features selected by each algorithm.

Feature selected	KSA	BAT	WSA-MP	ACO	PSO
Allaml	3113	2809	2759	2961	3950
Carcinom	1977	2015	2839	2093	2496
Gli_85	17826	12583	17817	12855	14852
Glioma	1146	1087	2189	1116	913
Lung	1406	1406	1406	1406	888
Prostate-GE	1878	1958	2299	2718	1657
SMK_CAN_187	6342	6480	6828	7775	15814
Tox_171	1181	1768	1219	2363	4525
CLL_SUB_111	2482	6177	2649	6510	9072

Table 4.33 shows the features that were selected from the respective datasets by each algorithm. It is observed that KSA selected a smaller number of features from four datasets, namely Carcinom, SMK_CAN_187, Tox_171 and CLL_SUB_111; PSO selected fewer features from three datasets, namely Glioma, Lung and Prostate-GE; and BAT and WSA-MP selected a smaller number of features from the Gli_85 and Allaml datasets respectively.

4.4.3 Statistical analysis of classification accuracy

A statistical test was conducted on the classification accuracy of each algorithm to identify the best algorithm. In order not to prejudice which algorithms outperformed each other, the means of all the algorithms were considered as equal for the statistical analysis. The reason for the statistical analysis of experimental results is to determine the significance of results on classification accuracy obtained from each optimizer (KSA, BAT, WSA-MP, ACO and PSO). In order to achieve this, a non-parametric statistical procedure was used as it does not make underlying assumptions about the distribution of parameters and underlying dataset for the evolutionary algorithm optimizers. In contrast, parametric statistical procedures make assumptions on parameters and distribution of datasets. A non-parametric statistical test was conducted to assess which of the algorithms have better performance in terms of the classification accuracy. In multiple-comparison situations, such as in this thesis, the Wilcoxon signed-rank test was applied to test how significant algorithms outperform each other in respect of detecting the differences in the mean and to find the probability of an error in suggesting that the medians of two algorithms are equivalent. This probability is called the *p*-value (Zar 1999). The advantage of the Wilcoxon test is that there is no need to make assumptions about the population used, since the test can guarantee about 95% (that is, 0.05 level of significance) of efficiency if the population is normally distributed. The following steps are applied in computing the Wilcoxon signed-rank test:

Step 1: Compute the difference *D* of paired samples in each algorithm. Any pairs with a difference of 0 are discarded.

Step 2: Find the absolute *D*.

Step 3: Compute the rank of signs (R_+ difference and R_- difference) from lowest to highest. The sum of ranks is expressed in Equation 4.12:

$$\sum R_+ + R_- = \frac{n(n+1)}{2} \qquad \textit{Equation 4.12}$$

where *n* is sample size.

Step 4: Compute the test statistic *T*. Thus, $T = min\{$R+, |R-|$\}$. The test statistic *T* is therefore the smallest value.

Step 5: Find the critical values based on the sample size *n*. If *T* is less than or equal to the critical value at a level of significance (that is, α=0.05), then a decision is made that algorithms are significantly different (García, Fernández, Benítez and Herrera 2007). In order to accomplish this, the Wilcoxon signed-rank table is consulted, using the critical value (α=0.05) and sample size *n* as parameters, to obtain the value within the table. If this value is less than the calculated value of the algorithmic comparison, this means that the algorithmic difference is significant.

In order to apply the Wilcoxon signed-rank test, an analysis was performed on classification accuracy, and the results are displayed in Table 4.34.

Table 4.34: Test statistics

Comparative algorithms	**Z**	**Asymp. Sig. (2-tailed)**
BAT – KSA	-0.420	0.674
WSA-MP – KSA	-1.680	0.093
ACO – KSA	-0.980	0.327
PSO – KSA	-1.007	0.314

The results on test statistics (p<0.05) shown in Table 4.34 show that the differences between the medians are not statistically significantly different in all the comparative algorithms. For instance, there are no statistically significant differences between the KSA and BAT at the level of significance of 0.05, because 0.674>0.05. Similarly, KSA compared to WSA-MP, ACO and PSO all have their p-values greater than the level of significance. This indicates that there are no statistically significant differences between KSA compared to WSA-MP, ACO, PSO and BAT.

4.4.4 Conclusion

The KSA has its own advantages in feature selection in classification. Compared to meta-heuristic algorithms, classification accuracy of KSA is comparable to ACO, BAT, WSA-MP and PSO. This suggests that the initial parameters that were chosen in KSA guarantee good solutions that are comparable to other meta-heuristic search methods on feature selection. The future work for KSA is to develop new versions of KSA with modifications and enhancements of code for feature selection in classification.

4.5 Summary

In this chapter, the bio-inspired computation model was applied to the current challenge of finding missing values at random. The model was implemented and compared to related bio-inspired algorithms, as indicated in the methodological framework for this study. The stages that were implemented were missing value estimation, duplicate text detection and feature selection. The programming codes were written to help transform the algorithmic structure in order to test the dataset.

The experimental results and statistical analysis conducted using the Wilcoxon signed-rank test show that the proposed KSA outperformed the Bat and Firefly algorithms in finding the optimal value in a synthetic dataset that has different dimensions with multiples of missing values. Further statistical analysis was conducted to find out whether the built-in function time calls and major function time calls might be equivalent or not. The Friedman test was conducted and the null hypothesis was that the median of the distribution is equivalent at a confidence interval value of 0.05. The results suggested that there is a significant difference in the quality of estimation among the algorithms, thus the algorithms are not the same. The results also showed that the proposed algorithm should be enhanced by further fine-tuning of its parameter.

The experimental results on the enhanced Smith-Waterman and Jaro-Winkler algorithms for duplicate detection of text in large datasets (that is, a health-related heart disease dataset) indicated that the enhanced Smith-Waterman algorithm is more accurate at detecting duplicate words than the enhanced Jaro-Winkler algorithm. This signifies that when both algorithms are applied on the same datasets, the Smith-Waterman algorithm is able to avoid any information loss, while the Jaro-Winkler algorithm results in an information loss. Thus, when there are large volumes of data to be analyzed for duplicate text, the Smith-Waterman algorithm will perform best without data loss because of the transitive closure and reflexive properties that were included as an enhancement.

Although deep learning methods have been applied to selection of features in the classification problem, current approaches to learning of a parameter for the classification process can either grow out of bound or shrink at each time step. This parameter resizing might result in inaccurate classification of features. To address this challenge, the study proposes an approach to learning parameters for the classification problem based on the behavior of kestrels, that is, random encircling from a hovering position and learning by imitating well-adapted behavior of individual kestrels to adjust learning rate. The proposed bio-inspired approach is integrated with a deep learning method (that is, recurrent neural network with long short term memory network). A benchmark dataset (with continuous data attributes) was chosen to test the proposed search algorithm. The results showed that KSA is comparable to BAT, ACO and PSO, as test statistics in Table 4.34 (that is, the Wilcoxon signed rank test) show no statistically significant differences between the mean of classification accuracy at the level of significance of 0.05. Meanwhile, compared with WSA-MP, KSA shows a statistically significant difference in the mean of classification accuracy.

The next chapter discusses the implementation and empirical testing of the proposed computational model of data mining on frequently changed patterns/items with time and numeric value dimensions. Test data that is characterized as having frequently changed items (that is, stock market data) is used to validate the proposed computational model.

CHAPTER 5: DEVELOPING, TESTING AND EVALUATING DATA MINING BASED ON KESTREL-BASED SEARCH ALGORITHM

5.1 Introduction

The chapter addresses the question of how to mathematically formulate and implement an associative data mining rule to extract frequently changed items with both numeric and time dimensions. The objective is to select interesting patterns/rules taking into consideration the time-closeness between the frequently changed items. In this chapter, a data mining model is provided to mine rules and disclose interesting patterns on frequently changed data. This chapter further shows how to apply the proposed model to tackle frequently changed items in stock market data. Stock market data has the characteristics of large scale and fast update of stock numerical values. The characteristics of stock data are similar to big data characteristics, such as having volume, velocity, value and variety (Longbottom and Bamforth 2013).

As discussed in Chapter One and Chapter Two, when data becomes very large (volume characteristic), it is possible that current approaches to mining interesting patterns lose their value (value characteristic) in terms of having to determine in a timely (relating to speed) manner the usefulness of an action. This timeliness in determination is important because large rules are discovered that might not show interesting patterns. The massive amount of data can be analyzed and interpreted to show interesting patterns using different techniques on algorithms, tools and models that are different from existing search methods of analyzing and interpreting data. This was the motivation to propose an approach (in the form of an algorithm called KSA) based on how animals behave, and basic mathematical expression were formulated from their behavior to develop an algorithm to analyze data. During the analysis, the formulated mathematical expression of KSA was integrated with a mathematical model on closeness preference (Railean *et al*. 2013), as discussed in section 3.3.2. The basis for the integration is to enable the proposed model to take into consideration both numeric and time dimensions for analysis of frequently changed items. The advantage of the closeness preference model is that it considers time interval and slope of preference.

The proposed KSA model and closeness preference interestingness measure helped extract association rule mining of interesting patterns for interpretation by the researcher. The algorithm from the computational model was compared with other meta-heuristic algorithms, namely the PSO, ACO, BAT and WSA-MP algorithms. The comparative algorithms helped validate the tabulated results from the proposed model.

The general outline in section 3.5 was followed by modeling an algorithm for data mining, as indicated in the methodological framework in Table 3.1.

5.2 Association rule

A rule is defined as a conditional statement that specifies an action for a certain set of conditions (Iglesia and Reynolds 2005). An association rule is an implication of the form *K*→*P*, where precondition *K* is referred to as the antecedent, the action *P* is called the consequent, and both *K* and *P* are itemsets with numeric value. The discovered rule is expressed in an "If … Then …" statement. Thus, If *K*, then *P*.

There are two groups of rules: simple rules and complex rules (Railean *et al.* 2013). The *s*imple rules are of the form $Vi \rightarrow Vn$, and complex rules are of the form $V_1V_2...V_{n-1} \rightarrow V_n$ (Railean *et al. 2013)*. In this instance, all rules of the simple form $Vi \rightarrow Vn$ were combined between all *Vi* itemsets. For example, having the rules $A \rightarrow Y$, $B \rightarrow Y$ and $C \rightarrow Y$, a complex rule is derived as $AB \rightarrow Y$, $AC \rightarrow Y$, $BC \rightarrow Y$, $ABC \rightarrow Y$.

The fitness function is used to evaluate the importance of each item and select interesting rules/patterns. The fitness function is expressed in Equation 5.1:

$$MCP_{sc\,pattern} = \phi * \frac{nr_{p\,o}}{nr_{p\,o}+nr_{i\,p}} * \frac{nr_{p\,o}}{|\,DB\,|} * f(nr, \Delta t_{average}) \qquad \textit{Equation 5.1}$$

where ϕ represents a control parameter between 0 and 1; $nr_{\text{pattern occurrences}}$ ($nr_{p\ o}$) represents frequently changed items; $nr_{\text{not freq. patterns}}$ ($nr_{n\ p}$) represents patterns that are not frequently changed; |DB| is the total size of the dataset; $f(nr, \Delta t_{average})$ is a function that is defined to take into consideration the size of the pattern *nr* and the medium time interval $\Delta t_{average}$ in a pattern.

Based on the mathematical formulation in Chapter Three (section 3.2), the proposed algorithm process for association rule mining is as follows:

1) User sets time preference and stopping criteria
2) Random initialization of each kestrel and of time dimension
3) Perform search for best encircling position
4) Compute the velocity
5) Compute the slope of glide
6) Evaluate fitness function (Equation 5.1)
7) Update encircling position
8) Check if the termination condition is met (if not, go back to Step 2)
9) Output best minimum as the support threshold
10) Mining of association rules
11) Output the frequently changed items results

The proposed computational algorithm is summarized in the form of pseudo-code and shown in Figure 5.1.

```
Random initialization of each kestrel and
Random initial time dimension;
set user time preferences;
set maximum confidence
WHILE ($t$ < stopping criteria not met)
//Finds minimum support threshold
  FOR $i$=1to $k$ //for each kestrel in the population
```

```
        Compute the encircling position
        Compute velocity, $k$=1,.., $n$
        Compute the slope of glide
        Evaluate fitness function //help filter rules
        Replace $f_2(MCP_{sc\ pattern})$ with $f_1(MCP_{sc\ pattern})$
        Update encircling position
  END IF
END FOR
END WHILE
FOR each item $n$
    Find difference of time dimension
    IF difference of time <=user preferences
         Compute the difference of numeric items
         Count the numeric items
         FOR each numeric item in a row
             IF count of numeric items equal to zero
                 Count not frequently changed numeric items
            ELSE IF count of numeric items not equal to zero
                 Count frequently changed numeric items
            END IF
        END FOR
         Compute the MCPs
         IF MCPs > minSup
             Evaluate the MCP sc pattern (referred to as MCP conf)
              IF $f_1(MCPsc\ pattern)$ > minConf
                 Extract rules on frequently changed numeric items
                END IF
        END IF
    END IF
  END FOR
            Output rules of frequently changed items
```

Figure 5.1: Proposed computational model for association rule mining

Based on this proposed computational process, association rules were generated within the user-specified time preferences. The difference with this computational process compared to other existing computations is the addition of $MCP_{sc\,pattern}$, as expressed in Equation 5.1. The time complexity (expressed in Equation 5.2 and Equation 5.3*)* of finding frequently changed itemsets was calculated using the expression in the calculation formulated by Minaei-Bidgoli, Barmaki and Nasiri (2013). Thus, the time complexity R is expressed as follows:

$$R = O(frequent\ itemset) + O(rule\ generation) \qquad \textit{Equation 5.2}$$

$$\begin{aligned}
&= O(N * d * 2^{d}) + O\left(\sum_{k=1}^{d-1}\left[\binom{d}{k} \times \sum_{j=1}^{d-k}\binom{d-k}{j}\right]\right) \\
&= O(N * d * 2^{d}) + O(3^{d} - 2^{d+1} + 1) \\
&= O(N * d * 2^{d}) + O(3^{d}) \\
&= O(2^{d+1})
\end{aligned} \qquad \textit{Equation 5.3}$$

where d is the number of items/attributes being considered, k is the number of iterations, and N is the number of records in the dataset.

5.3 Experimental setup

The proposed algorithm was implemented in MATLAB. For the purpose of ensuring that good parameters are selected, several parameters (in terms of numeric parameters) were experimented. Table 5.1 presents the parameters and the corresponding parameters on the comparative algorithms (PSO, ACO, BAT and WSA-MP), as stated in the methodological framework (see Table 3.1).

Table 5.1: Algorithm and initial parameters

Algorithm	Initial parameters

KSA	pa=0.97; % Frequency of bobbing
	zmin=0.2; % perched parameter
	zmax=0.8; % flight parameter
	half-life=0.5; % half-life parameter

PSO	w=1; % Inertia Weight
	c1=2.5; % Personal/cognitive Learning Coefficient
	c2=2.0; % Global/social Learning Coefficient

ACO	α=1; % Pheromone Exponential Weight
	ρ=0.05; % Evaporation Rate

BAT	α=0.9; % constant parameter
	γ=0.9; % constant parameter
	β=1; % random vector which is drawn from a uniform distribution [0, 1]
	A=1; % Loudness (constant or decreasing)
	r=1; % Pulse rate (constant or decreasing)

WSA-MP	v=1; % radius of the visual range
	pa=0.25; % escape possibility; how frequently an enemy appears
	Tol=1.0e-3; % tolerance
	Velo=0.8; % velocity factor (α) of wolf

Table 5.1 shows the initial parameters of the algorithms. In respect of PSO, the parameters $c1$ and $c2 \in [0, 2]$ are called acceleration coefficients, namely the cognitive and social parameter, respectively (Alatas and Akin 2008). These parameters control how far a particle will move in a single iteration. As default values, Kennedy and Eberhart (1995) propose $c1 = c2 = 2$. Recent work has suggested that it might be better to choose a larger cognitive parameter $c1$ (Alatas and Akin 2008) to allow adequate learning. On this basis, the personal/cognitive learning coefficient parameter was changed from 1.5 to 2.5. A larger inertia weight w achieves the global exploration, and a smaller inertia weight tends to facilitate the local exploration to fine-tune the current search area (Shi and Eberhart 1998). Therefore, the inertia weight w is critical for the PSO's convergence behavior. A suitable value for the inertia weight usually provides balance between global and local exploration abilities and consequently results in a better optimum solution. The initial weight was set to 1, as suggested by Kennedy and Eberhart (1995).

The parameter used in WSA-MP were set as indicated by Yamany, Emary and Hassanien (2014) to guarantee best performance results. Also, the initial parameters on ACO were experimentally tested by the authors of the algorithms as the parameters that guarantee best performance. With reference to Yang (2010), the parameters α and γ in the Bat algorithm were set to 0.9, and β was set to 1.

In selecting a dataset to test the proposed model, there were two important considerations. First, the dataset must have a characteristic of frequently changed numeric attributes. Second, each numeric attribute dataset could have frequently changed time dimensions. In this context, each row element that represents an attribute has several multiples of numeric attributes and the respective different time dimensions. Based on the two characteristics, the stock market data was chosen from the classic "UCI machine learning repository" (Lichman 2013) as the appropriate dataset for testing the proposed model. The test dataset is a 10x3000 matrix, where 10 represents elements of the row matrix (that is, the stock items) and 3000 represents the numeric attributes of the column matrix. All simple and complex rules are applied on the test dataset. In the testing phase, the rules are considered to be valid if all of their itemsets occur in the time-window of length ωt (Railean *et al*. 2013).

During the experimental setup, initial parameters for each algorithm were set as presented in Table 5.1. The stock market dataset was loaded and transposed, and each algorithm was executed independently on the dataset. The transposed dataset helps transform the original column attributes into row attributes with corresponding numeric values. In order to refine output results to obtain the minimum value, 100 iterations were performed. The minimum output value was used as the initial minimum support threshold to mine rules on frequently changed stock items. Thus, the stock item is considered frequently changed if the support is above the minimum support threshold. Although different user time preferences (measured in seconds) and MCP confidence (that is, MCPconf) could be set, a user time preference was set as 0.7 seconds, and a minimal MCP confidence value was set as 0.9. Additionally, all time intervals are measure in seconds in this chapter.

5.4 Experimental results

The proposed model generated a minimum support threshold value of 1.4823e-14, and based on the minimum support threshold, the following association rules were generated and tabulated:

Table 5.2: Association rules for stock market dataset using KSA

Rule #	MCPconf value	Time interval in seconds	# of rules extracted	Example of rules extracted
1	90.3246	0.15054	36	X1,X2,X3,X4,X5,X6,X8→ X7,X10
2	99.1485	0.50753	1	X4→X1,X2,X3,X5,X6,X7, X9,X10
3	94.5423	0.50753	26	X1,X4→X2,X3,X5,X7,X8, X9X10
4	95.4177	0.50753	7	X1,X8→X2,X3,X4,X5,X7, X9, X10
5	96.2931	0.50753	3	X2,X4→X1,X3,X5,X7,X8, X9, X10
6	93.5961	0.5067	1	X2→X1,X3,X4,X9
7	95.4094	0.5067	2	X1, X2→X3,X5,X8
8	94.5341	0.5067	8	X1,X5→X2,X3,X8
9	95.8549	0.47927	35	X1,X2 ,X4→X3,X5,X6,X7
10	91.6942	0.22924	21	X1,X2,X3,X4, X6→X5,X8
11	98.6783	0.10964	2	X9→X1,X2,X3,X4,X5,X6
12	98.6607	0.10962	2	X1→X2,X9
13	98.6607	0.10962	2	X9→X1,X2
14	99.0389	0.058258	1	X4→X1,X3
15	94.4377	0.0087442	2	X1,X3→X10
16	92.7195	0.18544	28	X1,X2,X3,X4,X5,X6→X7,X9
Mean	**95.5632**	**0.31191**		

Table 5.2 shows the results obtained from the proposed model. Frequently changed stock items are anonymously represented by $x_i...x_n$, where n represents the number of stock items. A total number of 177 complex rules were extracted, which are of the form $X_1X_2...X_{n-1} \rightarrow X_n$ (Railean *et al*. 2013) and also of the form $X_n \rightarrow X_1X_2...X_{n-1}$. There were no simple rules extracted from the stock dataset.

For instance, at rule 1, with a user time interval of 0.1505 seconds and am MCPconf of 90.3246%, a total of 36 different complex rules were extracted. Although all the complex rules were not shown, the example presented in Table 5.2 is an instance of the rule extracted, that is, $X1,X2,X3,X4,X5,X6,X8 \rightarrow X7,X10$ where $X1,X2,X3,X4,X5,X6,X8$ represents the antecedent part of the rule, and $X7,X10$ represents the consequent part of the rule. Thus, if $X1,X2,X3,X4,X5,X6,X8$ are frequently changed stock items, then this leads to $X7,X10$ stock items that are also frequently changed. The MCPconf of 90.3246% helps select a high percentage of frequently changed rules.

From Table 5.2, it is observed that the mean for the MCPconf value is 95.5632%, and the mean of time is 0.31191 seconds. This indicates that the KSA extracted an average of 95.5% of rules within a 0.31-second time interval.

Table 5.3 shows the experimental results obtained using the ACO, using the same user time preference of 0.7 seconds and MCPconf value to 0.9. The minimum support value obtained was 1.8168e-13. Based on the minimum support, the following association rules on frequently changed items were extracted, as shown in Table 5.3.

Table 5.3: Association rules for stock market dataset using ACO

Rule #	MCPconf value	Time interval in seconds	# of rules extracted	Example of rules extracted
1	92.9081	0.50565	25	X1,X2,X3,X4,X5,X6→X7,X9
2	93.4948	0.49641	1	X2→X1,X3,X4, X9
3	95.3061	0.49641	2	X1,X2→X3,X5,X8
4	94.4317	0.49641	8	X1,X3→X2,X5,X8
5	95.3061	0.49641	2	X1,X8→X2,X3,X5
6	98.6503	0.49602	2	X1→X2, X9
7	99.0285	0.49602	1	X4→X1,X3
8	94.4278	0.49602	2	X1,X3→X10
9	95.3022	0.49602	1	X1,X10→X3
10	94.4278	0.49602	1	X3,X10→X1
11	95.8196	0.49521	35	X1,X2,X3→X4,X5,X6 ,X7
12	91.6604	0.49521	21	X1,X2,X3,X4,X5→X6,X8
13	98.642	0.49521	2	X1→X2, X3,X4,X5,X6, X9
Mean	**95.3389**	**0.49669**		

A total of 100 complex rules were extracted from the stock dataset. For instance, at rule 11 with an MCPconf of 95.8196%, thirty-five complex rules were extracted. An example of such a rule is *X1,X2,X3→X4,X5,X6,X7,* where *X1,X2,X3* is the antecedent part of the rule, while *X4,X5,X6,X7* represents the consequent part of the rule. Thus, *X1,X2,X3* are frequently changed stock items in a sequence that lead to *X4,X5,X6,X7* stock items of frequently changed nature. Based on the output results, it was observed that no simple rules on frequently changed items were extracted.

From Table 5.3, it is observed that the mean for the MCPconf value is 95.3389%, and the mean of time is 0.49669 seconds. This indicates that the ACO algorithm extracted an average of 95.3% of rules within a 0.49-second time interval.

Table 5.4 shows the experimental results obtained using the PSO. The same user time preference of 0.7 seconds and MCPconf value of 0.9 were used. The minimum support value obtained was 0.048569. Based on the minimum support threshold, the following association rules on frequently changed items were extracted, as shown in Table 5.4.

Table 5.4: Association rules for stock market dataset using PSO

Rule #	MCPconf value	Time interval in seconds	# of rules extracted	Example of rules extracted
1	99.2603	0.50355	1	X7→X1,X2,X3,X4,X5,X6,X8
2	90.5613	0.50204	1	X4→ X1
3	94.9438	0.49673	1	X2→X1,X3
4	90.5107	0.49673	1	X4→X1,X2
5	90.5103	0.49668	1	X4→X1,X2,X3,X5,X6
Mean	**93.1573**	**0.49915**		

As shown in Table 5.4, a total of five rules, both simple and complex, were extracted from the stock dataset as having interesting patterns upon which a user can take an action. For instance, a user may decide to invest in stock item *X4* that certainly leads to stock item *X1*. From Table 5.4, it is observed that one simple rule was extracted, while four complex rules were extracted during the iteration. For instance, at rule 2 with MCPconf of 90.5613%, a simple rule was extracted at time 0.5020 seconds as *X4→X1*, where *X4* is the antecedent part of the rule, and *X1* represents the consequent part of the rule. Thus, a frequently changed stock *X4* leads to *X1* stock items. At rule 4, a complex rule was extracted as *X4→X1,X2*, thus a frequently changed stock *X4* leads to a sequence of *X1,X2* frequently changed stock items.

From Table 5.4, it is observed that the mean for the MCPconf value is 93.1573% and the mean of time is 0.49915 seconds. This indicates that the PSO algorithm extracted an average of 93.1% of rules within a 0.49-second time interval.

Table 5.5 shows the experimental results obtained using the Bat algorithm. The *same* user time preference of 0.7 seconds and MCPconf value of 0.9 were used. The minimum support value obtained was 3.0109e-07. Based on the minimum support, the following association rules on frequently changed items were extracted, as shown in Table 5.5.

Table 5.5: Association rules for stock market dataset using BAT

Rule #	MCPconf value	Time interval in seconds	# of rules extracted	Example of rules extracted
1	90.5716	0.50311	1	X4→X1
2	99.2277	0.50042	1	X7→X1,X2,X3,X4,X5,X6,X8
3	94.9603	0.49837	1	X2→X1,X3
4	90.5264	0.49837	1	X4→X1,X2
5	90.526	0.49833	1	X4→X1,X2,X3,X5,X6
Mean	**93.1624**	**0.49972**		

As observed from Table 5.5, a total of five rules, both simple and complex, were extracted from the stock dataset as having interesting patterns upon which a user can take an action within a time frame. It is observed that one simple rule was extracted, while four complex rules were extracted. For instance, at rule 2 with an MCPconf of 99.2277%, a complex rule was extracted at time 0.50042 seconds as *X7→X1,X2,X3,X4,X5,X6 X8,* where *X7* is the antecedent part of the rule, and *X1,X2,X3,X4,X5,X6,X8* represents the consequent part of the rule. Thus, a frequently changed stock *X7* leads to *X1,X2,X3,X4,X5,X6,X8* stock items in sequence. At rule 1, a simple rule was extracted as *X4→X1*, thus a frequently changed stock *X4* leads to frequently changed stock item *X1*.

From Table 5.5, it is observed that the mean for the MCPconf value is 93.1624% and the mean of time is 0.49972 seconds. This indicates that the Bat algorithm extracted an average of 93.1% of rules within a 0.49-second time interval.

The experimental results obtained using WSA-MP are shown in Table 5.6. A user time preference of 0.7 seconds and MCPconf value of 0.9 were used. The minimum support value obtained was 6.7486e-07. Based on the minimum support, the following association rules on frequently changed items were extracted, as presented in Table 5.6.

Table 5.6: Association rules for stock market dataset using WSA-MP

Rule #	MCPconf value	Time interval in seconds	# of rules extracted	Example of rules extracted
1	93.5951	0.5066	1	X2→X1,X3,X4,X9
2	95.4084	0.5066	2	X1,X2→X3,X5,X8
3	94.5331	0.5066	8	X1,X3→X2,X5,X8
4	95.925	0.50566	35	X1,X2,X3→X4,X5,X6,X7
5	91.7612	0.50566	21	X1,X2,X3,X4,X5→X6,X8
6	98.7505	0.50566	2	X1→X2,X3,X4,X5,X6,X9
7	92.8651	0.50124	25	X1,X2,X3,X4,X5,X6→X7,X9
8	98.6756	0.49844	2	X1→X2,X9
9	99.0538	0.49844	1	X4→X1,X3
10	95.3265	0.49844	1	X1→X3,X10
11	94.452	0.49844	2	X3,X10→X1
Mean	95.486	0.50289		

From Table 5.6, a total of 100 complex rules were extracted from the stock dataset. However, no simple rule was extracted. For instance, rule 5 has an MCPconf value of 95.925% with thirty-five complex rules that were extracted from the dataset. An example of such a rule is *X1,X2,X3→X4,X5,X6,X7*, where *X1,X2,X3* is the antecedent part of the rule while *X4,X5,X6,X7* represents the consequent part of the rule. Thus, *X1,X2,X3*, which frequently changes in sequence, leads to *X4,X5,X6,X7* frequently changed stock items.

From Table 5.6, it is observed that the mean for the MCPconf value is 95.486% and the mean of time is 0.50289 seconds. This indicates that the WSA-MP algorithm extracted an average of 95.4% of rules within a 0.50-second time interval.

Table 5.7 shows the algorithms and summary of the number of rules, both complex and simple, extracted during the experiment.

Table 5.7: Algorithms and number of rules extracted

Algorithm	Best minimum value	#of complex rules extracted	# of simple rules extracted
KSA	1.4823e-14	177	0
ACO	1.8168e-13	100	0
PSO	0.048569	4	1
BAT	3.0109e-07	4	1
WSA-MP	6.7486e-07	100	0

Table 5.7 shows the number of complex and simple rules extracted after each iteration during the experiment. KSA, ACO and WSA-MP extracted complex rules of the form $V_1V_2...V_{n-1} \rightarrow V_n$ (Railean, Lenca, Moga and Borda 2013), and no simple rules of the form $Vi \rightarrow Vn$ were extracted from the ten stock items in the original dataset. PSO and BAT extracted both simple rules of the form $Vi \rightarrow Vn$ and complex rules of the form $V_1V_2...V_{n-1} \rightarrow V_n$ (Railean *et al*. 2013) from the ten stock items. Also, KSA has a total of 177 rules, both ACO and WSA-MP have 100 rules, and both PSO and BAT have 5 rules. It is observed that KSA has the highest number of rules extracted, which could be attributed to its best minimum value (minimum support threshold). Based on Table 5.7, it is observed that PSO has the highest minimum value (0.048569), while KSA has the lowest minimum value (1.48E-14). The significance of the minimum support threshold is that it helps with mining close interesting patterns. Thus, it could be inferred that when the minimum support threshold is smaller, it is possible to extract many complex rules. The nature of each algorithm (in terms of the behavior of the algorithm, as explained in Chapter Two, and parameters) could have contributed to the best minimum values that were obtained. The proposed algorithm is better in some aspects because it improves on the weaknesses of existing bio-inspired algorithms (see Appendix 2) so as to help improve on the performance of big data algorithms. In formulating this algorithm, the researcher improved on the quality of machine learning algorithms, which was identified by Tsai *et al*. (2015) as an issue of big data analysis platforms.

5.5 Conclusion

The proposed KSA was used for association rule mining of frequently changed items with numeric attributes and a time dimension for stock market data. The proposed search algorithm (that is KSA) has the best minimum value (minimum support threshold) compared with the comparative algorithms (PSO, ACO, BAT, WSA-MP). The best minimum value obtained with KSA is attributed to the behavior and characteristics exhibited by kestrels, which the comparative algorithms could not exhibit. In terms of the number of rules discovered, the proposed KSA algorithm disclosed a high number of complex rules without any simple rules at an MCPconf value of 0.9 and time window of 0.7 seconds. Although the proposed model (KSA and closeness preference) generated complex rules, a limited amount of time was used, and this makes the proposed model the preferred algorithm for frequently changed items, as evident from the experimental results.

In addition, the experimental results showed that KSA had the best minimum value of 1.48E-14, and an average of 95.56% of rules were extracted at a mean time interval of 0.3 seconds. In comparison, ACO had a minimum value of 1.82E-13 and extracted an average of 95.33% rules at a mean time interval of 0.4 seconds; PSO had a minimum value of 0.048569 with an average of 93.1573% rules extracted at a mean time interval of 0.4 seconds; BAT had a minimum value of 3.01E-07, and an average of 93.1624% rules were extracted at a mean time interval of 0.4; and WSA-MP had a minimum value of 6.75E-07 and extracted on average 95.486% of rules at a mean time interval of 0.5 seconds. Based on the mean MCPconf value and mean time interval, KSA analyzed data on frequently changed items and discovered a high number of patterns in the shortest possible time interval. What makes the proposed KSA different from other bio-inspired search-based methods is the use of random encircling and close user preferences model for mining frequently changed items. Based on the experimental results, it could be concluded that KSA has the best minimum value of 1.4823e-14, in which 95.56% of rules were extracted at a mean of time of 0.31 seconds, which is the best among the comparative algorithms.

5.6 Summary

In summary, the chapter discussed the experimental setup suitable to implement the proposed computational model (based on KSA and closeness preference) on frequently changed items. The chapter discussed how the proposed computation model, which was mathematically formulated and expressed as a search algorithm, was implemented and tested. The algorithmic structure was implemented as a search algorithm that helped find frequently changed rules.

The next chapter discusses how to implement the computational model for visualization of data via dung beetle behavior of moving dung as a ball rolled from one location to another. The points of movement of the balls are referred to as data points, which are then plotted on a data grid.

CHAPTER 6: DEVELOPING, TESTING AND EVALUATING DATA VISUALIZATION BASED ON DUNG BEETLE ALGORITHM

6.1 Introduction

This chapter is the third phase of the proposed model, which seeks to visualize frequently changed data from the big data environment. The big data environment is characterized as having large datasets that are dynamic and can change within a second. However, when large amounts of information from datasets are available, it makes it difficult for users to visualize interesting patterns. Therefore, modern approaches to visualization could assist users by reducing the effort needed, considering the different user preferences in respect of time that may be required (Bikakis 2018). As time taken to view data is significant, there is a motivation to find new ways to reduce computational time during data visualization. The present approach is inspired by the behavior of animals, particularly dung beetles. The significance of a bio-inspired behavior, such as dung beetle behavior, for big data visualization is the ability to navigate and perform path integration with minimal computational power. The dung beetle behavior, when expressed as an algorithm, can find the best possible approach to visualize discrete data using minimal computational power, which is suitable when data coming from different sources have to be visualized quickly with less computational time.

Basic mathematical formulations were expressed and translated into algorithmic structure to depict the behavior of dung beetles and were then applied to help visualize frequently changed data in two-dimensional view to aid understanding. The general outline in section 3.5 was followed to model an algorithm for data visualization in a two-dimensional graph, as indicated in phase three of the methodological framework in Table 3.1.

6.2 Data visualization

Data visualization presents data in pictorial or graphical format and enables the display of interesting patterns for decision-making activities (Bikakis 2018). Usually, data a visualization tool (that is, software or an algorithm) is used to generate the graphical format. The basic mathematical expression was used to depict the dung beetle behavior and display interesting patterns in a graphical format. In creating the visual pattern, the self-adapting basic rules that were formulated to depict the dynamic behavior of dung beetles were applied to find an optimal solution to create a visual pattern of data points on a grid. The algorithm on the basic rules formulation is expressed as follows:

```
Objective function f(x), $x=(x_1,x_2,..x_d)^T$
Initialization of parameters;
Population of dung beetle $x_i(i=1,2,..,k)$;
Choose a random “real Home”
WHILE ( t < stopping criteria not met)
  FOR i=1: k //for each dung beetle in the population
      Roll ball
      Perform a dance
      Integrate path
      Evaluate position within external reference point
      Compare positions to find minimum
        IF position1 <position2
         Swap position1 with position2
        END IF
   END FOR
Update external reference point
Check stopping criteria
END WHILE
```

6.3 Evaluation of visualization technique

In line with Dull and Tegarden's (1999) and Risden and Czerwinski's (2000) approach to measuring effectiveness of visualization, an evaluation technique was applied in terms of computational time to complete a visualization task, and the quality of the task's optimal value was considered in terms of the optimal solution from the proposed bio-inspired visualization algorithm and the comparative visualization algorithms, namely the bee algorithm and ACO for data visualization. As explained in the literature review, bees are able to perform a dance in order to make a decision to choose their food source and revisit their food source for further exploration (Karaboga 2005). This self-organizing behavior of bees motivated the use of the bee algorithm for data visualization in this aspect of the thesis. Additionally, ACO was adopted as visualization method because of its ability to communicate the presence of a food source using its pheromone substance (Stützle and Dorigo 2002). This pheromone substance enables ants to converge and exploit the food source. Ants are also well adapted to reinforce the pheromone trail so that other ants can follow it. The unique behaviors of ants and bees were the reason for choosing the ant and bee algorithms as comparative algorithms for data visualization, instead of PSO, Firefly and WSA-MP.

6.4 Experimental setup

The algorithm for the proposed DBA was implemented in MATLAB and tested on a stock market dataset. The basic parameters for the DBA is defined as follows: error ε is 0.05, and β_m represents motion cues and is set to 0.2. The reason for the error parameter of 0.05 is to allow 95% accuracy in selecting the best path and a 5% chance of choosing an incorrect path that may lead to an "imaginary home". The reason for the 0.2 motion cue (or 20%) is the fact that other factors (e.g. hills and other impediments that can lead dung beetles to getting stuck in one place) in the environment (that accounts for 80%) may obscure the view of dung beetles when moving dung from one location to another.

6.5 Experimental results

The experiment was conducted in two stages: Firstly, association rules generated from Chapter 5 on stock market data were used with an initial modified closeness preference of support-confidence (MCPconf) (also referred to as modified closeness preference confidence) value of 0.9 and user time preference of 0.7. The reason for using a value of 0.9 is to extract high confidence association rules between 90% and 100% and within a time interval of 0 to 0.7 seconds. Secondly, DBA was applied to create graphical displays of results on MCPconf values. The results of association rules from Chapter 5 were viewed using the proposed DBA for data visualization and compared with data visualization algorithms such as the bee algorithm and ACO for data visualization. The results are presented in three parts: Firstly, the use of DBA to visualize data mining results from bio-inspired algorithms such as KSA, ACO, PSO, BAT and WSA-MP; secondly, the use of the bee algorithm to visualize data mining results from KSA, ACO, PSO, BAT and WSA-MP; and thirdly, the use of ACO for data visualization to view data mining results from KSA, ACO, PSO, BAT and WSA-MP. The results on visualization are tabulated and presented using graphs (where the *x*-axis represents the iteration and the *y*-axis represents either best computational cost or MCPconf value). Where appropriate, a distinction is made on what the *y*-axis represents due to different graphs that were used.

6.5.1 Visualization using DBA

The first part involved the use of DBA for visualization of association rule mining results from KSA, ACO, PSO, BAT and WSA-MP. Table 6.1 shows the MCPconf values that were generated from the association rules as follows:

Table 6.1: MCPconf values from KSA

Rule #	MCPconf value (%)
1	90.3246
2	99.1485
3	94.5423
4	95.4177

5	96.2931
6	93.5961
7	95.4094
8	94.5341
9	95.8549
10	91.6942
11	98.6783
12	98.6607
13	98.6607
14	99.0389
15	94.4377
16	92.7195

It is observed from Table 6.1 that sixteen association rules were generated within the MCPconf value of 0.9. Based on Table 6.1, the nature of the graphical display using DBA is as follows (where the *y*–axis represents the best computational cost and the *x*-axis represents the iteration):

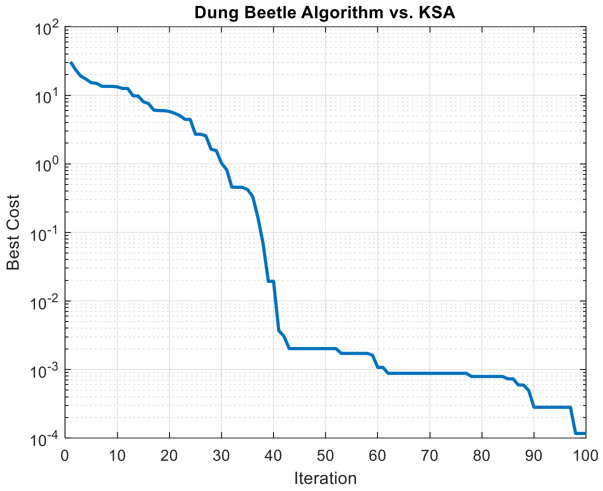


Figure 6.1: Dung beetle display of best cost on path traversed by KSA

Figure 6.1 shows a display of the computational cost in each iteration towards an optimal solution. The path descends gradually from the start of the iterations, although there are steep slopes along the path. The algorithm converges to an optimal value of 0.00011665 with an elapsed time of 0.401061 seconds. The MCPconf values are plotted on the *y*-axis of the data grid in Figure 6.2 as follows:

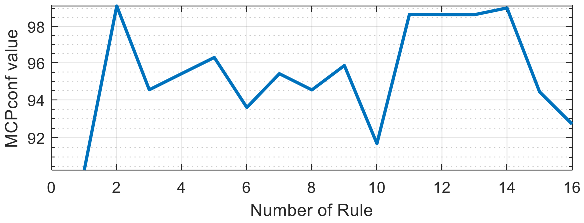


Figure 6.2: Dung beetle display on MCPconf value for KSA

Figure 6.2 shows the graphical view created using DBA from the MCPconf values in Table 6.1. The lowest MCPconf value was 90.3246%, while the highest was 99.1485%.

Table 6.2 shows the MCPconf values that were generated using ACO to generate mining association rules. These are tabulated as follows:

Table 6.2: MCPconf values from ACO

Rule #	MCPconf value (%)
1	92.9081
2	93.4948
3	95.3061
4	94.4317
5	95.3061
6	98.6503
7	99.0285
8	94.4278
9	95.3022
10	94.4278
11	95.8196
12	91.6604
13	98.6420

It is observed from Table 6.2 that thirteen association rules were generated within the MCPconf value of 0.9. Based on Table 6.2, the nature of the graphical display using DBA is as follows (where the *y*–axis represents the best computational cost and the *x*-axis represents the iteration):

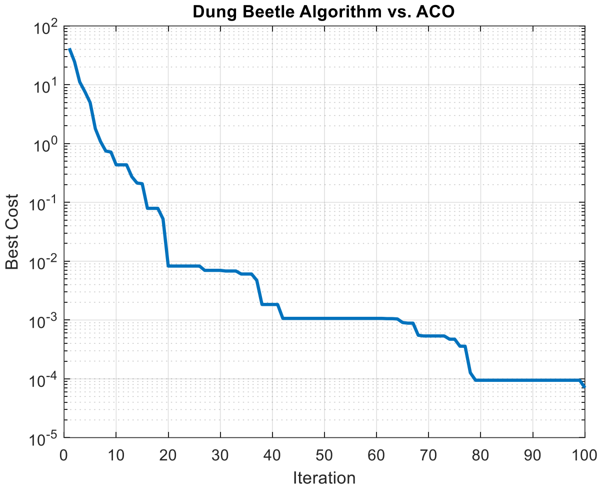


Figure 6.3: Dung beetle display of best cost on path traversed by ACO

Figure 6.3 shows the display of the best computational cost in each iteration towards an optimal solution. The path descends gradually from the start of iteration, maintains a steep slope and maintains a constant best cost between the 40th and 60th iterations before converging to an optimal value of 7.0315e-05 at an elapsed time of 0.485009 seconds.

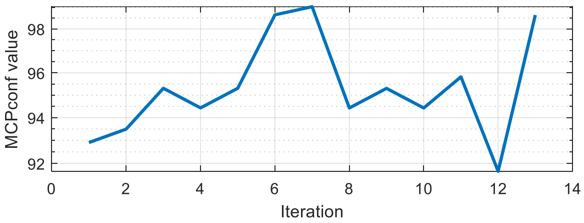


Figure 6.4: Dung beetle display of MCPconf value for ACO

Figure 6.4 shows the graphical view created by DBA on the MCPconf values generated for ACO during each iteration. The lowest MCPconf value was 91.6604%, while the highest was 99.0285%.

Table 6.3 shows the MCPconf values that were generated using PSO to mine association rules. These are tabulated as follows:

Table 6.3: MCPconf values from PSO

Rule #	MCPconf value (%)
1	99.2603
2	90.5613
3	94.9438
4	90.5107
5	90.5103

It is observed from Table 6.3 that five association rules were generated within the MCPconf value of 0.9. Based on Table 6.3, the nature of the graphical display using DBA is as follows (where the *y*–axis represents the best cost and the *x*-axis represents the iteration):

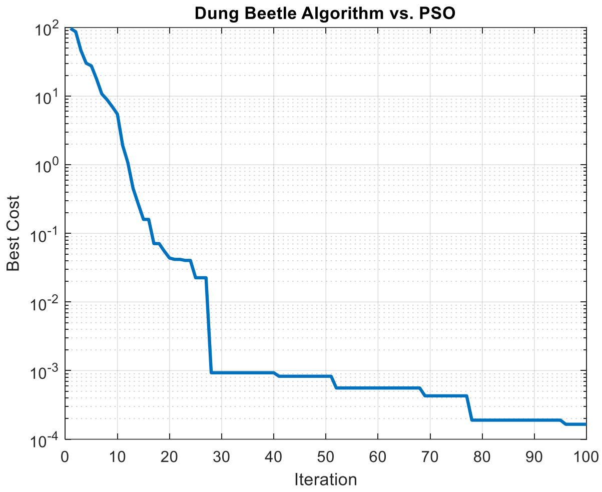


Figure 6.5: Dung beetle display best cost on path traversed by PSO

Figure 6.5 shows the display of the best computation cost (on the *y*-axis) for DBA in each iteration (on the *x*-axis) towards an optimal solution. The path descends gradually from the start of the iterations to the 20th iteration and then maintains a steep slope. The path also maintains a constant best cost (in terms of minimal value) before finally converging to an optimal value of 0.00016533 at an elapse time of 0.493069 seconds.

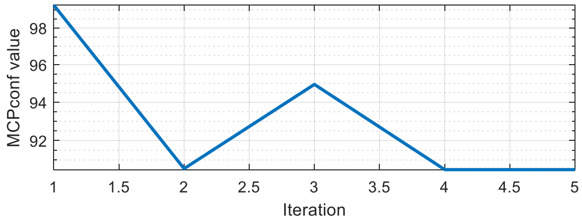


Figure 6.6: Dung beetle display on MCPconf value for PSO

Figure 6.6 shows the graphical view created by DBA on the MCPconf values generated for PSO during each iteration. The lowest MCPconf value was 90.5103%, while the highest was 99.2603%.

Table 6.4 shows the MCPconf values that were generated using BAT to mine association rules. These are tabulated as follows:

Table 6.4: MCPconf values from BAT

Rule #	MCPconf value (%)
1	90.5716
2	99.2277
3	94.9603
4	90.5264
5	90.5260

It is observed from Table 6.4 that five association rules were generated within the MCPconf value of 0.9. Based on Table 6.4, the nature of the graphical display using DBA is as follows (where the *y*–axis represents the best computational cost and the *x*-axis represents the iteration):

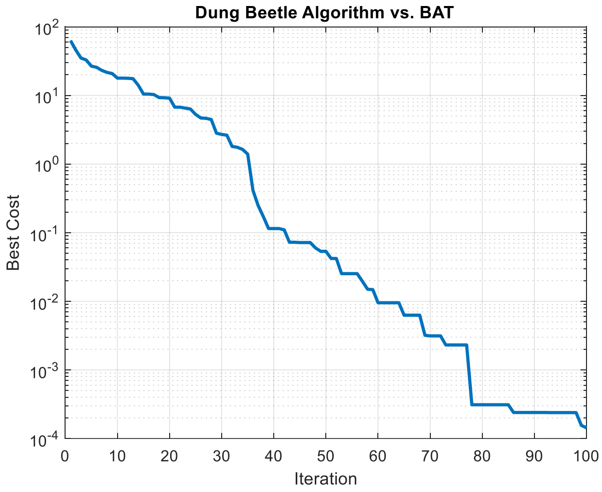


Figure 6.7: Dung beetle display best cost on path traversed by BAT

Figure 6.7 shows the display of the best computational cost (on the *y*-axis) for DBA in each iteration (on the *x*-axis) towards an optimal solution. The path descends gradually from the start of iteration, maintains a steep slope towards the 80th iteration, and maintains a constant best cost on the curve before converging to an optimal value of 0.00014318 at an elapse time of 0.589264 seconds.

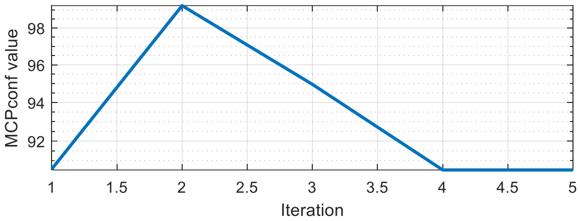


Figure 6.8: Dung beetle display of MCPconf value for BA

Figure 6.8 shows the graphical view created by DBA of the MCPconf values (on the *y*-axis), generated for BAT during each iteration interval (on the *x*-axis). The lowest MCPconf value was 90.5260% while the highest was 99.2277%.

Table 6.5 shows the MCPconf values that were generated using WSA-MP to mine association rules. These are tabulated as follows:

Table 6.5: MCPconf values from WSA-MP

Rule #	MCPconf value (%)
1	93.5951
2	95.4084
3	94.5331
4	95.9250
5	91.7612
6	98.7505
7	92.8651
8	98.6756
9	99.0538
10	95.3265
11	94.4520

It is observed from Table 6.5 that eleven association rules were generated within the MCPconf value of 0.9. Based on Table 6.5, the nature of the graphical display using DBA is as follows (where the *y*–axis represents the best computational cost and the *x*-axis represents the iteration):

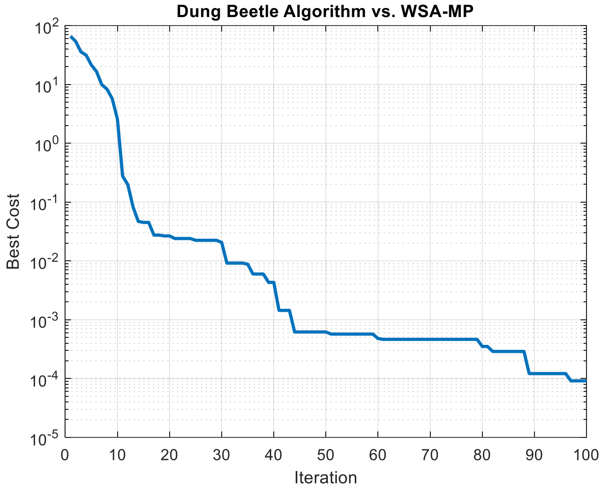


Figure 6.9: Dung beetle display of best cost on path traversed by WSA-MP

Figure 6.9 shows the display of the best computational cost (on the *y*-axis) in each iteration (on the *x*-axis) towards an optimal solution. The path descends gradually from the start of the iterations, maintains a constant best cost (in terms of minimal value) and converges to an optimal value of 9.1295e-05 in an elapsed time of 0.582776 seconds.

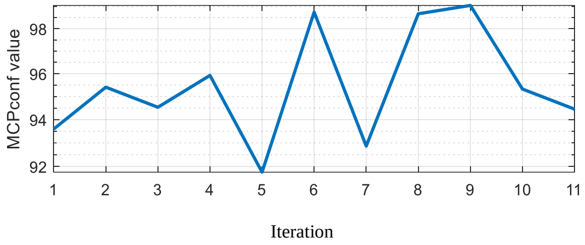


Figure 6.10: Dung beetle display of MCPconf value for WSA-MP

Figure 6.10 shows the graphical view created by DBA on the MCPconf values (on the *y*-axis) generated for WSA-MP during each iteration (on the *x*-axis). The lowest MCPconf value was 91.7612%, while the highest was 99.0538%.

6.5.2 Visualization using the bee algorithm

The bee algorithm for data visualization was also used to visualize results of association rules that were mined using KSA, ACO, PSO, BAT and WSA-MP. The MCPconf values in Table 6.1 were used to avoid repetition. In this sub-section, the *x*-coordinates of the graph represent iterations and the *y*-coordinates are the best cost. The nature of the graphical display using the bee algorithm on KSA is as follows:

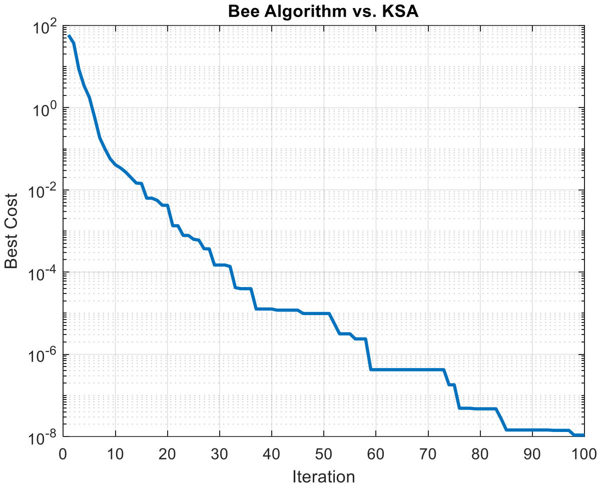


Figure 6.11: Bee algorithm display of best cost by KSA

Figure 6.11 shows the display of the best cost for the bee algorithm in each iteration towards an optimal solution. The curve descends gradually from the start of the iterations, maintains a constant best cost (in terms of minimal value) between the 60th and 70th iterations before converging to an optimal value of 1.0844e-08. Elapsed time is 2.167966 seconds.

For ACO, the MCPconf values in Table 6.2 were used to avoid repetition. The nature of the graphical display using the bee algorithm on ACO results is as follows:

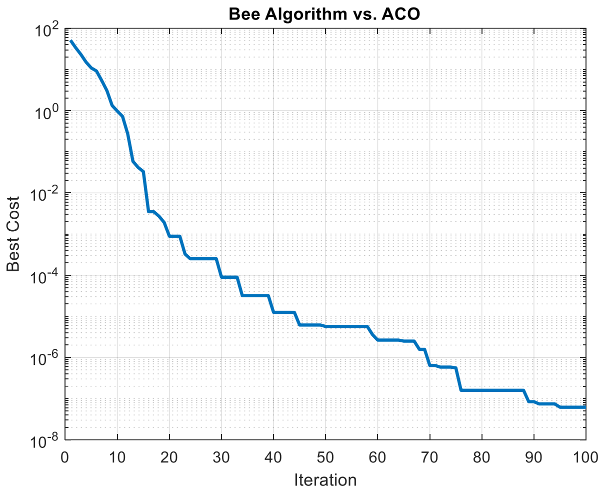


Figure 6.12: Bee algorithm display of best cost by ACO

Figure 6.12 shows the display of the best computational cost for bee algorithm in each iteration towards an optimal solution. The curve descends gradually from the start of the iterations and converges to an optimal value of 6.1772e-08 at an elapsed time of 2.134924 seconds.

For PSO, the MCPconf values in Table 6.3 were used to avoid repetition. The nature of the graphical display using the bee algorithm on PSO is as follows:

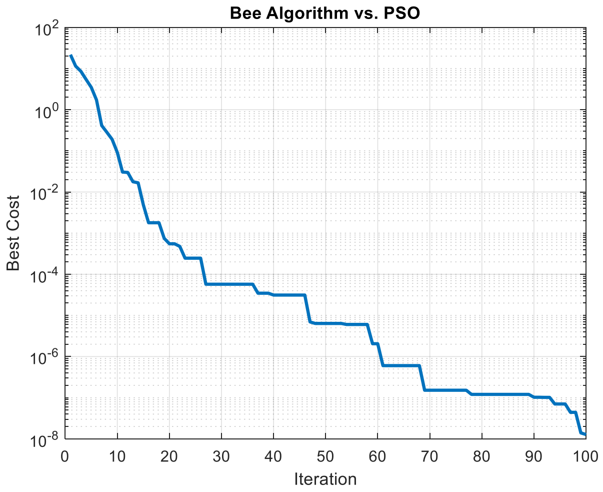


Figure 6.13: Bee algorithm display of best cost by PSO

Figure 6.13 shows the display of the best computational cost for the bee algorithm in each iteration towards an optimal solution. The curve descends gradually from the start of the iterations and converges to an optimal value of 1.2743e-08. Elapsed time is 2.186376 seconds.

For BAT, the MCPconf values in Table 6.4 were used to avoid repetition. The nature of the graphical display using the bee algorithm on BAT is as follows:

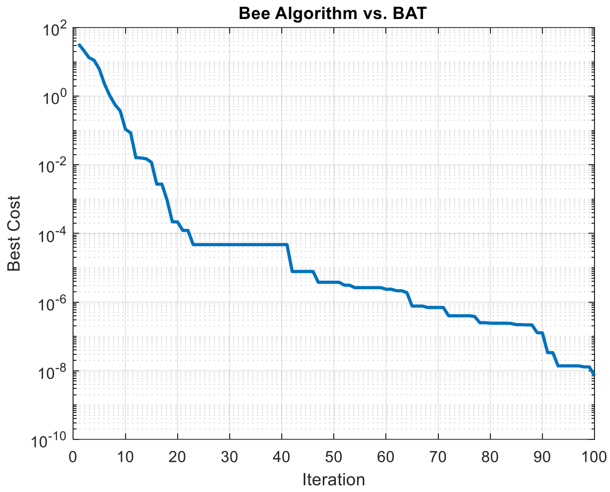


Figure 6.14: Bee algorithm display of best cost by BAT

Figure 6.14 shows the display of the best computational cost for the bee algorithm in each iteration towards an optimal solution. The curve descends gradually from the start of the iterations and maintains a constant best cost (in terms of minimal value) after the 20th iteration to the 40th iteration, until finally converging to an optimal value of 7.1857e-09. Elapsed time is 2.309688 seconds.

For WSA-MP, the MCPconf values in Table 6.5 were used to avoid repetition. The nature of the graphical display using the bee algorithm on WSA-MP is as follows:

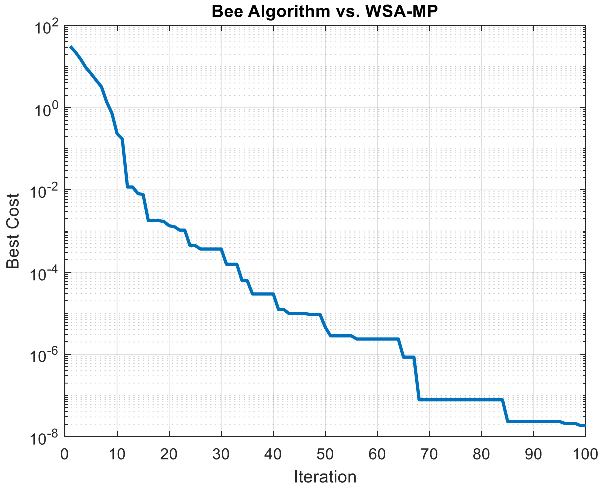


Figure 6.15: Bee algorithm display of best cost by WSA-MP

Figure 6.15 shows the display of the best computational cost for the bee algorithm in each iteration towards an optimal solution. The curve descends gradually from the start of the iterations and converges to an optimal value of 1.8478e-08. Elapsed time is 2.150612 seconds.

6.5.3 ACO for data visualization

Thirdly, ACO for data visualization was used to visualize the results on association rules that were mined from KSA, ACO, PSO, BAT and WSA-MP. In this sub-section, the *x*-coordinate of a graph represents iterations and the *y*-coordinate is the best cost.

For KSA, the MCPconf values in Table 6.1 were used to avoid repetition. The nature of the graphical display using ACO for data visualization on KSA is as follows:

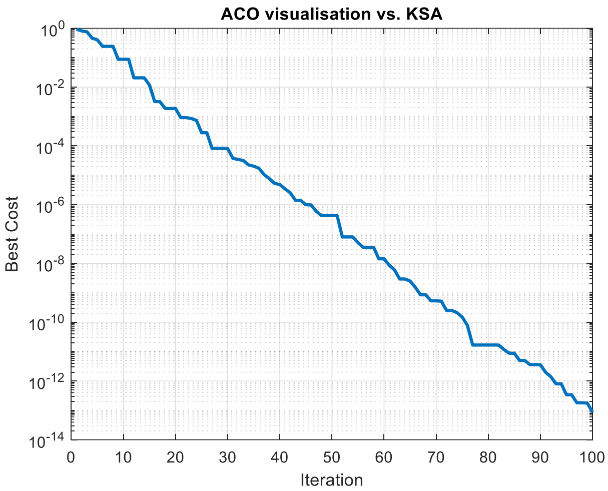


Figure 6.16: ACO algorithm display on best cost by KSA

Figure 6.16 shows the display of the best computational cost for the ACO algorithm in each iteration towards an optimal solution. The nature of the curve is linear or related to a straight line. However, the curve converges to an optimal value of 1.1458e-12 at an elapsed time of 1.020023 seconds.

For ACO, the MCPconf values in Table 6.2 were used to avoid repetition. The nature of the graphical display using ACO for data visualization on ACO for mining results is as follows:

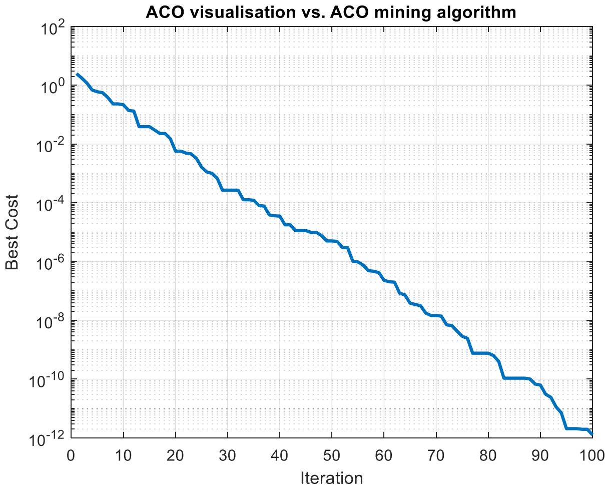


Figure 6.17: ACO for visualization display of best cost by ACO from data mining phase

Figure 6.17 shows the display of the best computational cost for the ACO algorithm in each iteration towards an optimal solution. The nature of the curve is linear but maintains a constant best cost towards the 100th iteration before converging to an optimal value of 1.2667e-12 at an elapsed time of 1.042381 seconds.

For PSO, the MCPconf values in Table 6.3 were used to avoid repetition. The nature of the graphical display using ACO for data visualization on PSO mining results is as follows:

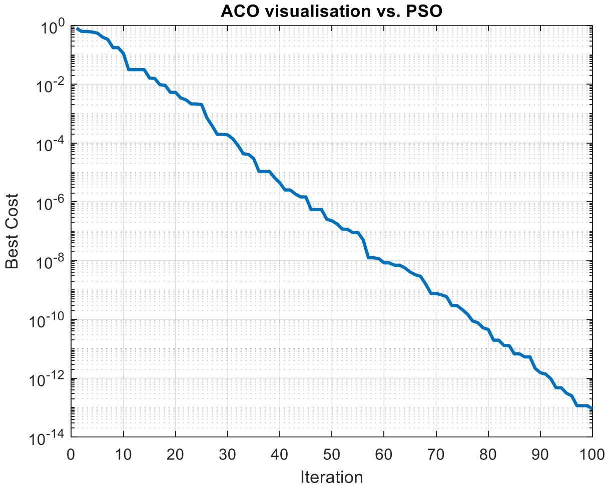


Figure 6.18: ACO for visualization display of best cost using results on PSO from data mining phase

Figure 6.18 shows the display of the best computational cost for the ACO in each iteration towards an optimal solution. The nature of the curve is linear. However, the curve converges to an optimal value of 8.9363e-14 with an elapsed time of 0.913326 seconds.

For BAT, the MCPconf values in Table 6.4 were used to avoid repetition. The nature of the graphical display using ACO for data visualization on BAT mining results is as follows:

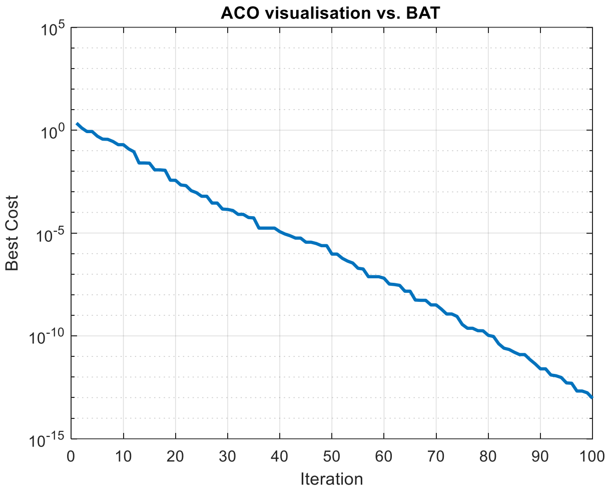


Figure 6.19: ACO for visualization display of best cost by BAT from data mining phase

Figure 6.19 shows the display of the best computational cost for the ACO algorithm in each iteration towards an optimal solution. The nature of the curve is linear. However, the curve converges to an optimal value of 9.2904e-14. Elapsed time is 0.956751 seconds.

For WSA-MP, the MCPconf values in Table 6.5 were used to avoid repetition. The nature of the graphical display using ACO for data visualization on WSA-MP mining results is as follows:

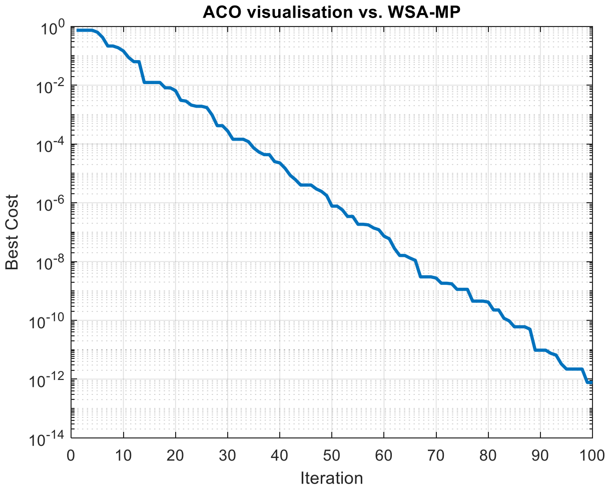


Figure 6.20: ACO for visualization display of best cost using results of WSA-MP from data mining phase

Figure 6.20 shows the display of the best computational cost for the ACO algorithm in each iteration towards an optimal solution. The nature of the curve is linear. However, the curve converges to an optimal value of 7.6804e-13. Elapsed time is 0.958605 seconds.

The nature of curves that were obtained from the bio-inspired data visualization algorithms, namely DBA, the bee algorithm and ACO for data visualization, indicates that while DBA and the bee algorithm maintained a curved path, ACO for data visualization is linear until it converges to a best optimal value. The high best computational cost observed at the initial iteration contributed to the nature of the graph in ACO for data visualization.

6.6 Evaluation of data visualization algorithms

The experimental results that were obtained on the computational time (that is, elapsed time) and optimum value (in terms of the best cost) are tabulated in Table 6.6 and Table 6.7. The computation time is measured in seconds. Table 6.6 shows the tabulated results on the optimum value of data visualization algorithms, namely the proposed DBA, the bee algorithm and ACO for data visualization. Meanwhile, Table 6.7 shows the computational time required by the algorithm to output both optimum results and visualization of data mining results on a data grid. The bio-inspired data mining algorithms considered are KSA, ACO, PSO, BAT and WSA-MP.

Table 6.6: Summary of optimum values from bio-inspired data visualization algorithms

	Bio-inspired data visualization algorithms		
Bio-inspired data mining algorithms	Proposed DBA (s)	Bee algorithm	ACO for data visualization
KSA	0.00011665	1.0844e-08	1.1458e-12
ACO	7.0315e-05	6.1772e-08	1.2667e-12
PSO	0.00016533	1.2743e-08.	*8.9363e-14*
BAT	0.00014318	7.1857e-09	*9.2904e-14*
WSA-MP	9.1295e-05	1.8478e-08	*7.6804e-13*
Mean	0.00012	2.457E-08	6.7E-13

Table 6.6 illustrates the optimal value (in terms of best computational cost) required for each algorithm to compute and display the results in a graphical format. It is observed that ACO for data visualization has the least optimum values among the bio-inspired data visualization algorithms, as evident from each bio-inspired data mining algorithm. It is possible that the number of search agents in the ACO for data visualization may have contributed to the algorithm's generating the fewest optimum values, as many parameters are tuned in the search space to produce each best optimal value.

Table 6.7: Summary of computation time obtained from bio-inspired data visualization algorithms.

	Bio-inspired data visualization algorithms		
Bio-inspired data mining algorithms	**Proposed DBA**	**Bee algorithm**	**ACO for data visualization**
KSA	0.401061	2.167966	1.020023
ACO	0.485009	2.134924	1.042381
PSO	0.493069	2.186376	*0.913326*
BAT	0.589264	2.309688	*0.956751*
WSA-MP	0.582776	2.150612	*0.958605*
Mean	0.510236	2.189913	0.978217

Table 6.7 illustrates the computational time (measured in seconds) required for each algorithm to compute and display results on a grid for users to view. The results shown in Table 6.7 indicate that the proposed DBA has the least computational time for each bio-inspired data mining algorithm. The computational time could be attributed to the parameters used in the algorithm. In order to find the mean computational time spent by each bio-inspired data visualization algorithm, the mean computational time was computed over all five algorithms. The proposed DBA spent 0.510236 seconds, the bee algorithm spent 2.189913 seconds, and ACO for data visualization spent 0.978217 seconds. The mean of the computational time shows that the computational time for ACO for data visualization is twice that of DBA, while the computational time for the bee algorithm is approximately four times that of DBA. Consequently, these computational time results indicate that the proposed DBA spent less computational time compared to

the bee algorithm and ACO for data visualization. Figure 6.21 shows a graph of the computational time for each bio-inspired data mining algorithm (where the *x*-coordinate represents the comparative bio-inspired algorithms for association rule mining and the *y*-coordinate is the computational time measure in seconds with unit intervals of 0.5):

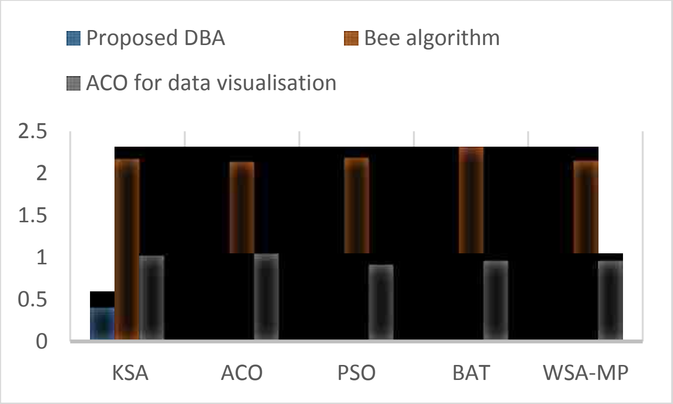


Figure 6.21: Graphical display of computational time for each bio-inspired data mining algorithm

In Figure 6.21, it is observed that the proposed DBA spent the least computational time in all the bio-inspired data mining algorithms. This computational time could be attributed to the nature of the algorithm in terms of the parameters that were used to enable it to converge at optimal solutions. Also, it is observed from Figure 6.21 that the bee algorithm has high computational time, which may be due to the nature of the algorithm's search for the best possible parameters in the search space.

6.7 Profile statistics on visualization algorithms

The profile summary report showed the overall executing of functions underlying the behavior of each algorithm for data visualization. The aspect of profiling, which is a step in the general outline of procedures (see section 3.5), is explained in the following using the function calls, total time and self time. Table 6.8 illustrates the major function calls extracted during profiling of the algorithms.

Table 6.8 Major function names of the comparative algorithms

	DBA	"Calls"	Total time	Self time
f1	DBAApproach>main	1	7.240	2.470
f2	DBAApproach>Orient	22510	1.087	1.087
f3	DBAApproach>PerformDance	17500	1.679	0.715
	Mean	13337	3.33533	1.424
	Bee algorithm			
f1	BeeApproach>main	1	1.901 s	0.724 s
f2	BeePhase3>Sphere	4010	0.107 s	0.107 s
f3	BeeApproach>PerformDance	3500	0.412 s	0.166 s
	Mean	2503.67	0.80667	0.33233
	ACO for visualization algorithm			
f1	ACOApproach>main	1	2.494 s	1.111 s
f2	ACOPhase3>Sphere	3010	0.152 s	0.074 s
f3	ACOphase3>RouletteWheelSelection	30000	0.696 s	0.696 s
	Mean	11003.67	1.114	0.627

Table 6.8 shows the different test functions (namely, function name, call, self time and total time) in each comparative visualization algorithm. The DBA, bee and ACO for visualization algorithms each have three major functions (one main- function f1 and two sub-functions).

In order to ensure a true reflection of the nature of built-in functions that were extracted, all built-in functions were considered for analysis. It is observed that when some built-in functions were called, the total time was zero seconds, thus making those built-in function calls inconsequential in terms of execution time. However, these inconsequential built-in functions were taken into consideration so as not to lose track of any function calls made during the visualization of results. Table 6.9 shows the results on the built-in function of the visualization algorithms.

Table 6.9: Major functions, mean of total_time and self_time of comparative algorithms

	Algorithm	**Number of major function "Calls"**	**Sum of "Calls" on major functions**	**Mean of major function "Calls"**	**Mean of major function "total_time (s)"**	**Mean of major function "self_time" (seconds (s))**	**Time difference (Td= (total_time - self_time)**
1.	ACO for visualization	3	33011	11003.67	1.114	0.627	0.487
2.	Bee	3	7511	2503.67	0.80667	0.33233	0.47434
3.	DBA	3	40011	13337	3.3353	1.424	1.9113

It is observed from Table 6.9 that ACO for visualization has a time difference of 0.487, the bee algorithm has a time difference of 0.47434, and DBA has a time difference of 1.9113. In terms of number of major function calls, ACO for visualization recorded 33011, the bee algorithm recorded 7511 and DBA recorded 40011. The results from Table 6.9 reveal that ACO for visualization has the smallest time difference of 0.487 seconds to call a total of 33011 major functions; the bee algorithm spent 0.47434 seconds to call a total of 7511 major functions; and DBA spent 1.9113 seconds to call a total of 40011 major function calls.

In order to ensure a true reflection of the nature of built-in functions that were extracted, all built-in functions were considered for analysis. Table 6.10 presents the comparative algorithms' built-in functions.

Table 6.10: In-built function calls on the comparative algorithms

No.	**Algorithm**	**Number of in- built functions**	**Sum of built-in function "Calls"**	**Sum of "self_time" (s)**	**Sum of "total_time" (seconds (s))**
1.	DBA	22	82559	4.357	3.007
2.	ACO for visualization	45	896	0.831	1.03
3.	Bee	45	6596	0.796	1.264

It is observed from Table 6.10 that, in terms of number of built-in functions, DBA has 22, ACO for visualization has 45 and Bee algorithm has 45. In terms of the sum of built-in function “calls”, DBA has 82,559, ACO for visualization has 896 and the bee algorithm has 6596. In terms of total time, DBA spent 3.007 seconds, ACO for visualization spent 1.03 seconds and the bee algorithm spent 1.264 seconds.

Table 6.11: Mean of built-in function calls on the comparative algorithms

No.	**Algorithm**	**Mean of built-in function “Calls”**	**Mean of “self_time” (s)**	**Mean of “total_time “(seconds (s))**	**Td**
1.	DBA	14290.5	0.198045	0.136682	-0.06136
2.	ACO	19.91111	0.018467	0.022889	0.004422
3.	Bee	146.5778	0.017689	0.028089	0.0104

It is observed from Table 6.11 that, although the mean of built-in functions in DBA was 14290.5, there is a negative time difference (-0.06136 seconds). This suggests that the mean of time of built-in functions did not play a role in terms of the performance of DBA. In contrast, ACO for visualization has the time difference of 0.004422 seconds, while the bee algorithm has a time difference of 0.0104 seconds.

In summary, firstly, DBA recorded a mean optimal value of 0.00012 with a mean computational time of 0.510236 seconds. The mean of time difference of the major function is 1.9113 seconds, and the mean of built-in function calls is -0.06136 seconds. This suggests that DBA spent less computational time on built-in functions and more time on major functions. This is possibly due to navigation and orientation at the major function in order to get an optimal value. Therefore, it is optimal rather to dance and find the nearly optimal value than to call other built-in functions.

Secondly, the bee algorithm recorded a mean optimal value of 2.457E-08 with a mean computational time of 2.189913 seconds. The mean of time difference of the major functions is 0.47434 seconds, and the mean of built-in function calls is 0.0104 seconds.

This suggests that although the bee algorithm gave a mean optimal value that is better than DBA, the bee algorithm spent a computational time of 2.189913 seconds, which is significantly higher and not suitable for a data visualization algorithm that handles large volumes of data.

Thirdly, ACO for data visualization recorded a mean optimal value of 6.7E-13 with a mean computational time of 0.978217 seconds. The mean of time difference of the major function is 0.487 seconds, and the mean of built-in function calls is 0.004422 seconds. This suggests that although the ACO for data visualization algorithm gave a mean optimal value that is better than both Bee and DBA, it spent a computational time of 0.978217 seconds, which make it unsuitable for a data visualization algorithm that handles large volumes of data.

6.8 Conclusion

The test dataset for the DBA is based on the modified closeness preference support-confidence (MCPsc or MCPconf) value obtained from the experimental results in Chapter 5. The DBA was compared with Bee and ACO for data visualization. In order to select the best algorithm, two criteria were used: firstly, the algorithm should have a minimum computational time; secondly, the algorithm should have an optimal value. However, an algorithm that meets the first criterion is given prominence. The results on average computational time (in seconds) showed that DBA had a minimum computational time of 0.510 seconds, Bee had a minimum computational time of 2.189 seconds, and ACO for data visualization had a minimum computational time of 0.978 seconds. Furthermore, the mean optimal value for DBA was 0.000117, for the bee algorithm was 2.46E-08, and for ACO for data visualization was 6.73E-13. The results indicated that DBA used the least computation time while ACO for data visualization had the lowest optimal value. Since the study is interested in the algorithm with minimum computational time, DBA is considered the preferred algorithm for data visualization when large datasets are involved. The results from DBA confirmed the suggestion that dung beetles are known to use minimal computational power for navigation and orientation using celestial polarization patterns (Wits University 2013).

6.9 Summary

In this chapter, mathematical formulations were expressed of the unique characteristics of dung beetles (that is, path integration with repulsion and attraction of trace, dance during orientation, and ball rolling on straight line) in creating imaginary homes after displacement of their food (dung) source. The mathematical formulations were translated into an algorithmic structure that searches for the best possible path and displays patterns using a simple two-dimensional view. The experimental results suggested that DBA uses minimum computation time to visualize data.

The next chapter draws conclusions from the findings and discusses the experimental results in line with the research questions that were formulated, as well as concepts that were considered in the literature review chapter of this study. The purpose of the next chapter is to either confirm or refute propositions that were made by researcher in respect of missing value estimation, duplicate text detection, feature selection, association rule mining and data visualization of frequently changed items in large datasets.

CHAPTER 7: DISCUSSION, CHALLENGES AND CONCLUSIONS

7.1 Introduction

This chapter concludes the study by stating the research questions (as stated in Chapter 1) and giving responses to the questions, and by discussing experimental results, the challenges of the proposed computational model, conclusions and recommendations for future work.

7.2 Research question

In this section, an overview is given of the research questions and responses to these research questions.

7.2.1 Research question 1:

Can a largely meta-heuristic/bio-inspired data preprocessing approach be modelled to extrapolate missing values, identify and remove duplicate text, and select features in subsets?

Response

Mathematical models based on the characteristics of an animal (the kestrel) can be formulated to extrapolate missing values (as shown in section 3.2.1), to identify and remove duplicate text (as shown in section 3.2.1.4) and to select features in classification of subsets (using the equations as shown in section 3.3).

The general outline of the procedure in section 3.5 was followed during the experiment on the data cleansing/preprocessing approach.

7.2.2 Research question 2:

- Can a mathematical expression and subsequent algorithm be formulated based on the hunting behavior of the kestrel to discover association rules on frequently changed patterns with numeric value and time dimensions?

Response

A mathematical model can be formulated on the selected hunting behavior of the kestrel. The detailed formulation is shown in section 3.2.1 of this thesis.

The general outline in section 3.5 was followed to model an algorithm for the discovery of association rules on frequently changed patterns, which include numeric value and time dimensions, as indicated in phase two of the methodological framework in Table 3.1.

7.2.3 Research question 3:

Based on the frequently changed rules, can a bio-inspired algorithm for the visualization of these association mining results be modelled?

Response

A mathematical model can be formulated on the behavior of the dung beetle for visualization of frequently changed items with numeric value, with less computational time in a two-dimensional graph. The detailed formulation is shown in section 3.4 of this thesis. Additionally, the general outline in section 3.5 was followed to model an algorithm for KSA and DBA, as indicated in the methodological framework in Table 3.1.

7.2.4 Research question 4:

Can a model and algorithmic structure be empirically validated on a benchmark dataset and evaluated against comparative meta-heuristic algorithms?

Response

The model and algorithmic structure can be empirically validated on a benchmark dataset and be evaluated against comparative meta-heuristic algorithms (shown in Table 3.1). The detailed validation and evaluation were shown in Chapters 4, 5 and 6 of this thesis.

7.3 Discussion of experimental results

The experimental results are discussed below in line with the research questions and methodological framework (see Table 3.1). The experimental results are used to evaluate the phase-based methodological framework.

7.3.1 Discussion of experimental results on extrapolating missing values

The proposed KSA for extrapolating missing values, when tested on a synthetic dataset with different dimensions/scales with multiple missing values at random, demonstrated uniqueness in terms of minimal value over the comparative meta-heuristic algorithms, namely BAT, WSA-MP and Firefly. The WSA-MP had the best minimum value in all six dimensions/scales of the dataset that was used, while the proposed KSA outperformed both BAT and Firefly in all six dimensions/scales of the dataset. The concept of parameter tuning, which has similarly been used in the maximum likelihood method for estimating missing values, helps in selecting a set of parameters or values that provides the best value to estimate missing values at random. In other words, parameter tuning provides an approximate parameter that becomes closer to the missing value (Zhao, MacKinnon and Gallup 2005) when there is very limited knowledge about the optimal solution (Luke 2015). Thus, the results from the proposed KSA point to the fact that there was limited knowledge on the nature of the proposed algorithm to adapt to different search problem domains. Since the proposed KSA could not perform better than WSA-MP, the idea of parameter tuning was applied to fine-tune the output results in subsequent phases of the proposed model.

Although the proposed KSA could not perform as expected for data imputation, a study was further conducted to examine the nature of the major function calls and built-in function calls in each algorithm. The concept of profiling (Sorensen *et al*. 2012) (that is, a function in MATLAB toolbox) was applied to extract the time of major function calls and inbuilt function calls for further tests. A statistical test was conducted on the major function calls and built-in function calls. The reason for this was to statistically test whether time of calling a function could change the performance of each function call in each algorithm. In order to achieve this, the Wilcoxon test was conducted to help rank each meta-heuristic algorithm, namely KSA, BAT, WSA-MP and Firefly, in terms of the major function calls and built-in function calls.

The results ranked WSA-MP first, followed by KSA, which was ranked second in terms of the built-in function calls. In terms of the major function calls only, the results ranked all meta-heuristic algorithms equally. The results of the Wilcoxon test showed that the total time to call a function in each algorithm did not change the performance of major functions. In order words, total time could not result in significant change in performance of inbuilt function calls in each meta-heuristic algorithm. A further statistical test was conducted using the Friedman test to check which meta-heuristic algorithm could be used as a way to control any possible error if algorithms' results were compared to each other. The results showed that KSA is the best algorithm to use as a control algorithm for multiple comparison of output results. This suggests that the combined results control (offset) the results from each meta-heuristic algorithm.

The significance of extrapolating a missing value approach to the big data analytics framework is that it improves on the quality of data analysis results (Rahm and Do 2000; Elmagarmid *et al*. 2007). Narang's (2013) description of missing data is, however, subjective in the sense that the user can best explain why data was missing. Nonetheless, this approach of attaching the missing value to reasons best known to the user (which can be either deliberate or not deliberate) can be minimized by using the proposed KSA for extrapolating missing values at random in big data. Big data is characterized by large volume; it is likely that since there are many users in big data environments, there could be many missing data values that can best be explained by the exponential number of users in the real world, which can be minimized using the proposed KSA.

7.3.2 Discussion of experimental results on duplicate text detection

The Naumann's (2013) framework provides a simplified process namely identity, similarity measure, algorithm used and evaluation. The study focuses on improving algorithm used and similarity measure. Two algorithms were evaluated and improved for duplicate word detection. The significance of duplicate detection of words/text is that it avoids inconsistency (that is, when data items referring to the same object contradict each other) in datasets, which might affect the quality of the analysis results (Elmagarmid *et al*. 2007). The inconsistency may be further exacerbated when the volume of data is very large. Thus, the unique feature of the enhanced algorithms (that is, the Smith-Waterman and Jaro-Winkler algorithms) is that, by applying the transitive closure and property of equality (symmetry property) to the algorithms, large volumes of data with duplicate words/text in a dataset can be identified, while mismatched (misspelt) words can be detected.

The experimental results showed that the Smith-Waterman algorithm guarantees accurate pairwise word comparison without missing any words in the dataset, while the Jaro-Winkler algorithm can miss words/text, leading to information loss. The ability of the Smith-Waterman algorithm to avoid missing words is considered as the basis for it to have been used to develop the BLAST algorithm (Altschul *et al*. 1990). However, the challenge of the BLAST algorithm is that it could not guarantee accurate results on duplicate words/text (Shpaer *et al*. 1996). The advantage of the Smith-Waterman algorithm is that when there are no similar words, no matching is done (Altschul *et al*. 1990).

In the experiment conducted using the heart disease dataset, the enhanced Smith-Waterman algorithm used a time of 0.055746s, while the enhanced Jaro-Winkler algorithm used a time of 0.140284 seconds to search for duplicate words/text. Thus, the enhanced Smith-Waterman algorithm has the lowest computational time to search for duplicate words. Altschul *et al*. (1990) indicate that the challenge with the Smith-Waterman algorithm is the high computational time in searching for duplicate words. This thesis provided an enhanced algorithm that takes into consideration the current dispensation of large volumes of data by using the property of transitive closure and the symmetry property of equality, which were applied to reduce the computational time for detection of duplicate words/text. The enhanced Smith-Waterman algorithm also guarantees accurate results irrespective of how large the dataset is. The results on the Smith-Waterman algorithm are significant for the big data analytics framework because when the velocity (speed of data) of processing information matters, in terms of less time to search for duplicate words/text without losing any information being processed, then the enhanced Smith-Waterman algorithm can use less computational time and guarantee more accurate results than the enhanced Jaro-Winkler algorithm.

7.3.3 Discussion of experimental results on feature selection

The proposed algorithm was evaluated against comparative bio-inspired algorithms, namely PSO, ACO, WSA-MP and BAT. The findings (shown in section 4.4) indicate that KSA produces a minimum learning rate in five out of nine datasets. Meanwhile, KSA produces the highest classification accuracy in four out of nine datasets. In terms of comparison of classification accuracy using the Wilcoxon signed-rank test, the findings of the test statistics suggest that there are no statistically significant differences between the comparative algorithms and the proposed algorithm. This suggests that KSA could be used as an alternative approach to feature selection for a classification problem. Also, it suggests that the initial parameters that were chosen in KSA guarantee good solutions that are comparable to other meta-heuristic search methods on feature selection. Aside from the choice of parameters, it also indicates that the random encircling, half-life of decay, frequency of bobbing and imitative behavior of kestrels guarantees good results in high volumes of datasets. In all the findings on feature selection, KSA produced the best classification accuracy (in terms of fewer errors) and the least number of features in a subset. This confirms the statement that for an algorithm to be considered the best, it should have higher classification accuracy and a smaller number of features in a subset (Mafarja and Mirjalili 2018).

In summary, the proposed algorithm (KSA) was compared to other bio-inspired algorithms such as the wolf, bat and Firefly algorithms. The results of the comparison showed that the KSA demonstrated potential uniqueness in its search for optimal values. The uniqueness that was demonstrated can be attributed to parameters that were applied in fine-tuning the mathematical formulation in order to obtain optimal results. The results from KSA are relevant because when presented with large volumes of data, KSA can extrapolate optimal results on numerical data that are considered missing. Statistical tests (Wilcoxon signed-rank test) conducted on the test function calls in respect of time showed that total time to call functions in the algorithms to extrapolate missing values could not result in significant changes in performance. The performance results are significant since algorithms written for big data analytics frameworks should not be constrained by time in executing a function to extrapolate missing values when speed of extrapolation (that is, the velocity characteristic of the big data analytics framework) is necessary. Thus, the proposed KSA would be suitable for handling velocity in big data analytic frameworks. Longbottom and Bamforth (2013) relate velocity to how fast incoming data is processed by algorithms and how quickly results are processed by algorithms. In this context, the results from function calls indicated that KSA could be applied since the algorithm is not constrained by time.

In order to test the accuracy of results from the proposed algorithm (that is, the veracity aspect of big data analytics framework or quality in terms of accuracy (Garcia, Luengo and Herrera 2015)), multiple comparison (that is, the Friedman test rank) was performed with other bio-inspired algorithms (Wolf, Bat and Firefly algorithms). The reason for multiple comparison was to find an algorithm that can be used to control the error rate from the comparative algorithms. The results of the Friedman test ranked KSA as the best control algorithm.

The experimental approach, through the use of mathematical formulations, was applied to detect duplicate text in data. A health-related dataset with multiples of duplicate text data was used to test the computational model for duplicate detection. Two duplicate detection techniques, namely the Jaro-Winkler algorithm and the Smith-Waterman algorithm were applied. In order to adapt these two algorithms to characteristics of big data analytics frameworks (such as the volume characteristic), the researcher wrote a program in MATLAB to implement the transitive closure and property of equality (reflexive property) to detect duplicate words from a data source. The experimental results showed that while the enhanced Smith-Waterman algorithm is accurate at pairwise word comparison without missing any words/text, the enhanced Jaro-Winkler algorithm could not perform pairwise word comparison on the n^{th}-1 instance (row) if the total number of instances (row) was an odd number, resulting in some data loss. The significance of these results is that the enhanced Smith-Waterman algorithm could perform accurate pairwise comparison when there is a large data volume with multiple duplicate words.

The experimental approach, through the use of mathematical formulations, was also applied to select best features and reduce the large volumes of data without losing relevant data. The proposed algorithm guaranteed optimal results when tested on high-dimensional datasets (bioinformatics dataset).

The researcher developed a data preprocessing model that could be adopted by big data analytics frameworks to extrapolate missing values, detect duplicate words/text and select relevant features when large volume is needed.

7.3.4 Discussion of experimental results on mining association rules

Vreeken and Tatti (2014) indicate that the use of frequency of items is not a very good measure of interestingness because it is subjective in the sense that it relies on users' evaluation of each pattern to disclose interestingness. This is problematic since a user has to search exponentially many potentially interesting patterns within predefined criteria and report on those patterns (Vreeken and Tatti 2014). Huynh (2010) indicates that action can change with time, thus making it tedious to search through large volumes of data. The challenge with the current frequency of items framework is that the time in which an action has taken place was not considered.

This thesis revealed that time can be included as part of the frequency of items framework so as to make provision for changing time dimensions. Although time may change, the user should be allowed to focus his/her attention on a particular period of interest by pre-defining the time interval criteria. After the criteria are defined, the proposed computational model searches for patterns within a close time dimension. Interestingly, from the experiment using the stock market dataset, the proposed KSA discovered patterns with a close time dimension of 0.058258 and 0.0087442. In this regard, Kaytoue, Kuznetsov and Napoli (2011) indicate that an item is close if it has the smallest time dimension for a set of items (Kaytoue *et al*. 2011) to occur together. Thus, the best time dimension of 0.0087442 produced an interesting pattern ($X1, X3 \rightarrow X10$ from the stock market dataset), which represents a sequence of actions (where $X1$, $X3$ and $X10$ are anonymous stock items).

In respect of time dimensions from other comparative meta-heuristic algorithms such as PSO (close time of 0.49668), ACO (close time of 0.49521), BAT (close time of 0.49833) and WSA-MP (close time of 0.49844), it is observed that all four meta-heuristic algorithms have time dimensions that were higher compared to the time dimension of the proposed KSA within a time interval of 0 to 0.7. Thus, time dimensions of the four meta-heuristic algorithms were not less than the time dimension from the proposed KSA.

Further analysis was conducted using the mean of time of each algorithm (as shown in section 5.4). The results of the experimental analysis showed that KSA had a mean time interval of 0.3, whereas the comparative algorithms, namely ACO, PSO, BAT and WSA-MP had mean time intervals of 0.4, 0.4, 0.4 and 0.5 respectively. Based on the mean MCPconf value and mean of time interval, KSA analyzed data on frequently changed items in the shortest possible time interval.

Tseng, Liang and Chu (2006) propose that the use of occurrence of items to measure pattern interestingness is insufficient in selecting actionable sequences for an organization (see also Yin *et al*. 2013). The experimental results suggested that including the time dimension provides sufficient measure to identify sequences of action that a user could take. These sequences of action are represented in terms of the sequence of rules that represent a pattern.

Additionally, the optimal value that was generated from the numeric value of the meta-heuristic algorithms, namely the proposed KSA (optimal value of 1.4823e-14), ACO (optimal value of 1.8168e-13), PSO (optimal value of 0.048569), BAT *(*optimal value of 3.0109e-07) and WSA-MP (optimal value of 6.7486e-07), showed that the proposed KSA had the best optimal value (in terms of minimum MCPconf) with 177 rules (complex rules) generated. Moreover, PSO and BAT both generated four complex rules and one simple rule, while WSA-MP and ACO both generated 100 complex rules. The minimum MCP support value that was computed from the proposed KSA using the stock market dataset accounted for the number of rules that were generated. Thus, the lower the MCPsupport value, the higher the possible chance to generate more rules.

Based on the experimental results on the mean time dimension, mean MCPconf and optimal value, it could be concluded that KSA had the best minimum value of 1.4823e-14 in which an average of 95.56% of rules were extracted at a mean of time of 0.31 seconds, which was the best among the comparative algorithms.

In summary, an experimental approach was adopted through the use of mathematical formulations to depict the behavior of kestrels. The mathematical expressions were translated into an algorithmic structure and tested on actual datasets. During the testing, two processes were followed: firstly, KSA was applied to automatically find the best minimum support threshold value from the numeric aspect of the dataset; and secondly, association rules were mined within the time dimension. The mathematical model was adopted from Railean *et al*. (2013) to address the aspect of time dimension.

The proposed computational model was tested on an actual dataset characterized as having volume and velocity, that is, stock market data. The results indicated that KSA had the optimal minimum value (minimum support threshold) compared to other meta-heuristic algorithms, namely PSO, ACO, BAT and WSA-MP. The optimal minimum value obtained with KSA is attributed to the fine-tuned parameters that were applied and the use of simplified basic rules applied to discover interesting patterns in the dataset. Additionally, the proposed algorithm showed a short time dimension of 0.008 among the comparative algorithms. Kaytoue *et al*. (2011) indicate that an item is close if it has the smallest time dimension for a set of items to occur together. The significance is that if the time interval is key to determining the best stock market items that had frequently changed, then KSA should be the preferred algorithm over ACO, PSO, BAT and WSA-MP. In view of the inclusion of the time dimension and numeric value aspects to frequency of item frameworks for mining association rules, this study has filled the gap identified in literature on the occurrence framework.

7.3.5 Discussion of experimental results on data visualization

Moere *et al*. (2006) note that extensive computational time is required for the use of ants to perform continuously pairwise localized negotiation of colors, that is, swapping the position of one ant with another ant, which relates to swapping one color with another in a single cluster and shape size scale negotiation. Keim *et al*. (1994) indicate that there is a lack of quantitative evidence of measuring the effectiveness of data visualization techniques. The quantitative measure uses a dataset and correlation coefficient of two dimensions, that is, the mean and variance of the dimensions, size and shape of clusters. Marghescu (2008) looks at the effectiveness of visualization techniques and observes that visualization is effective when it enables the user to read, understand and interpret the visual display easily, accurately, quickly, et cetera. Thus, effectiveness depends not only on the graphical design but also on the users' visual capabilities (Marghescu 2008). The limitation of these approaches is that the evaluation is based on only the users' experience and use of the visualization techniques. Card *et al*. (1999) indicate that effectiveness is the capability of a human to view a graphical display and interpret the results faster and convey distinctions in a display with fewer errors.

Based on the research by Moere *et al*. (2006), Keim *et al*. (1994) and Marghescu (2008), the present study used a different approach that is based on the optimal value from the visualization techniques (namely DBA, the bee algorithm and ACO for data visualization), which provides a quantitative measure on the quality (in terms of minimal value) of visualization technique. Additionally, effectiveness of visualization techniques is measured in terms of required time to complete a visualization task, in line with the statements of Dull and Tegarden (1999) and Risden and Czerwinski (2000). Additionally, the reason for proposing the dung beetle is its ability to use minimal computation time for navigation and orientation so as to help draw a visual pattern on a data grid. Thus, the computational time and optimal value from each meta-heuristic algorithm was used as two parameters to evaluate the visualization techniques (the proposed DBA, the bee algorithm and ACO for data visualization). The basis for using these two parameters is that as rules frequently change, visualization algorithms should use less time to compute the near-optimal results and display the outcome to users. The results after the implementation of the proposed DBA showed that DBA is effective at visualizing data mining results with less computational time, while the quality of near-optimal results/solutions could be improved. The significance of the results on effectiveness for big data analytics frameworks is that when limited computation time is required to visualize numeric data from large volumes of datasets, then the proposed DBA could be adopted in big data management when time taken to view volumes of data is important.

In summary, the experimental approach used a mathematical formulation to depict the behavior of dung beetles. The mathematical expression was translated into an algorithmic structure, and this helped empirically validate how the algorithm was applied to visualize (using two-dimensional graphs) numeric value results from the data mining phase of the proposed methodological framework. Additionally, aspects of the comparative meta-heuristic algorithms (bee algorithm and ACO for data visualization) were enhanced by the researcher to enable intake of the actual test dataset. The computational time and optimal results that were obtained were the two parameters that were used to help evaluate the effectiveness and quality of the data visualization techniques. The analysis results showed that DBA was effective at visualizing data mining results, while the quality of the optimum solution could be improved. The significance of the results on effectiveness for big data analytics frameworks is that when limited computation time is required to visualize numeric values in large volumes of datasets, DBA could be applied. Thus, DBA could be applied to improve on computational time required to view results in visual formats (that is, two-dimensional).

7.4 Challenges of the proposed model

The key challenge observed during the experiment was the quality of results on data visualization from DBA, and this can be enhanced and tested on different real-world problems to improve on the quality of results. In addition, the algorithm could be improved to help with the visualization of data in three-dimensional format.

Although the datasets used during the experiment were considered to have characteristics of big data (such as volume, velocity and value), it is possible that when the proposed computational model on feature subset selection is tested on high volumes (such as terabytes) of data, the computational performance may be one of the key challenges due to the reliance on a slow machine learning technique and complexity of design.

Although the various approach/algorithms were executed and evaluated at each phase of the methodological framework, the entire phase-based framework that is proposed was not executed as a whole package on a single dataset that has duplicate text, missing values, different features etc. which would test each algorithm in every phase. In view of this, evaluating the efficiency of the entire phase-based framework and comparing with other frameworks is a challenge because of dissimilarity of framework.

7.5 Conclusion and future work

This section presents the conclusion in term of the advantages of the proposed model, summary of contributions, success of the study and future work. The sub-section starts with the conclusion as follows:

7.5.1 Conclusion

The thesis proposed a computational model to address the challenge of frequently changed items with numeric and time dimensions through a three-phase approach. This approached applied largely a bio-inspired algorithm to address the gaps in literature on the occurrence framework. The novel aspects of the thesis include the basic mathematical formulation on the behavior of kestrels, the enhancement of the Smith-Waterman and Jaro-Winkler algorithms for big data frameworks, and the use of dung beetle behavior for data visualization of frequently changed patterns.

The experimental test conducted to validate the computational model indicates that KSA showed promising results that can be implemented in big data frameworks. Again, the minimal computation time of DBA also showed promising results for two-dimensional visualization of data (that is, numeric values) in big data frameworks.

In view of the largely bio-inspired nature of this thesis, the proposed model took into consideration the disadvantages of other bio-inspired algorithms (shown in Appendix 2) to arrive at the following conclusions on the advantages of the proposed model (KSA):

- the ability to self-adapt parameter values during random encircling, which was a challenge with WSA (as indicated in Appendix 2)
- the ability to move or switch the search space from exploration to exploitation within the right time or user-specified time, which was a challenge with BAT (as indicated in Appendix 2)
- the ability to find both local and global optima because of the exploration and exploitation behavior, which hitherto was a challenge with the Firefly algorithm (as indicated in Appendix 2)
- the ease and simplicity of implementation.

These advantages make the proposed algorithm different from other meta-heuristic algorithms considered in this study. Additionally, the mathematical formulation of the concept of half-life of substances, hitherto an idea in other science disciplines such as chemistry, makes the proposed KSA algorithm different from other meta-heuristic algorithms. The advantage of half-life is that it gives a lifespan for data items and adds interestingness to any data item.

Summary of contributions

- The researcher developed a bio-inspired algorithm that outperformed the best meta-heuristic algorithms considered in this study for missing value extrapolation.
- The researcher developed an enhanced algorithm that detected duplicate text from large data using the Smith-Waterman and Jaro-Winkler algorithms. The property of equality was applied to enhance these algorithms to enable more accurate duplicate detection from large datasets.
- The researcher developed a bio-inspired algorithm for a learning parameter in selecting relevant features for deep learning networks. This learning parameter is

significant in reducing the amount of time needed to make the network more feasible.

- The researcher developed a bio-inspired algorithm (KSA) for mining frequently changed patterns with a time and numeric dimension. This algorithm has adaptability for varying exploration and exploitation of pattern space.
- The researcher developed a bio-inspired algorithm for data visualization (DBA) in which movement and direction are self-regulated to have either exact or approximate paths that offer a full path integration, in contrast to PSO (see Appendix 2). This is a light-weight, inexpensive computational visualization approach.
- The uncooperative behavior of kestrels, which is part of swarm intelligence, was overcome by focusing on kestrels' learning through successful imitative behavior. This learning was mathematically formulated and implemented to provide the other advantage of skill rate and to improve accuracy of data visualization.

Success of the study

- The mathematical formulations in section 3.3, the proposed algorithms (KSA and DBA) and the comparative algorithms were successfully implemented in MATLAB and tested on benchmarked datasets, namely a bioinformatics dataset and a stock market dataset.

7.5.2 Future work

The future work on KSA for feature selection in classification is to develop new versions of KSA with modifications and enhancements of code. Similarly, aspects of the code of DBA for visualization could be modified and enhanced, and applied to different problem dimensions of data analysis for possible publication in academic journals.

Furthermore, based on the successful implementation of the proposed computational model on data mining, it is recommended that the proposed model (KSA) should be tested on parallel processing environments for the discovery of frequently changed patterns. Future work on KSA also requires rigorous comparison of the algorithm using the latest/state-of-the-art versions of other techniques on data mining.

Future work also requires the comparison of different frameworks (that is data cleansing, data mining and visualization) to identify the efficient framework for big data environment. Although, efficiency of framework is significant, it is not the focus of this thesis, and this could be explored provided that the proposed phases of the methodological framework is similar enough to others for an adequate comparison. Additionally, a possible benchmark with a threshold that shows the efficiency of the proposed phase-based framework compared with other existing frameworks is recommended, once similar enough frameworks emerge. Moreover, most big data analytics framework uses Extract Transform Load (ETL) which encompassed only part of the proposed framework, notably the data cleansing part. Other parts of the proposed framework, such as data mining, and visualization could not be part of frameworks and hence, they can not be compared with each other. Thus, in future work, when similar frameworks emerges, they can be compared. In view of this, evaluating the efficiency of the entire phase-based framework with others is outside the scope of this thesis.

Future work also requires the combination of KSA with the Smith-Waterman algorithm for duplicate text detection. The reason is that the Smith-Waterman algorithm is able to find duplicate text without losing any information, whereas KSA is able to find the best parameters in any given search problem space and is easy to implement. Additionally, future work requires the application of Naumann’s (2013) framework to identify duplicate text from multiple data sources.

Future work also requires the application of KSA to emerging research fields such as fog computing to handle data preprocessing and analysis. Basically, fog computing framework is an intermediary layer that allow sensor-based devices to connect with cloud computing environment. The objective of fog computing framework is to enable quick processing of raw data before being store on cloud environment. The proposed KSA could be explored on fog computing framework for data processing of sensor-based devices. In this regard, emerging frameworks on data processing and analysis could consider the use of KSA in the design of their framework.

References

Aamodt, T. 2015. Predicting stock markets with neural networks: A comparative study. Master's dissertation. University of Oslo. Available: https://www.duo.uio.no/bitstream/handle/10852/44765/aamodt-master.pdf?sequence=7 (Accessed 8 March 2017).

Abd-Alsabour, N., Randall, M. and Lewis, A. 2012. Investigating the effect of fixing the subset length using ant colony optimization algorithms for feature subset selection problems. In: *13th International Conference on Parallel and Distributed Computing, Applications and Technologies. IEEE*. DOI: 10.1109/PDCAT.2012.84 (Accessed 5 September 2018).

Abdel-Hamid, O., Mohamed, A.-R., Jiang, H., Deng, L., Penn, G. and Yu, D. 2014. Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on Audio, Speech and Language Processing*, 22(10): 1533–1545. DOI: 10.1109/TASLP.2014.2339736 (Accessed 5 September 2018).

Abdella, M. and Marwala, T. 2006. The use of genetic algorithms and neural networks to approximate missing data in database. *Computing and Informatics*, 24: 77–589. Available: http://www.cai.sk/ojs/index.php/cai/article/viewFile/401/320 (Accessed 8 May 2018).

Abdullah, A., Deris, S., Anwar, S. and Arjunan, S. N. V. 2013. An evolutionary firefly algorithm for the estimation of nonlinear biological model parameters. Available: https://doi.org/10.1371/journal.pone.0056310 (Accessed 8 May 2018).

Aboudi, N. E. and Benhlima, L. 2016. Review on wrapper feature selection approaches. Available: DOI: 10.1109/ICEMIS.2016.7745366 (Accessed 8 May 2018).

Acock, A. C. 2005. Working with missing values. *Journal of Marriage and Family*, 67: 1012–1028. Available: http://www.scirp.org/(S(czeh2tfqyw2orz553k1w0r45))/reference/ReferencesPapers.aspx?ReferenceID=1719773 (Accessed 8 May 2018).

Adamo, J-M. 2001. *Data mining for association rules and sequential patterns*. New York: Springer-Verlag. Available: DOI: 10.1007/978-1-4613-0085-4 (Accessed 5 February 2017).

Agbehadji, I. E. 2011. Solution to the travel salesman problem, using omicron genetic algorithm. Case study: tour of national health insurance schemes in the Brong Ahafo region of Ghana. M.S.c (Industrial Mathematics). Kwame Nkrumah University of Science and Technology. Available: DOI: 10.13140/RG.2.1.2322.7281 (Accessed 4 February 2017).

Agbehadji, I. E., Millham, R. and Fong, S. 2016. Wolf search algorithm for numeric association rule mining. In: *2016 IEEE International Conference on Cloud Computing and Big Data Analysis (ICCCBDA 2016), Chengdu, China*. 146-151. DOI: 10.1109/ICCCBDA.2016.7529549 (Accessed 4 March 2018).

Agbehadji, I. E., Millham, R., Fong, S. J. and Yang, H. 2018. Bio-inspired computational approach to missing value estimation. *Mathematical Problems in Engineering-Hindawi*, 2018: 16.b Available: https://doi.org/10.1155/2018/9457821 (Accessed: 19 February 2018).

Agbehadji, I. E., Millham, R., Fong, S. J. and Yang, H. 2018a. The comparative analysis of Smith-Waterman algorithm with Jaro-Winkler algorithm for the detection of duplicate health related records. In: *Proceedings of International Conference on Advances in Big Data, Computing and Data Communication Systems, IEEE. 1-10.* Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8465458

Agbehadji, I. E., Millham, R., Fong, S. J. and Yang, H. 2018b. Kestrel-based Search Algorithm (KSA) for parameter tuning onto Long Short Term Memory (LSTM) Network for feature selection in classification of high-dimensional bioinformatics datasets. *Federation Conference of Computer Science and Information systems (FedCSIS). 15:15-20.* https://annals-csis.org/proceedings/2018/drp/pdf/52.pdf

Aggarwal, C. C. and Han, J. 2014. *Frequent pattern mining*. Springer Available: DOI: 10.1007/978-3-319-07821-2_3 (Accessed 10 August 2017).

Aggarwal, S. and. Rani, B. 2013. Optimization of association rule mining process using apriori and ant colony optimization algorithm. *International Journal of Current Engineering and Technology*, 3 (2): 623. Available: http://inpressco.com/wp-content/uploads/2013/06/Paper73620-623.pdf (Accessed 10 August 2017).

Agrawal. R. and Srikant, R. 1995. Mining sequential patterns. In: *Proceedings of International Conference on Data Engineering (ICDE '95)*. Department of Computer Science, University of Wisconsin, Madison. pp. 3–14. Available: https://pdfs.semanticscholar.org/d6a0/e0b04a020ac6422b98b8e63027a6178060fd.pdf (Accessed 10 August 2018).

Ahmed, H. and Glasgow, J. 2012. Swarm intelligence: Concepts, models and applications. Technical Report 2012-585. School of Computing, Queen's University, Kingston, Ontario, Canada. pp. 1-51. Available: https://pdfs.semanticscholar.org/116b/67cf2ad2c948533e6890a9fccc5543dded89.pdf (Accessed 10/8/2018).

Ákos, Z., Nagy, M., Leven, S. and Vicsek, T. 2010. Thermal soaring flight of birds and unmanned aerial vehicle. *Bioinspiration and Biomimetics*. 5: 12. Available: https://arxiv.org/ftp/arxiv/papers/1012/1012.0434.pdf (Accessed 10

August 2018).

Al-Ani, A. 2007. Ant colony optimization for feature subset selection. *World Academy of Science, Engineering and Technology International Journal of Computer, Electrical, Automation, Control and Information Engineering*, 4(1): 1–4. Available: https://pdfs.semanticscholar.org/bb34/e4ba7cc9a90bb937c13daff76cdfc5b01ee4.pdf (Accessed 10 August 2018).

Al-Ani, A., Alsukker, A. and Khushaba, R. N. 2012. Feature subset selection using differential evolution and a wheel based search strategy. *Swarm and Evolutionary Computation*, 9: 15–26. Available: http://dx.doi.org/10.1016/j.swevo.2012.09.003 (Accessed 10 August 2018).

Alatas, B. and Akin, E. 2006. An efficient genetic algorithm for automated mining of both positive and negative quantitative association rules. *Soft Computing*, 10(3): 230–237. Available: http://sci2s.ugr.es/keel/pdf/algorithm/articulo/2006%20-%20SC%20-%20Alatas%20-%20An%20efficient%20genetic%20algorithm%20for%20automated%20mining%20of%20both%20positive.pdf (Accessed 10 August 2018).

Alatas, B., Akin E. and Karci A. 2007. Modenar: Multiobjective differential evolution algorithm for mining numeric association rules. *Applied Soft Computing*, 8(1): 646–656. Available: DOI: 10.1016/j.asoc.2007.05.003 (Accessed 23 May 2017).

Alatas, G. and Akin, E. 2008. Chaos rough particle swarm optimization and its applications. *Information Sciences*, 163: 194–203. Available: https://pdfs.semanticscholar.org/0c56/34c1242871cda43c66adda614d625de6c8fa.pdf?_ga=2.114524743.1505521392.1536153887-1317843931.1533388075 (Accessed 5 September 2018).

Allison, P. D. 2012. Handling missing data by maximum likelihood. Haverford, PA. Available: https://statisticalhorizons.com/wp-content/uploads/MissingDataByML.pdf (Accessed 23 May 2017).

Almuallim, H. and Dietterich, T. G. 1994. Learning boolean concepts in the presence of many irrelevant features. *Artificial Intelligence*. 69(1–2): 279–305. Available: http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=1D673AA2C834BDB18469741C45A3E2AC?doi=10.1.1.47.5088&rep=rep1&type=pdf (Accessed 23 May 2017).

Altschul, S. F., Gish, W., Miller, W., Myers, E. W. and Lipman, D. J. 1990. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3): 403–410. Available: https://publications.mpi-cbg.de/Altschul_1990_5424.pdf (Accessed 10 July 2017).

Andel, D. and Wehner, R. 2004. Path integration in desert ants, cataglyphis: How to make a homing ant run away from home. *Proceeding Biological Science*. 271 (1547): 1485–1489. Available: DOI: 10.1098/rspb.2004.2749 (Accessed 10 August 2018).

Attaway, S. 2009. Matlab: *A practical introduction to programming and problem solving*. Boston: Elsevier. Available: http://www.dm.unibo.it/~piccolom/didattica/num_met/SAmatlab_09.pdf (Accessed 2 February 2018).

Ayres, J., Gehrke, J., Yiu, T. and Flannick, J. 2002. Sequential pattern mining using a bitmap representation. In: *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD' 02)*. 429–435. Available: http://delivery.acm.org/10.1145/780000/775109/p429-ayres.pdf?ip=196.21.61.167&id=775109&acc=ACTIVE%20SERVICE&key=64

6D7B17E601A2A5%2E20146AEDC3D229CC%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&__acm__=1535526039_c428a19ef9cbd643e4eb175b7afc41a2 (Accessed 2 February 2018).

Bai, Q. 2010. Analysis of particle swarm optimization algorithm. *Computer and Information Science*, 3(1): 1–5. Available: https://pdfs.semanticscholar.org/6d00/7fd5f734cb4c6c9457e22f46a58be5edb662.pdf (Accessed 20 May 2018).

Bandura, A. 1971. *Social learning theory*. New York: General Learning Press. Available: www.asecib.ase.ro/mps/Bandura_SocialLearningTheory.pd (Accessed 10 June 2018).

Banupriya, S. and Vijayadeepa, V. 2015. Data flow of motivated data using heterogeneous method for complexity reduction. *International Journal of Innovative Research in Computer and Communication Engineering*, 3(9): 1–7. Available: http://www.ijircce.com/upload/2015/september/110_DATA.pdf (Accessed 10 June 2018).

Bard, G. V. 2007. Spelling-error tolerant, order-independent pass-phrases via the Damerau–Levenshtein string-edit distance metric. In: *Proceedings of the Fifth Australasian Symposium on ACSW Frontiers*. Ballarat, Australia, 30 January – 2 February 2007, Conferences in Research and Practice in Information Technology, 68. Darlinghurst, Australia: Australian Computer Society, 117–124. Available: *https://dl.acm.org/citation.cfm?id=1274545* (Accessed 10 June 2018).

Barto, A. G. and Sutton, R. 1997. *Introduction to reinforcement learning*. Cambridge, Massachusetts: The MIT Press. Available: http://incompleteideas.net/book/bookdraft2017nov5.pdf (Accessed: 5 September 2018).

Batres-Estrada, G. 2015. Deep learning for multivariate financial time series.

Online Master of Science Thesis in Engineering. Stockholm: KTH, School of Engineering Sciences, Mathematics Department, Mathematical Statistics. Available: https://www.math.kth.se/matstat/seminarier/reports/M-exjobb15/150612a.pdf (Accessed 20 May 2018).

Bellman, R. 1957. *Dynamic programming*. Princeton: Princeton University Press. Available: https://data.epo.org/gpi/409729456.pdf?download=true (Accessed 10 August 2018).

Ben-Bassat, M. 1982. Pattern recognition and reduction of dimensionality. In: Krishnaiah, P. R. and Kanal, L.N. eds. *Handbook of statistics II*. North Holland, 773–791. Available: http://www.scirp.org/(S(lz5mqp453edsnp55rrgjct55))/reference/ReferencesPapers.aspx?ReferenceID=1337614 (Accessed 10 August 2018).

Bertsekas, D. P. 2005. *Dynamic programming and optimal control*. Belmont. Massachusetts: Athena Scientific.

Bikakis, N. 2018. *Big data visualization tools: Encyclopedia of big data technologies*. Springer.

Bilenko, M., Mooney, R. J., Cohen, W. W., Ravikumar, P. and Fienberg, S. E. 2003. Adaptive name matching in information integration. *IEEE Intelligent Systems*, 18(5): 16–23. Available: https://ieeexplore.ieee.org/document/1234765/ (Accessed 10 August 2018).

Bishop, C. M. 1995. *Neural networks for pattern recognition*. New York: Oxford University Press.

Bishop, C. M. 2006. *Pattern recognition and machine learning*. New York: Springer. Available: http://users.isr.ist.utl.pt/~wurmd/Livros/school/Bishop%20-%20Pattern%20Reco gnition%20And%20Machine%20Learning%20-%20Springer%20%202006.pdf (Accessed 10 May 2017).

Blum, C. and Roli, A. 2003. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, 35(3): 268–308. Available: https://www.iiia.csic.es/~christian.blum/downloads/blum_roli_2003.pdf (Accessed 20 May 2018).

Bonabeau, E., Dorigo, M. and Theraulaz, G. 2000. Inspiration for optimisation from social insect behaviour. *Nature*, 406(6791): 39–42. Available: http://cognition.ups-tlse.fr/IMG/pdf/35.pdf (Accessed 20 May 2018).

Borschev, V. and Partee, B. 2001. Lecture 3: Properties of relations. *Mathematical Linguistics.* 1–4. Available: http://people.umass.edu/partee/726_01/lectures/lecture3.pdf (Accessed 20/05/2018).

Boser, B. E., Guyon, I. M., Vapnik, V. N. A. 1992. Training algorithm for optimal margin classifiers. http://w.svms.org/training/BOGV92.pdf (Accessed 10 May 2017).

Breiman, L. 1999. *Using adaptive bagging to debias regressions: Technical report 547*. Berkeley: Statistics Department, University of California at Berkeley. Available: https://pdfs.semanticscholar.org/636c/243ae176bcfa9766f8d4fca7eb441819e21d.pdf (Accessed 20 May 2018).

Breiman, L. 2001. Random forests. Available: https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf (Accessed 10 May 2017).

Bryson, S., Kenwright, D. N., Cox, M., Ellsworth, D. and Haimes, R. 1999. Visually exploring gigabyte datasets in realtime. *Communication of the ACM*, 42(8): 82–90. Available: http://www.dcs.ed.ac.uk/teaching/cs4/www/visualisation/SIGGRAPH/gigabyte_datasets2.pdf (Accessed 10 August 2018).

Bu, Y., Borkar, V. R., Carey, M. J., Rosen, J., Polyzotis, N., Condie, T., Weimer, M. and Ramakrishnan, R. 2012. Scaling datalog for machine learning on big data. Available: http://dblp.uni-trier.de/db/journals/corr/corr1203.html#abs-1203-0160 (Accessed 10/9/2017).

Card, S. K., Mackinlay, J. D. and Shneiderman, B. 1999. *Readings in information visualization: Using vision to think*. Academic Press. Available: https://www.researchgate.net/publication/220691172_Readings_in_Information_Visualization_Using_Vision_To_Think (Accessed: 05-September-2018).

Carter, R. L. 2006. Solutions for missing data in structural equation modeling. *Research and Practice in Assessment*, 1(1): 1–6. Available: https://pdfs.semanticscholar.org/1217/c3e4d02dc40ee941b92473df4077458d5571.pdf (Accessed 10 April 2018).

Cavnar, W. B. and Trenkle, J. M. 1994. N-gram-based text categorization. In: *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*. pp. 1–14. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.53.9367 (Accessed 10 September 2017).

CLC bio. 2007. Bioinformatics explained: BLAST. Available: http://www.ccg.unam.mx/~vinuesa/tlem/pdfs/Bioinformatics_explained_BLAST.pdf

Cohen, W. W., Ravikumar, P. and Fienberg, S. E. 2003. A comparison of string distance metrics for name-matching tasks. In: *Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web (IIWeb-03).* pp. 1–6. Available: http://www.cs.cmu.edu/~wcohen/postscript/ijcai-ws-2003.pdf

Cong, S., Han, J. and Padua, D. 2005. Parallel mining of closed sequential patterns. Available: http://hanj.cs.illinois.edu/pdf/kdd05_parseq.pdf (Accessed 4 June 2017).

Cormen, T. H., Leiserson, C. E. and Rivest, R. L. 1990. *Introduction to algorithms*. 3rd ed. Cambridge: MIT Press. Available: http://labs.xjtudlc.com/labs/wldmt/reading%20list/books/Algorithms%20and%20optimization/Introduction%20to%20Algorithms.pdf (Accessed 3 May 2018).

Creswell, J. W. 2013. *Research design: Qualitative, quantitative, and mixed method approaches I.* 2nd ed. Thousand Oaks, London and New Delhi: SAGE Publications. Available: https://www.ucalgary.ca/paed/files/paed/2003_creswell_a-framework-for-design.pdf (Accessed 9 June 2018).

Crone, S. F., Lessmann, S, and Stahlbock, R. 2006. The impact of preprocessing on data mining: An evaluation of classifier sensitivity in direct marketing. *European Jornal of Operational Research*, 173: 781–800. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.302.48&rep=rep1&type=pdf (Accessed 9 June 2018).

Cui, X. and Potok, T. E. 2006. A distributed agent implementation of multiple species flocking model for document partitioning clustering. In: *International Workshop on Cooperative Information Agents, Springer, Berliin, Heidelberg: Cooperative information agents*. 124–137. Available: https://pdfs.semanticscholar.org/96f7/71c0d09c8283b8d9aa6a4c297e31eaca07f9.pdf (Accessed 9 June 2018).

Curtin, R. R., Cline, J. R., Slagle, N. P., March, W. B., Ram, P., Mehta, N. A. and Gray, A. G. 2013. MLPACK: A scalable C++ machine learning library. *J Mach Learn Res*. 14: 801–805. Available: http://www.jmlr.org/papers/volume14/curtin13a/curtin13a.pdf (Accessed 2 August 2018).

Dacke, M., Byrne, M. J., Baird, E., Scholtz, C. H. and Warrant, E. J. 2011. How dim is dim? Precision of the celestial compass in moonlight and sunlight. *Philos. Trans. R. Soc. Lond. B Biol. Sci*., 366: 697–702. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3049003/ (Accessed 2 August 2018).

Damerau, F. J. 1964. A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 7(3): 171–176. Available: doi.10.1145/363958.363994 (Accessed 2 August 2018).

Dănăilă, I., Dinu, L. P., Niculae, V. and Sulea, O.-M. 2012. String distances for near-duplicate detection. Available: http://nlp.unibuc.ro/papers/danaila12.pdf (Accessed 2 August 2018).

Darwin, C. 1868. *The variation of animals and plants under domestication*. London: John Murray.

Dash, M. and Liu, H. 1997. Feature selection for classification. *Intelligent Data Analysis*, 1(1–4): 131–156. Available: https://www.sciencedirect.com/science/article/pii/S1088467X97000085 (Accessed 2 August 2018).

Dell'Amore, C. 2013. Dung beetles navigate via the milky way, first known in animal kingdom. News Watch. *National Geographic Society* (blog). Available: https://blog.nationalgeographic.org/2013/01/24/dung-beetles-navigate-via-the-milky-way-first-known-in-animal-kingdom/ (Accessed 2 August 2018).

Demirkan, H. and Delen, D. 2013. Leveraging the capabilities of service-oriented decision support systems: Putting analytics and big data in cloud. *Decis Support Syst*., 55(1): 412–421. Available: DOI: 10.1016/j.dss.2012.05.048 (Accessed 3 May 2018).

Deng, L. and Chen, J. 2014. Sequence classification using the high-level features extracted from deep neural networks. In: *Proceedings of International Conference on Acoustics Speech and Signal Processing (ICASSP)*. pp. 1–5. Available: DOI: 10.1109/ICASSP.2014.6854926 (Accessed 3 May 2018).

Deng, L. and Yu, D. 2013. Deep learning: Methods and applications. *Foundations and Trends in Signal Processing*, 7(3–4): 197–387. Available: http://dx.doi.org/10.1561/2000000039 (Accessed 9 June 2018).

Deng, L. and Yu, D. 2013. Deep learning: Methods and applications. *Foundations and Trends in Signal Processing*, 7(3–4): 197–387. Available: https://scholar.google.co.za/scholar?q=Deng,+LI.+and+Yu,+D.+2013.+Deep+learning:+Methods+and+applications&hl=en&as_sdt=0&as_vis=1&oi=scholart (Accessed 9 June 2018).

Devakunchari, R. 2014. Analysis on big data over the years. *International Journal of Scientific and Research Publications*, 4(1). Available: http://www.ijsrp.org/research-paper-0114/ijsrp-p2573.pdf (Accessed 3 May 2017).

Dietterich, T. 1998. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting and randomization. *Machine Learning*, 40(2): 1–22. Available: DOI: 10.1023/A:1007607513941 (Accessed 3 May 2018).

Ding, X., Zhang, Y., Liu, T. and Duan, J. 2015. Deep learning for event-driven stock prediction. In: *Proceedings of the 24th International Joint Conference on*

Artificial Intelligence (IJCAI 2015). Available: https://www.ijcai.org/Proceedings/15/Papers/329.pdf (Accessed 6 September 2018).

Dorigo M. and Cambardella, L. M. 1997. Ant colony system: A cooperative learning approach to traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1): 53–66. Available: http://people.idsia.ch/~luca/acs-ec97.pdf (Accessed 3 May 2018).

Dull, R. B. and Tegarden, D. P. 1999. A comparison of three visual representations of complex multidimensional accounting information. *Journal of Information Systems*, 13(2): 117–131. Available: https://pdfs.semanticscholar.org/cfcc/ab4e9daff7edddfbffb85a6680ce970966ee.pdf (Accessed 3 May 2018).

Eberhart, R. C., Shi, Y. and Kennedy, J. 2001. *Swarm intelligence: The Morgan Kaufmann series in artificial intelligence*. Elsevier. Available: https://www.elsevier.com/books/swarm-intelligence/eberhart/978-1-55860-595-4 (Accessed 3 May 2018).

Elisseeff, A. and Guyon, I. 2003. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3: 1157–1182. Available: http://delivery.acm.org/10.1145/950000/944968/3-1157-guyon.pdf?ip=196.21.61.167&id=944968&acc=OPEN&key=646D7B17E601A2A5%2E20146AEDC3D229CC%2E4D4702B0C3E38B35%2E6D218144511F3437&__acm__=1535534581_0ddaee2daaf021a98d15105675f80a03 (Accessed 3 October 2017).

Elmagarmid, A. K., Ipeirotis, P. G. and Verykios, S. V. 2007. Duplicate record detection: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 19(1): 1–16. Available: https://www.cs.purdue.edu/homes/ake/pub/TKDE-0240-

0605-1.pdf (Accessed 3 October 2017).

Englert, P., Paraschos, A., Peters, J. and Deisenroth, M. P. 2013. Probabilistic model-based imitation learning. 1-18. Available: http://www.ias.tu-darmstadt.de/uploads/Publications/Englert_ABJ_2013.pdf (Accessed 3 May 2017).

Essa, Y. M., Attiya, G. and El-Sayed, A. 2013. New framework for improving big data analysis using mobile agent based. *International Journal of Advanced Computer Science and Applications (IJACSA)*, 5(3): 1–8. Available: http://thesai.org/Downloads/Volume5No3/Paper_3-New_Framework_for_Improving_Big_Data_Analysis_Using_Mobile_Agent.pdf (Accessed 3 April 2018).

Etienne, A. S. and Jeffery, K. J. 2004. Path integration in mammals. *Hippocampus*, 14: 180–192. Available: https://onlinelibrary.wiley.com/doi/epdf/10.1002/hipo.10173 (Accessed 3 April 2018).

Etienne, A. S., Maurer, R. and Saucy, F. 1988. Limitations in the assessment of path dependent information. *Behavior*, 106: 81–111. DOI: 10.1163/156853988X00106 (Accessed 22 April 2017).

Ferchichi, S. E., Laabidi, K. and Zidi, S. 2009. Feature selection using an SVM learning machine. In: *3rd International Conference on Signals, Circuits and Systems (SCS). IEEE Xplore.* DOI: 10.1109/ICSCS.2009.5412341 (Accessed 5 September 2018).

Fister, I. Jr., Fong, S., Bresta, J. and Fister, I. 2014. Towards the self-adaptation of the bat algorithm. In: *Proceedings of the IASTED International Conference, Innsbruck, Austria: Artificial Intelligence and Applications*. pp. 1–7. Available: http://www.iztok-jr-fister.eu/static/publications/36.pdf (Accessed 10 April 2017).

Fong, S., Yang, X.-S. and Deb, S. 2013. How meta-heuristic algorithms contribute to deep learning in the hype of big data analytics. *Progress in Intelligent Computing Techniques: Theory, Practice, and Applications*, 3–25. DOI:10.1007/978-981-10-3373-5_1 (Accessed 6 January 2018).

Friedman, M. 1937. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200): 675–701. Available: http://sci2s.ugr.es/keel/pdf/algorithm/articulo/1937-JSTOR-Friedman.pdf. (Accessed 10 April 2017).

Friedman, M. A. 1940. A comparison of alternative tests of significance for the problem of m rankings. *Annals of Mathematical Statistics*, 11(1): 86–92. Available: https://projecteuclid.org/download/pdf_1/euclid.aoms/1177731944 (Accessed 20 May 2017).

Fung, B. C. M., Wang, K. and Liu, J. 2012. Direct discovery of high utility itemsets without candidate generation. In: *IEEE 12th International Conference on Data Mining.* pp. 1–6. Available: DOI: 10.1109/ICDM.2012.20 (Accessed 20 May 2017).

Gali, N., Mariescu-Istodor, R. and Fränti, P. 2016. Similarity measures for title matching. In: *23rd International Conference on Pattern Recognition (ICPR), Cancún Center, Cancún, Mexico*. Available: http://cs.uef.fi/sipu/pub/TitleSimilarity-ICPR.pdf (Accessed 3 September 2017).

García, S., Fernández, A., Benítez, A. D. and Herrera, F. 2007. Statistical comparisons by means of non-parametric tests: A case study on genetic based machine learning. In: *II Congreso Español de Informática*. pp. 1–10 Available: http://www.lsi.us.es/redmidas/CEDI07/%5B9%5D.pdf (Accessed 3 May 2017).

Garcia, S., Luengo, J. and Herrera, F. 2015. *Data preprocessing in data mining*.

Switzerland: Springer. Available: DOI: 10.1007/978-3-319-10247-4_3 (Accessed 3 May 2017).

García, S., Molina, D. Lozano, M. and Herrera, F. 2008. A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: A case study on the CEC'2005 Special Session on Real Parameter Optimization. *Heuristics*. 15: 617–644. Available: https://link.springer.com/content/pdf/10.1007%2Fs10732-008-9080-4.pdf (Accessed 3 July 2018).

Ghosh, A. and Nath, B. 2004. Muti-objective rule mining using genetic algorithms. *Information Sciences*, 163: 123–133. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.98.6291&rep=rep1&type=pdf (Accessed 3 July 2018).

Golani, I., Benjamini, Y. and Eilam, D. 1993. Stopping behavior: Constraints on exploration in rats (Rattus norvegicus). *Behav Brain Res*, 53: 21–33. Available: https://www.sciencedirect.com/science/article/pii/S0166432805802633?via%3Dihub (Accessed 3 July 2018).

Goldberg, D. 1986**.** Genetic algorithms in search, optimization and machine learning. Boston: Addison-Wesley Longman. Available: https://pdfs.semanticscholar.org/899d/b253ce11bcfa29cacb9363d60a1639fd1fe8.pdf (Accessed 5 September 2018).

Goldberg, D. E. and Holland, J. H. 1988. Genetic algorithms and machine learning. *Machine learning*, 3(2): 95–99. Available: https://link.springer.com/content/pdf/10.1007%2FBF00113892.pdf (Accessed 3 July 2018).

Gordon, S. P. and Gordon, F. S. 1994. *Contemporary statistics: A computer approach*. USA: McGraw-Hill.

Grzymala-Busse, J. W., Goodwing, L. K., Grzymala-Busse, W. J. and Zheng, X. 2005. Handling missing attribute values in preterm birth data sets. Available: https://pdfs.semanticscholar.org/29e5/317f5eb70eff4a6b9bde64fcded211c48bdb.pdf (Accessed 3 July 2018).

Guntsch, M. and Middendorf, M. 2002. Applying population based ACO to dynamic optimization problems. *Ant Algorithms, Proceedings of Third International Workshop ANTS*, 2463: 111–122. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.12.6580&rep=rep1&type=pdf (Accessed 3 July 2018).

Guo, X., Wang, H. and Devabhaktuni, V. 2011. Design of a FPGA-based parallel architecture for blast algorithm with multi-hits detection. In: *Eighth International Conference on Information Technology: New Generations*. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5945320 (Accessed 3 July 2017).

Gupta, M. K. and Sikka, G. 2013. Association rules extraction using multi-objective feature of genetic algorithm. In: *Proceedings of the World Congress on Engineering and Computer Science*. San Francisco. pp. 1–6. Available: https://pdfs.semanticscholar.org/57fa/7fbace0d5ce249a721795095945fb09e2ebb.pdf (Accessed 3 July 2018).

Hall, M. A. 2000. Correlation-based feature selection for discrete and numeric class machine learning. In: *Proc. 17th Int'l Conf. Machine Learning*. pp. 359–366. Available: https://researchcommons.waikato.ac.nz/bitstream/handle/10289/1024/uow-cs-wp-2000-08.pdf?sequence=1&isAllowed=y (Accessed 3 July 2018).

Hamming, R. W. 1950. Error detecting and error correcting codes (PDF). *Bell System Technical Journal*, 29(2): 147–160. Available: DOI: 10.1002/j.1538-7305.1950.tb00463.x. MR 0035935. (Accessed 3 July 2017).

Han, J. and Kamber, M. 2006. *Data mining concepts and techniques*. 3rd ed. USA: Morgan Kaufmann. Available: http://myweb.sabanciuniv.edu/rdehkharghani/files/2016/02/The-Morgan-Kaufmann-Series-in-Data-Management-Systems-Jiawei-Han-Micheline-Kamber-Jian-Pei-Data-Mining.-Concepts-and-Techniques-3rd-Edition-Morgan-Kaufmann-2011.pdf (Accessed 3 July 2018).

Han, J., Cheng, H., Xin, D. and Yan, X. 2007. Frequent pattern mining: Current status and future directions. *Data Min Knowl Disc*, 15: 55–86. Available: https://link.springer.com/content/pdf/10.1007%2Fs10618-006-0059-1.pdf (Accessed 3 July 2018).

Han, J., Pei, J. and Yu. Y. 2000. Mining frequent patterns without candidate generation. In: *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, 29(2): 1–12. DOI: 10.1145/335191.335372 (Accessed 5 September 2018).

Han, J., Pei, J., Mortazavi-Asl, B., Chen, Q., Dayal, U. and Hsu, M. C. 2000. FreeSpan: Frequent pattern projected sequential pattern mining. In: *Proc. ACM SIGKDD International Conference: Knowledge Discovery and Data Mining (SIGKDD'00)*. Boston. pp. 355–359. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.12.7211&rep=rep1&type=pdf (Accessed 3 July 2018).

Han, J., Wang, J., Lu Y. and Tzvetkov, P. 2002. Mining top-K frequent closed patterns without minimum support. In: *Proceedings: 2002 IEEE International Conference on Data Mining*. pp. 211–218. Available:

https://experts.illinois.edu/en/publications/mining-top-k-frequent-closed-patterns-without-minimum-support (Accessed 6 May 2016).

Hand, D., Mannila, H. and Smyth, P. 2001. *Principles of data mining*. London: The MIT Press. Available: https://doc.lagout.org/Others/Data%20Mining/Principles%20of%20Data%20Mining%20%5BHand%2C%20Mannila%20%26%20Smyth%202001-08-01%5D.pdf (Accessed 9 May 2017).

Hansen, N. and Ostermeier, A. 2001. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2): 159–195. Available: http://www.cmap.polytechnique.fr/~nikolaus.hansen/cmaartic.pdf (Accessed 3 July 2018).

Hartigan, J. A. and Wong M. A. 1979. A k-means clustering algorithm. Available: Https://www.labri.fr/perso/bpinaud/userfiles/downloads/hartigan_1979_kmeans.pdf (Accessed 10 April 2017).

Hasan, S., Shamsuddin, S. and Lopes, N. 2013. Soft computing methods for big data problems. In: Cai, Y. and See, S. (eds.), *Proceedings of the symposium on GPU computing and applications*. Singapore: Springer. pp. 235–247. Available: DOI: 10.1007/978-981-287-134-3_15 (Accessed 3 July 2018).

Hernandez, M. A. and Stolfo, S. J. 1995. The merge/purge problem for large databases. In: *Proc. ACM SIGMOD International Conference on Management of Data*, *San Jose, California*, 24(2): 127–138. Available: DOI: 10.1145/223784.223807 (Accessed 13 August 2018).

Hernandez, M. A. and Stolfo, S. J. 1998. Real-world data is dirty: Data cleansing and the merge/purge problem. *Data Mining and Knowledge Discovery*, 2(1): 9–37. Available: https://link.springer.com/content/pdf/10.1023%2FA%3A1009761603038.pdf

(Accessed 13 August 2018).

Hinton, P. R. 1995. *Statistics explained: A guide for social science students*. Padstow, Cornwall: T. J. Press.

Hirate, Y., Iwahashi, E. and Yamana, H. 2004. TF^2P-growth: An efficient algorithm for mining frequent patterns without any threshold. In: *Proceedings of IEEE ICDM'04: Workshop on alternative techniques for data mining and knowledge discovery*. Available: https://pdfs.semanticscholar.org/22d5/b3362a9d0923687e1d88defb9f5b334e539c.pdf (Accessed 5 September 2018).

Holland, J. 1975. *Adaptation in natural and artificial systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. Ann Arbor, MI: University of Michigan Press.

Hong, T.-P. and Lee, Y.-C. 2008. *An overview of mining fuzzy association rules, fuzzy sets and their extensions: Representation, aggregation and models*. Springer. Available: https://pdfs.semanticscholar.org/a7a5/1bc8cc9cf76e03e9a7a66aa80bda84384c56.pdf (Accessed 13 August 2018).

Hopcroft, J. E. and Ullman, J. D. 1973. Set merging algorithms. *SIAM Journal on Computing*, 2: 294–303. Available: https://doi.org/10.1137/0202024 (Accessed 13 August 2018).

Hornik, K. 1991. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4: 251–257. Available: https://www.sciencedirect.com/science/article/pii/089360809190009T (Accessed 13 August 2018).

Huai, Y., Lee, R., Zhang, S., Xia, C. H. and Zhang X. 2011. DOT: A matrix model for analyzing, optimizing and deploying software for big data analytics in distributed systems. *Proceedings of the ACM symposium on cloud computing*, 4: 1–14. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.228.8651&rep=rep1&type=pdf (Accessed 13 August 2018).

Huang, J. W., Lin, S. C. and Chen, M. S. 2010. DPSP: Distributed sequential pattern mining on the cloud. Advances in Knowledge Discovery and Data Mining, 6119: 27–34. Available: DOI: 10.5120/16461-6187 (Accessed 13 August 2018).

Huang, K., Chang, C., Tung, J. and Ho, C. 2006. COBRA: Closed sequential pattern mining using bi-phase reduction approach. Available: https://staff.csie.ncu.edu.tw/chia/pub/cobralncs.pdf (Accessed 13 August 2018).

Huynh, X.-H. 2010. Interestingness measures for association rules in a KDD Process: Post Processing of Rules with ARQAT Tool. Available: https://tel.archives-ouvertes.fr/tel-00482649/document (Accessed 13 August 2018).

Iglesia, B. and Reynolds. A. 2005. The use of meta-heuristic algorithms for data mining. Available: DOI: 10.1109/ICICT.2005.1598541 (Accessed 13 August 2018).

Ilyankou, I. 2014. Comparison of jaro-winkler and ratcliff/obershelp algorithms in spell check. Available: https://ilyankou.files.wordpress.com/2015/06/ib-extended-essay.pdf (Accessed 3 May 2017).

Jaitly, N. 2014. Exploring deep learning methods for discovering features in speech signals. Ph.D. Thesis in Computer Science. Department of Computer Science, University of Toronto. Available:

http://www.cs.toronto.edu/~ndjaitly/Jaitly_Navdeep_201411_PhD_thesis.pdf (Accessed 6 January 2018).

Janecek, A. G. K. and Gansterer, W. N. 2008. On the relationship between feature selection and classification accuracy. *JMLR: Workshop and Conference Proceedings*, 4: 90–105. Available: http://proceedings.mlr.press/v4/janecek08a/janecek08a.pdf (Accessed 3 May 2017).

Jaro, M. A. 1989. Advances in record-linkage methodology as applied to matching the 1985 census of tampa, Florida. *J. Am. Statistical Assoc*., 84(406): 414–420. Available: DOI: 10.1080/01621459.1989.10478785 (Accessed 13 August 2018).

Jaro, M. A. 1995. Probabilistic linkage of large public health data files (disc: P687-689). *Statistics in Medicine*, 14: 491–498. Available: https://doi.org/10.1002/sim.4780140510 (Accessed 13 August 2018).

Kannan, V. and Gurusamy, S. 2014. Preprocessing techniques for text mining. Available: https://www.researchgate.net/publication/273127322_Preprocessing_Techniques_for_Text_Mining (Accessed 13 August 2018).

Kantardzic, M. 2003. Data mining: Concepts, models, methods and algorithm. *IEEE*, 165–176. Available: https://search.proquest.com/docview/213699988/fulltextPDF/DBBF81E4E33548E7PQ/1?accountid=10612 (Accessed 13 August 2018).

Karaboga, D. 2005. *An ideal based on honey bee swarm for numerical optimization technical report*. Available: https://pdfs.semanticscholar.org/015d/f4d97ed1f541752842c49d12e429a785460b.pdf (Accessed 13 August 2018).

Kaytoue, M., Kuznetsov, S. O. and Napoli, A. 2011. Revisiting numerical pattern mining with formal concept analysis. Higher school of economics, State University. Russia. 1–10. DOI:10.5591/978-1-57735-516-8/IJCAI11-227 (Accessed 5 September 2018).

Ke, K., Cheng J. and Ng W. 2008. An information theoretic approach to quantitative association rule mining. *Journal Knowledge and Information Systems*, 16(2): 112–114. Available: http://www.cs.ust.hk/faculty/wilfred/paper/kais07.pdf (Accessed 13 August 2018).

Keim, D. 2000. Designing pixel-oriented visualization techniques: Theory and applications. *IEEE Transaction on Visualization and Computer Graphics*, 6(1): 59–78. Available: http://kops.uni-konstanz.de/bitstream/handle/123456789/5890/TVCG00.pdf (Accessed 23 June 2018).

Keim, D. A. 2002. Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics*, 8(1). Available: DOI: 10.1109/2945.981847 (Accessed 23 June 2018).

Keim, D. A., Bergeron, R. D. and Pickett, R. M. 1994. Test data sets for evaluating data visualization techniques. Available: https://pdfs.semanticscholar.org/7959/fd04a4f0717426ce8a6512596a0de1b99d18.pdf (Accessed 3 February 2017).

Kennedy, J. 2011. Particle swarm optimisation. In: *Encyclopedia of machine learning*. Springer. pp. 760–766. Available: https://pdfs.semanticscholar.org/9655/b25a85ea2fd1b50cd9eb3c4e298aa15bb012.pdf (Accessed 23 June 2018).

Kennedy, J. and Eberhart, R. C. 1995. Particle swarm optimization. In: *Proceedings of IEEE International Conference on Neural Networks, Piscataway, NJ*. pp. 1942–1948. Available: https://www.cs.tufts.edu/comp/150GA/homeworks/hw3/_reading6%201995%20particle%20swarming.pdf (Accessed 23 June 2018).

Kim, J. W. 2013. Classification with deep belief networks. Available: https://www.ki.tu-berlin.de/fileadmin/fg135/publikationen/Hebbo_2013_CDB.pdf (Accessed 3 May 2017).

Kline, R. B. 1998. *Principles and practices of structural equation modeling*. New York: Guilford. Available: ftp://158.208.129.61/suzuki/PP_SEM_3e.pdf (Accessed 23 June 2018).

Kohavi, R. and John, G. H. 1996. Wrappers for feature subset selection. *AIJ Special Issue on Relevance*. Available: http://www-ai.cs.uni-dortmund.de/LEHRE/PG/PG445/literatur/kohavi_john_97a.pdf (Accessed 23 June 2018).

Kołcz, A. and Chowdhury, A. 2008. Lexicon randomization for near-duplicate detection with I-Match. *Journal of Supercomputing*, 45: 255–276. Available: https://link.springer.com/content/pdf/10.1007%2Fs11227-007-0171-z.pdf (Accessed 23 June 2018).

Kondrak, G. N. 2005. Gram similarity and distance. Department of Computing Science, University of Alberta, Canada. Available: https://pdfs.semanticscholar.org/5cfb/028595fff3e550c9a5fd2a51e4f23b4b58b6.pdf (Accessed 3 September 2017).

Khana, D. M., Mohamudally, N. and Babajee, D. K. R. 2013. A Unified Theoretical Framework for Data Mining. Information Technology and

Quantitative Management. *Procedia Computer Science,* 17(2013): 104 – 113. Available: doi:10.1016/j.procs.2013.05.015 (Accessed 14 November 2018).

Kotsiantis, S. B. 2007. Supervised machine learning: A review of classification techniques. *Informatica*, (31): 249–268. Available: https://datajobs.com/data-science-repo/Supervised-Learning-[SB-Kotsiantis].pdf (Accessed 5 September 2018).

Krause, J., Cordeiro, J., Parpinelli, R. S. and Lopes, H. S. 2013. A survey of swarm algorithms applied to discrete optimization problems. Available: DOI: 10.1016/B978-0-12-405163-8.00007-7 (Accessed 23 June 2018).

Krizhevsky, A., Sutskever, I. and Hinton, G. E. 2012. ImageNet classification with deep convolutional neural networks. Available: https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf (Accessed 5 September 2018).

Kumar, R. 2015. Grey wolf optimizer (GWO). Electrical Engineering MNIT Jaipur. Available: https://drrajeshkumar.files.wordpress.com/2015/05/wolf-algorithm.pdf. (Accessed 3 September 2017).

Kumar, V. and Minz, S. 2014. Feature selection: A literature review. *Smart Computing Review*, 4(3): 211–229. Available: https://www.cc.gatech.edu/~hic/CS7616/Papers/Kumar-Minz-2014.pdf (Accessed 23 June 2018).

Kumar, V., Xindong, W., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G. J., Ng, A., Liu, B., Yu, P. S., Zhou, Z. H., Steinbach, H., Hand, D. J., Steinberg, D. 2007. Top 10 algorithms in data mining. *Knowledge Information System*, 14: 1–37. Available: https://link.springer.com/content/pdf/10.1007%2Fs10115-007-0114-2.pdf (Accessed 23 June 2018).

Kuo, R. J. and Shih, C. W. 2007. Association rule mining through the ant colony system for national health insurance research database in Taiwan. *Computers and Mathematics with Applications*, 54: 1303–1318. Available: https://www.sciencedirect.com/science/article/pii/S0898122107000910 (Accessed 3 September 2017).

Kuo, R. J., Chao, C. M. and Chiu, Y. T. 2011. Application of particle swarm optimization to association rule mining. *Applied Soft Computing*, 11: 326–336. Available: DOI: 10.1016/j.asoc.2009.11.023 (Accessed 23 June 2018).

Lakshminarayan, K., Harp, S. A. and Samad, T. 1999. Imputation of missing data in industrial databases. *Applied Intelligence*, 11: 259–275. Available: https://link.springer.com/content/pdf/10.1023%2FA%3A1008334909089.pdf (Accessed 10 April 2017).

Laney, D. 2001. 3D data management: Controlling data volume, velocity and variety. Available: http://smbresearch.net/big-data-and-volume-velocity-and-variety/ (Accessed 10 April 2017).

Laurila, J. K., Gatica-Perez, D., Aad, I., Blom, J., Bornet, O., Do, T., Dousse, O., Eberle, J. and Miettinen, M. 2012. The mobile data challenge: Big data for mobile computing research. In: *Proceedings of the mobile data challenge by Nokia workshop*. Infoscience EPFL scientific publication. pp. 1–8. Available: https://pdfs.semanticscholar.org/8dae/ecc84fcaf42172cba7ef58e5068fae7bbcbc.pdf (Accessed 10 April 2017).

Le, Q. V. 2015. A tutorial on deep learning. Part 1: Nonlinear classifiers and the backpropagation algorithm. Available: https://cs.stanford.edu/~quocle/tutorial1.pdf

LeCun, Y., Bengio, Y. and Hinton, G. 2015. Review: Deep learning. *Nature*, 521: 436–444. Available:

https://www.evl.uic.edu/creativecoding/courses/cs523/slides/week3/DeepLearning_LeCun.pdf

LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P. 1998. Gradient-based learning applied to document recognition. In: *Proceedings of the IEEE*. pp. 1–46. Available: http://vision.stanford.edu/cs598_spring07/papers/Lecun98.pdf

Lee, H., Grosse, R., Ranganath, R. and Ng, A. Y. 2009. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In: *Proceedings of the 26th International Conference on Machine Learning, Montreal, Canada*. pp. 1–8. Available: https://web.eecs.umich.edu/~honglak/icml09-ConvolutionalDeepBeliefNetworks.pdf (Accessed: 05-September-2018).

Lee, H., Largman, Y., Pham, P. and Ng. A. Y. 2009. Unsupervised feature learning for audio classification using convolutional deep belief networks. Computer Science Department, Stanford University, Stanford. Available: https://ai.stanford.edu/~ang/papers/nips09-AudioConvolutionalDBN.pdf (Accessed 5 September 2018).

Lee, J., Hong, S. and Lee, J. H. 2014. An efficient prediction for heavy rain from big weather data using genetic algorithm. *Proceedings of the international conference on ubiquitous information management and communication*. 25: 1–7. Available: http://delivery.acm.org/10.1145/2560000/2558048/a25-lee.pdf?ip=196.21.61.167&id=2558048&acc=ACTIVE%20SERVICE&key=646D7B17E601A2A5%2E20146AEDC3D229CC%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&__acm__=1535539638_39e14e83d0d97ed78bad3cae6c53ca89 (Accessed 10 April 2017).

Lee, K. Y. and Park, J.-B. 2006. Application of particle swarm optimization to economic dispatch problem: Advantages and disadvantages. Available: DOI: 10.1109/PSCE.2006.296295 (Accessed 10 April 2016).

Leke, C. and Marwala, T. 2016. Missing data estimation in high-dimensional datasets: A swarm intelligence-deep neural network approach. Available: https://arxiv.org/pdf/1607.00136.pdf (Accessed 3 May 2017).

Leke, C. and Marwala, T. 2016. Missing data estimation in high-dimensional datasets: A swarm intelligence-deep neural network approach. University of Johannesburg, Johannesburg, South Africa. Available: https://link.springer.com/chapter/10.1007/978-3-319-41000-5_26 (Accessed 10 April 2016).

Leung, C. K., Kononov, V. V., Pazdor, A. G. M. and Jiang, F. 2016. PyramidViz: Visual analytics and big data visualization of frequent patterns. In: *IEEE 14th International Conference on Dependable, Autonomic and Secure Computing, 14th International Conference on Pervasive Intelligence and Computing, 2nd International Conference on Big Data Intelligence and Computing and Cyber Science and Technology Congress*. Auckland. pp. 913–916. Available: DOI: 10.1109/DASC-PICom-DataCom-CyberSciTec.2016.158 (Accessed 10 April 2016).

Leung, C. S., MacKinnon, R. and Jiang, F. 2014. Reducing the search space for big data mining for interesting patterns from uncertain data. In: *Proceedings of the international congress on big data*. pp. 315–322. Available: DOI: 10.1109/BigData.Congress.2014.53 (Accessed 10 April 2016).

Levenshtein, V. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics – Doklady10*. 10: 707–710. Available: https://nymity.ch/sybilhunting/pdf/Levenshtein1966a.pdf (Accessed 3 April 2018).

Li, J., Fong, S., Wong, R. K., Millham, R. and Wong, K. K. L. 2017. Elitist binary wolf search algorithm for heuristic feature selection in high-dimensional bioinformatics datasets. Available: DOI: 10.1038/s41598-017-04037-5 (Accessed 20 April 2018).

Lichman, M. 2013. *UCI machine learning repository*. Irvine, CA: University of California, School of Information and Computer Science. Available: http://archive.ics.uci.edu/ml (Accessed 10 May 2017).

Lin, C.-J. 2006. Support vector machines: Status and challenges. Available: https://www.csie.ntu.edu.tw/~cjlin/talks/caltech.pdf (Accessed 3 May 2017).

Lin, J. C.-W, Yang, L., Fournier-Viger, P., Hong, T.-P. and Voznak, M. 2016. A binary PSO approach to mine high-utility itemsets. *Soft Computing*, 1–19. Available: http://www.philippe-fournier-viger.com/spmf/HUIMGATree.pdf (Accessed 3 August 2017)

Lin, M. Y., Lee, P. Y. and Hsueh, S. C. 2012. Apriori-based frequent itemset mining algorithms on mapreduce. *Proceedings of the International Conference on Ubiquitous Information Management and Communication*, 76: 1–76:8. Available: http://delivery.acm.org/10.1145/2190000/2184842/a76-lin.pdf?ip=196.21.61.167&id=2184842&acc=ACTIVE%20SERVICE&key=646D7B17E601A2A5%2E20146AEDC3D229CC%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&__acm__=1535543138_1f233e1e8363128bdb908f5d9e68e9bb (Accessed 3 May 2016).

Lin, M., Hsueh, S. and Chang, C. 2008. Mining closed sequential patterns with time constraints. *Journal of Information Science and Engineering*, 24: 33–46. Available: https://www.iis.sinica.edu.tw/page/jise/2008/200801_03.pdf (Accessed 3 May 2016).

Lipman, D. J and Pearson, W. R. 1985. Rapid and sensitive protein similarity searches. *Science Signaling*, 227(4693): 1435–1441. DOI: 10.1126/science.2983426

Little, R. J. A and Rubin, D. B. 1989. The analysis of social science data with missing values. *Sociol Methods Res*, 18: 292–326.

Little, R. J. A. and Rubin, D. B. 1987. *Statistical analysis with missing data*. New York: Wiley.

Liu, C., Wang, W., Zhao, Q., Shen, X. and Konan, X. 2017. A new feature selection method based on a validity index of feature subset. *Pattern Recognition Letters*, 92: 1–8. Available: DOI: 10.1016/j.patrec.2017.03.018 (Accessed 3 May 2017).

Longbottom, C. and Bamforth, R. 2013. "Optimising the data warehouse": Dealing with large volumes of mixed data to give better business insights. *Quocirca*. Available: https://www.hindawi.com/journals/mpe/2018/9457821/ (Accessed 3 May 2018).

Luke, S. 2015. Essentials of metaheuristics. Department of Computer Science, George Mason University. Available: https://cs.gmu.edu/~sean/book/metaheuristics/Essentials.pdf (Accessed 3 May 2017).

Mafarja, M. and Mirjalili, S. 2018. Whale optimization approaches for wrapper feature selection. Applied Soft Computing. Available: https://doi.org/10.1016/j.asoc.2017.11.006 (Accessed 20/04/2018).

Malewicz, G., Austern, M. H., Bik, A. J., Dehnert, J. C., Horn, I., Leiser, N. and Czajkowski, G. 2010. Pregel: A system for large-scale graph processing. In: *Proceedings of the ACM SIGMOD international conference on management of data*, 2010: 135–146. Available: https://kowshik.github.io/JPregel/pregel_paper.pdf (Accessed 20 April 2018).

Mamduh, S. M., Kamarudin, K., Shakaff, A. Y. M., Zakaria, A. and Abdullah,

A. H., 2014. Comparison of Braitenberg vehicles with bio-inspired algorithms for odor tracking in laminar flow. *NSI Journals: Australian Journal of Basic and Applied Sciences*, 8(4): 6–15. Available: https://pdfs.semanticscholar.org/afc5/471733de0feef22a18564b85c13449ce5553.pdf (Accessed 20 April 2018).

Mangat, V. 2010. Swarm intelligence based technique for rule mining in the medical domain. *International Journal of Computer Applications*, 4(1): 19–24. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.206.3011&rep=rep1&type=pdf (Accessed 20 April 2018).

Marcus, G. 2018. Deep learning: A critical appraisal. *Artificial Intelligence*, 1–27. Available: https://arxiv.org/abs/1801.00631 (Accessed 3 May 2017).

Marghescu, D. 2008. Evaluating multidimensional visualization techniques in data mining tasks. Department of Information Technologies, Åbo Akademi University, Turku Centre for computer science. Turku, Finland. Available on: http://www.doria.fi/bitstream/handle/10024/69974/MarghescuDorina.pdf?sequence=3&isAllowed=y (Accessed 2 April 2018).

Marill, D. G. T. 1963. On the effectiveness of receptors in recognition systems. *IEEE Transactions on Information Theory*, 9(1): 11–17. Available: DOI: 10.1109/TIT.1963.1057810 (Accessed 2 April 2018).

Marwala, T. 2006. Computational intelligence for missing data imputation, estimation, and management: Knowledge optimization techniques. University of Witwatersrand, South Africa. Available: https://scholar.google.co.za/scholar?q=Marwala,+T.+2006.+Computational+intelligence+for+missing+data+imputation,+estimation,+and+management:+Knowledge+optimization+Techniques.&hl=en&as_sdt=0&as_vis=1&oi=scholart

(Accessed 2 April 2018).

Marwala, T. 2006. Probabilistic fault identification using vibration data and neural networks. *Mechanical systems and signal processing*, 15(6): 1109–1128. Available: http://www.tshilidzimarwala.com/imac3.pdf

Marwala, T. 2007. Bayesian training of neural networks using genetic programming. Available: DOI: 10.1016/j.patrec.2007.03.004 (Accessed 3 May 2017).

MathWorks. 2017. Documentation: The language of technical computing. Available: https://www.mathworks.com/tagteam/73554_91199v02_overview.pdf

Matsakis, N. E. 2010. Active duplicate detection with bayesian nonparametric models. PhD Thesis. Massachusetts Institute of Technology. Available on: https://pdfs.semanticscholar.org/b8b2/e325a2f25d9f0603c59be3c81558a0475ba4.pdf (Accessed 3 May 2017).

Mirjalili, S., Mirjalili, M. S. and Lewis, A. 2014. Grey wolf optimizer. *Advances in Engineering Software*, 69: 46–61. Available: Mittelstaedt, H. and Mittelstaedt, M.-L. 1982. Homing by path integration. In: Papi, F., Wallraff, H. G. eds. *Avian navigation*. New York: Springer, 290–297. Available: https://link.springer.com/chapter/10.1007/978-3-642-68616-0_29 (Accessed 20 May 2018).

Moere, A. V. 2004. Time-varying data visualization using information flocking boids. In: *IEEE Symposium on Information Visualization. Austin, USA*. pp. 1–8. Available: DOI: 10.1109/INFVIS.2004.65 (Accessed 20 May 2018).

Moere, A. V., Clayden, J. J. and Dong, A. 2006. *Data clustering and visualization using cellular automata ants*. Berlin and Heidelberg: Springer-Verlag. Available: https://link.springer.com/chapter/10.1007/11941439_87

(Accessed 20 May 2018).

Mong, L. L., Hongjun, L., Ling, T. W. and Ko, Y. T. 2002. Cleansing data for mining and warehousing. In: *Database and Expert Systems Applications*: *International Conference on Database and Expert Systems Applications*. pp. 751–760. Available: https://pdfs.semanticscholar.org/7a81/f7151b83b54b09ad2861aee52e5baa16fd33.pdf (Accessed 20 May 2018).

Monge, A. 2000. An adaptive and efficient algorithm for detecting approximately database records: Bulletin of the Technical Committee. *IEEE Computer Society Letters*, 23. Available: https://www.researchgate.net/profile/Alvaro_Monge/publication/2409102_An_Adaptive_and_Efficient_Algorithm_for_Detecting_Approximately_Duplicate_Database_Records/links/00b7d521d421658ea2000000/An-Adaptive-and-Efficient-Algorithm-for-Detecting-Approximately-Duplicate-Database-Records.pdf (Accessed 20 May 2018).

Monge, A. and Elkan, C. 1997. An efficient domain-independent algorithm for detecting approximately duplicate database records. In: *Proceedings of the ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery*. pp. 23–29. Available: https://pdfs.semanticscholar.org/405c/1614a009431a4d3eaa5b6b023055403aafe1.pdf (Accessed 20/05/2018).

Monge, A. E. 1997. Adaptive detection of approximately duplicate database records and the database integration approach to information discovery. Ph.D. Thesis in Computer Science and Engineering, Department of Computer Science and Engineering. University of California, San Diego. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.22.3368&rep=rep1&type=pdf (Accessed 20 May 2018).

Morris, G., Farnum, G., Afzal, S., Robinson, C., Greene, J. and Coughlin. C. 2014. Patient identification and matching final report. 1-93. Available: http://journal.ahima.org/wp-content/uploads/ONC-Patient-Identification-Matching-Final-Report-February-2014.pdf (Accessed 3 May 2016).

Moslehi, P., Bidgoli, B. M., Nasiri, M. and Salajegheh, A. 2011. Multi-objective numeric association rules mining via ant colony optimization for continuous domains without specifying minimum support and minimum confidence. *IJCSI International Journal of Computer Science*, 8(5): 34–41. Available: https://pdfs.semanticscholar.org/1dd2/59bdeffaaa0b343b9518845f4a90effe9b46.pdf (Accessed 20 May 2018).

Muro, C., Escobedo, R., Spector, L., Coppinger, R. 2011. Wolf-pack (Canis lupus) hunting strategies emerge from simple rules in computational simulations. *Behav Process*, 201(88): 192–197. Available: DOI: 10.1016/j.beproc.2011.09.006 (Accessed 20 May 2018).

Narang, R. K. 2013. *Inside the black box: A simple guide to quantitative and high frequency trading*. 2nd ed. USA: John Wiley and Sons. Available: https://leseprobe.buch.de/images-adb/78/04/78041046-b4fd-4cae-b31d-3cb2a2e67301.pdf (Accessed 20 May 2018).

Naumann, F. 2013. Duplicate detection. *Information Systems Group*. Hasso Plattner Institut, Universitat Potsdam. 1-52. Available: https://hpi.de/fileadmin/user_upload/fachgebiete/naumann/folien/SS13/DPDC/DPDC_11_Duplicate_Detection.pdf (Accessed 3 May 2017).

Naumann, F. and Herschel, M. 2010. An introduction to duplicate detection: synthesis lectures on data management. *Synthesis Lectures on Data Management*, 2(1): 1–87. Available: https://doi.org/10.2200/S00262ED1V01Y201003DTM003 (Accessed 3 May

2017).

Nelwamondo, F. V., Mohamed, S. and Marwala, T. 2007. Missing data: A comparison of neural network and expectation maximisation techniques. School of Electrical and Information Engineering, University of the Witwatersrand, South Africa. 1-24. Available: https://arxiv.org/ftp/arxiv/papers/0704/0704.3474.pdf (Accessed 10 August 2017).

Nolan, R. L. 1979. Managing the crises in data processing. *Harvard Business Review*, 57(1): 115–126. Available: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-015-0030-3 (Accessed 25 August 2018).

Osman, I. H. and Laporte, G. 1996. Metaheuristics: A bibliography. *Annals of Operational Research*, 63: 513–623. Available: https://s3.amazonaws.com/academia.edu.documents/46309065/Metaheuristics_A_Bibliography20160607-17999-wc81c1.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1535615211&Signature=Ew2Jz0mbjSq2G1A8VjSAQUIi1WE%3D&response-content-disposition=inline%3B%20filename%3DMetaheuristics_A_bibliography.pdf (Accessed 25 August 2018).

Oweis, N. E., Fouad, M. M, Oweis, S. R., Owais, S. S. and Snasel, V. 2016. A novel mapreduce lift association rule mining algorithm (mrlar) for big data. *International Journal of Advanced Computer Science and Applications (IJACSA)*, 7(3): 151–157. Available: DOI: 10.14569/IJACSA.2016.070321 (Accessed 25 August 2018).

Panda, S. K., Nag, S. and Jana, P. K. 2014. A smoothing based task scheduling algorithm for heterogeneous multi-cloud environment. In: *3rd IEEE International Conference on Parallel, Distributed and Grid Computing (PDGC*). Waknaghat: IEEE. pp. 62–67. Available: DOI: 10.1109/PDGC.2014.7030716 (Accessed 20 April 2018).

Patel, A. B., Nguyen, T. and Baraniuk, R. G. 2015. A probabilistic theory of deep learning. Department of Electrical and Computer Engineering, Rice University. 1-56. Available: https://arxiv.org/pdf/1504.00641v1.pdf (Accessed 5 September 2018).

Pearson, W. 1991. Searching protein sequence libraries: Comparison of the sensitivity and selectivity of the Smith-Waterman and FASTA algorithms. *Genomics*, 11(3): 635–650. Available: https://www.sciencedirect.com/science/article/pii/088875439190071L?via%3Dihub (Accessed 25 August 2018).

Pearson, W. 1995. Comparison of methods for searching protein sequence databases. *Protein Science*, 4(6): 1145. Available: http://compbio.berkeley.edu/class/c246/Reading/pearson-1995-proteins.pdf (Accessed 25 August 2018).

Pearson, W. R. 2014**.** BLAST and FASTA similarity searching for multiple sequence alignment. *Multiple Sequence Alignment Methods*, 1079: 75–101. Available: https://link.springer.com/protocol/10.1007%2F978-1-62703-646-7_5 (Accessed 5 September 2018).

Pei, J., Han, J., Mortazavi-Asl. B., Wang, J., Pinto, H., Chen, Q., Dayal, U. and Hsu, M. C. 2001. PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth. In: *Proc. of the Int'l Conf on data engineering (ICDE)*. 215–224. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.12.7211&rep=rep1&type=pdf (Accessed 3 July 2018).

Penaloza, C. I., Mae, Y., Ohara, K. and Arai, T. 2012. Social human behavior modeling for robot imitation learning. In: *Proceedings of 2012 IEEE International Conference on Mechatronics and Automation*. Chengdu, China. pp. 457–462. Available: https://zapdf.com/social-human-behavior-modeling-for-robot-imitation-learning.html (Accessed 25 July 2018).

Piatetsky-Shapiro, G. 1991. *Discovery, analysis, and presentation of strong rules. Knowledge discovery in databases*. Cambridge: AAAI/MIT Press. Available: https://pdfs.semanticscholar.org/a8cd/b6dd622e1f0d61abaef345a3ddbf9795c02f.pdf (Accessed 3 February 2017).

Priya, R. D. and Kuppuswami, S. 2012. A genetic algorithm based approach for inputing missing discrete attribute values in databases. *WSEAS Transactions on Information Science and Applications*, 9(6): 169–178. Available: https://pdfs.semanticscholar.org/167c/c1cbb5c0a587c95c8444e0a80d961c40a5cb.pdf (Accessed 3 February 2017).

Qodmanan H. R., Nasiri, M., and Minaei-Bidgoli, B. 2010. Multi objective association rule mining with genetic algorithm without specifying minimum support and minimum confidence. *Expert Systems with Applications*. 38(1): 288–298. Available: DOI: 10.1016/j.eswa.2010.06.060 (Accessed 3 February 2017).

Quinlan, J. R. 1989. Unknown attribute values in induction. In: *Proceedings of the Sixth International Workshop on Machine Learning, Ithaca*. NY: Morgan Kaufmann. pp. 164–168. Available: https://pdfs.semanticscholar.org/d1d8/ff72a970d03c37e776f1222051f3fd6c617c.pdf (Accessed 3 February 2017).

Radoop. 2015. *Radoop*. Available: https://rapidminer.com/products/radoop/. (Accessed 2 February 2015).

Rahm, E. and Do, H. H. 2000. Data cleaning: Problems and current approaches. 1-11. Available: https://www.betterevaluation.org/sites/default/files/data_cleaning.pdf (Accessed 3 February 2017).

Railean, I., Lenca, P., Moga, S. and Borda, M. 2013. Closeness preference a new interestingness measure for sequential rules mining. *Knowledge-Based-Systems.* 44: 48–56. Available: DOI: 10.1016/j.knosys.2013.01.025 (Accessed 2 February 2017).

Rajasekaran, S., Nick, H., Pardalos, P. M., Sahni, S. and Shaw, G. 2001. Efficient algorithms for local alignment search. *Journal of Combinatorial Optimization*, 5: 117–124. Available: https://link.springer.com/content/pdf/10.1023/A:1009893719470.pdf (Accessed 2 February 2017).

Raju, V. P. and Varma, G. P. S. 2015. Mining closed sequential patterns in large sequence databases. *International Journal of Database Management Systems (IJDMS)*, 7(1): 29–39. Available: https://pdfs.semanticscholar.org/e8b7/783db07a89917e8fb1a06227e3a9d64b4242.pdf (Accessed 3 July 2017).

Ramya, K. A. and Pushpa, M. 2016. A survey on lossless and lossy data compression methods. *International Journal of Computer Science and Engineering Communications*, 4(1): 1277–1280. Available: https://pdfs.semanticscholar.org/c129/889a27486322f5c9cc27adae78e04f8a0b1b.pdf (Accessed 3 July 2017).

Rebentrost, P., Mohseni, M. and Lloyd, S. 2014. Quantum support vector machine for big feature and big data classification. American Physical Society. Available: http://dblp.uni-trier.de/db/journals/corr/corr1307.html#RebentrostML13 (Accessed 3 May 2017).

Rehman, M. H. U. 2016. Big data reduction methods: A survey. *Data Science and Engineering*, 1(4): 265–284. Available: https://link.springer.com/article/10.1007/s41019-016-0022-0 (Accessed 3 May 2017).

Rehmana, M. H. U., Chang, V., Batool, A. and Waha, T. Y. 2016. Big data reduction framework for value creation in sustainable enterprises. *International Journal of Information Management*, 1–23. Available: https://core.ac.uk/download/pdf/46574748.pdf (Accessed 3 January 2017).

Risden, K. and Czerwinski, M. P. 2000. An initial examination of ease of use for 2d and 3d information visualizations of web content. *Int. J. Human-Computer Studies*, 53: 695–714. Available: DOI:10.1006/ijhc.2000.0413 (Accessed 3 January 2017).

Rouse, M. 2018. Big data analytics. *Search Business Analytics*. Available: whatIs.com (Accessed 9 April 2018).

Rubin D. B. 1977. Formalizing subjective notion about the effect of nonrespondents in sample surveys. *Journal of the American Statistical Association*, 72(359): 538–543.

Rubin, L. H., Witkiewitz, K., Andre, J. St. and Reilly, S. 2007. Methods for handling missing data in the behavioral neurosciences: Don’t throw the baby rat out with the bath water. *Journal of Undergraduate Neuroscience*, 5(2): 1–7. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3592650/ (Accessed 9 April 2018).

Rusu, F. and Dobra, A. 2011. GLADE: A scalable framework for efficient analytics. In: *Proceedings of LADIS workshop held in conjunction with VLDB*. pp. 1–6. Available: https://pdfs.semanticscholar.org/0347/6dd609ad7749a8fe7a9699800842c560e333.pdf (Accessed 9 April 2018).

Sakato, T., Ozeki, M. and Oka, N. 2012. A computational model of imitation and autonomous behavior. In: *13th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*. Japam: IEEE. pp. 13–18. Available: https://link.springer.com/chapter/10.1007/978-3-319-00738-0_4 (Accessed 9 April 2018).

Sakato, T., Ozeki, M. and Oka, N. 2013. Learning which features to imitate in a painting task. In: *Second IIAI International Conference on Advanced Applied Informatics.* USA: IEEE. pp. 379–384. Available: DOI: 10.1109/IIAI-AAI.2013.74 (Accessed 9 April 2018).

Santos, B. S. 2008. Evaluating visualization techniques and tools: What are the main issues? Universidade de Aveiro, Aveiro, Portugal. Available: http://www.dis.uniroma1.it/beliv08/pospap/santos.pdf (Accessed 3 May 2017).

Sarath, K. N. V. D. and Ravi, V. 2013. Association rule mining using binary particle swarm optimization. *Engineering Applications of Artificial Intelligence*. Available: www.elsevier.com/locate/engappai (Accessed 3 May 2017).

Sauleau, E. A., Paumier, J.-P. and Buemi, A. 2005. Medical record linkage in health information systems by approximate string matching and clustering. *BMC Medical Informatics and Decision Making*, 5(32): 1–13. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1274322/ (Accessed 3 March 2017).

Schatz, B., Chameron, S., Beugnon, G. and Collett, T. S. 1999. The use of path integration to guide route learning in ants. *Nature*, 399: 769–772. Available: www.nature.com/articles/21625 (Accessed 10 September 2017).

Schmidhuber, J. 2014. Draft: Deep learning in neural networks: An overview. *Neural Networks*, 61: 85–117. Available: http://www.idsia.ch/~juergen/DeepLearning30April2014.pdf (Accessed 10 February 2018).

Shekhawat, A., Poddar. P. and Boswal. D. 2009. Ant colony optimization algorithms: Introduction and beyond. Artificial Intelligence Seminar, Indian Institute of Technology, Bombay. Available: http://mat.uab.cat/~alseda/MasterOpt/ACO_Intro.pdf (Accessed 10 September 2018).

Shi, Y. and Eberhart, R. 1998. A modified particle swarm optimizer. In: *IEEE International Conference on Evolutionary Computation, Anchorage*. pp. 69–73. Available: http://dx.doi.org/10.1109/icec.1998.699146 (Accessed 5 September 2018).

Shneiderman, B. 2008. Extreme visualization: Squeezing a billion records into a million pixels. In: *ACM Conference on Management of Data (SIGMOD)*. Vancouver: ACM. Available: https://www.cs.umd.edu/~ben/papers/Shneiderman2008Extreme.pdf (Accessed 3 September 2017).

Shpaer, E. G., Robinson, M., Yee, D., Candlin, J. D., Mines, R. and Hunkapiller, T. 1996. Sensitivity and selectivity in protein similarity searches: A comparison of Smith-Waterman in hardware to blast and fasta. *Genomics*, 38(2): 179–191. Available: https://pdfs.semanticscholar.org/8889/2cf73e9ce788158572cb0ecf00742d8e67d1.pdf (Accessed 3 September 2017).

Shrubb, M. 1982. The hunting behaviour of some farmland Kestrels. *Bird Study*, 29: 121–128. Available: https://www.tandfonline.com/doi/pdf/10.1080/00063658209476746 (Accessed 3 September 2016).

Silberschatz, A. and Tuzhilin, A. 1995. On subjective measures of interestingness in knowledge discovery. In: *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD'95)*. Montreal: AAAI Press. pp. 275–281. Available: http://pages.stern.nyu.edu/~aumyarov/study/books/nyu/alex/silberschatz95subjective.pdf (Accessed 3 September 2016).

Sim, J., Lee, J. S. and Kwon, O. 2015. Missing values and optimal selection of an imputation method and classification algorithm to improve the accuracy of ubiquitous computing application. *Hindawi Publishing Corporation Mathematical Problems in Engineering*. Available: https://www.hindawi.com/journals/mpe/2015/538613/abs/ (Accessed 10 September 2017).

Sinkovits, D. 2006. Flocking behavior. *Semabtic Scholar*, 1–10. Available: http://guava.physics.uiuc.edu/~nigel/courses/569/Essays_Spring2006/files/Sinkovits.pdf (Accessed 10 September 2016).

Siripurapu, A. 2015. Convolutional networks for stock trading. Available: http://cs231n.stanford.edu/reports/2015/pdfs/ashwin_final_paper.pdf (Accessed 5 September 2018).

Smith, T. F. and Waterman, M. S. 1981. Identification of common molecular subsequences. *Journal of Mol. Biology*, 147(1): 195–197. Available: https://pdfs.semanticscholar.org/40c5/441aad96b366996e6af163ca9473a19bb9ad.pdf (Accessed 10 October 2017).

Smolka, J. and Dacke, M. 2017. The pure path-intergation system of homing dung beetles. Available: www.biology.lu.se/research/research-groups/lund-vision-group/research-projects/the-pure-path-integration-system-of-homing-dung-beetles (Accessed 10 September 2017).

Sohangir, S., Wang, D., Pomeranets, A. and Khoshgoftaar, T. M. 2018. Big data: Deep learning for financial sentiment analysis. *Journal of Big Data*. Available: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-017-0111-6 (Accessed 10 September 2017).

Song, M. and Rajasekaran, S. 2006. A transaction mapping algorithm for frequent itemsets mining. *IEEE Transactions on Knowledge and Data Engineering*. 18(4): 472–481. Available: http://doi.ieeecomputersociety.org/10.1109/TKDE.2006.52

Song, Q., Fong, S. and Tang, R. 2016. Self-adaptive Wolf Search algorithm. In: *5th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI).* Kumamoto, Japan: IEEE. pp. 576–582. Available: DOI: 10.1109/IIAI-AAI.2016.102 (Accessed 10 September 2017).

Sorensen, D. C., Lehoucq, R. B., Yang, C. and Maschhoff, K. 2012. Profiling for improving performance. Rice University. Available: www.https://www.mathworks.com/help/matlab/matlab_prog/profiling-for-improving-performance.html (Accessed 10 May 2018).

Spencer, R. L. 2002. Introduction to Matlab. Available: https://www.physics.byu.edu/courses/computational/phys330/matlab.pdf (Accessed 10 September /2017).

Srikant, R. and Agrawal, R. 1996. Mining quantitative association rules in large relational tables. In: *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Monreal, Canada*. New York. pp. 1–12.

Available: http://sci2s.ugr.es/keel/pdf/algorithm/congreso/1996%20-%20SrikantAgrawal-%20Mining%20Quantitative%20association%20rules%20in%20large%20relational%20tables.pdf (Accessed 10 September 2017).

Srivastava, S. 2014. Weka: A tool for data preprocessing, classification, ensemble, clustering and association rule mining. *International Journal of Computer Applications*, 88(10): 26–29. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.429.1463&rep=rep1&type=pdf (Accessed 9 June 2017).

Storn, R. and Price, K. 1997. Differential evolution: A simple and efficient heuristic for global optimisation over continuous spaces. *Journal of Global Optimisation*, 11(4): 341–359. Available: http://www1.icsi.berkeley.edu/~storn/TR-95-012.pdf (Accessed 9 June 2017).

Stützle, T. and Dorigo, M. 2002. *Ant colony optimization*. Cambridge, MA and London: MIT Press. Available: https://pdfs.semanticscholar.org/7c72/393febe25ef5ce2f5614a75a69e1ed0d9857.pdf (Accessed 9 June 2017).

Sumathi, S. and Sivanandam, S. N. 2006. Introduction to data mining principles. *Studies in Computational Intelligence (SCI)*, 29: 1–20. Available: www.springer.com/cda/content/.../cda.../9783540343509-c1.pdf (Accessed 3 May 2017).

Tang, R., Fong, S., Yang, X.-S. and Deb, S. 2012. Wolf search algorithm with ephemeral memory. In: *Seventh International Conference on Digital Information Management (ICDIM), Macau*. China: IEEE. pp. 1–8. Available: DOI: 10.1109/ICDIM.2012.6360147 (Accessed 3 May 2017).

Thakker, D., Osman, T. and Lakin, P. 2009. GATE JAPE grammar tutorial version 1.0. Available: https://gate.ac.uk/sale/thakker-jape-tutorial/GATE%20JAPE%20manual.pdf (Accessed: 19 February 2017).

Thirumuruganathan, S. 2010. A detailed introduction to k-nearest neighbor (KNN) algorithm. Available: https://saravananthirumuruganathan.wordpress.com/2010/05/17/a-detailed-introduction-to-k-nearest-neighbor-knn-algorithm/ (Accessed 3 May 2017).

Tian, Z., Lu, H., Ji, W., Zhou, A. and Tian, Z. 2002. An n-gram-based approach for detecting approximately duplicate database records. Available: https://link.springer.com/content/pdf/10.1007/s007990100044.pdf (Accessed 3 May 2017).

Trawiński, B., Smętek, B., Telec, Z. and Lasota, T. 2012. Nonparametric statistical analysis for multiple comparison of machine learning regression algorithms. *International Journal of Applied Mathematics and Computer Science*, 22(4): 867–881. Available: DOI: 10.2478/v10006-012-0064 (Accessed: 1 December 2017).

Tsai, C.-W. Lai, C.-F., Chao, H.-C. and Vasilakos, A. V. 2015. Big data analytics. *Journal of Big Data*, 2(21): 1–32. Available: https://doi.org/10.1186/s40537-015-0030-3 (Accessed: 19 February 2017).

Tseng, V. S., Liang, T. and Chu, C. 2006. Efficient mining of temporal high utility itemsets from data streams. In: *UBDM'06, Philadelphia, Pennsylvania*. Available: http://www2.ic.uff.br/~bianca/ubdm-camera-ready/tseng-ubdm06.pdf (Accessed: 19 February 2017).

Tu, V. and Koh, I. 2010. A tree-based approach for efficiently mining approximate frequent itemset. In: *2010 Fourth International Conference on Research Challenges in Information Science (RCIS).* Nice, France: IEEE. pp. 1–12. Available:

https://pdfs.semanticscholar.org/b28d/4efa8c6976df350ba2996161479789225f85.pdf (Accessed: 19 February 2017).

Ullman, S., Poggio, T., Harari, D., Zysman, D. and Seibert, D. 2014. Unsupervised learning clustering. Center for Brains, Minds and Machines. Available: www.mit.edu/~9.54/fall14/slides/Class13.pdf (Accessed 25 August 2018).

Uncu, O. and Turksen, I. B. 2007. A novel feature selection approach: Combining feature wrappers and filters. *Inf. Sci.*, 177: 449–466. Available: DOI: 10.1016/j.ins.2006.03.022 (Accessed: 19 February 2018).

Unler, A. and Murat, A. 2010. A discrete particle swarm optimization method for feature selection in binary classification problems. *European Journal of Operational Research*, 206(1): 528–539. Available: https://doi.org/10.1016/j.ejor.2010.02.032 (Accessed 5 September 2018).

Unler, A., Murat, A. and Chinnam, R. B. 2011. mr2PSO: A maximum relevance minimum redundancy feature selection method based on swarm intelligence for support vector machine classification. *Information sciences*, 20(181): 4625–4641. DOI:10.1016/j.ins.2010.05.037 (Accessed 5 September 2018).

Varland, D. E. 1991. *Behavior and ecology of post-fledging American Kestrels: Retrospective Theses and Dissertations Paper 9784*. Iowa State University Digital Repository. Available: https://lib.dr.iastate.edu/cgi/viewcontent.cgi?referer=https://www.google.com/&httpsredir=1&article=10783&context=rtd (Accessed 25 August 2018).

Viitala, J. E. Korplmäki, I., Palokangas, P. and Koivula, M. 1995. Attraction of kestrels to vole scent marks visible in ultraviolet light. *Nature*, 373: 425–427. Available: http://dx.doi.org/10.1038/373425a0 (Accessed 25 May 2018).

Vijayarani, S. and Janani, R. 2016. Text mining: Open source tokenization tools – An analysis. *Advanced Computational Intelligence: An International Journal (ACII).* 3(1): 37–47. Available: doi: 10.5121/acii.2016.3104 37 (Accessed 10 September 2017).

Village, A. 1990. *The Kestrel*. London: T. and. A.D. Poyser. Available: https://doi.org/10.1080/00306525.1992.9634182 (Accessed 10 September 2016).

Village, A. 2010. The Kestrel. *Journal of African Ornithology*, 63(1): 45. Available: https://books.google.co.za/books?isbn=1408138190 (Accessed 25 August 2018).

Vingron, M. and Waterman, M. S. 1994. Sequence alignment and penalty choice review of concepts, case studies and implications. *Journal of Mol. Biol.*, 235: 1–12. Available: https://www.sciencedirect.com/science/article/pii/S0022283605800063?via%3Dihub (Accessed 10 September 2016).

Vlachos, C., Bakaloudis, D., Chatzinikos, E., Papadopoulos, T. and Tsalagas, D. 2003. Aerial hunting behaviour of the lesser kestrel falco naumanni during the breeding season in thessaly (Greece). *Acta Ornithologica*, 38(2):129–134. Available: http://www.bioone.org/doi/pdf/10.3161/068.038.0210 (Accessed 10 September 2016).

Vreeken, J. and Tatti, N. 2014. *Interesting patterns*. Switzerland: Springer. Available: https://eda.mmci.uni-saarland.de/pubs/2014/fpmbook_int-vreeken,tatti.pdf (Accessed 10 September 2016).

Waad, B., Ghazi, B. M. and Mohamed, L. 2013. On the effect of search strategies on wrapper feature selection in credit scoring. In: *2013 International Conference on Control, Decision and Information Technologies (CoDIT)*. Hammamet, Tunisia: IEEE. pp. 218–223. Available: DOI: 10.1109/CoDIT.2013.6689547

(Accessed 10 September 2016).

Wang, J., Han, J. and Li, C. 2007. Frequent closed sequence mining without candidate maintenance. *IEEE Transaction Knowledge and Data Engineering*, 19(8): 1042–1056. Available: http://hanj.cs.illinois.edu/pdf/tkde07_wangj.pdf (Accessed 10 September 2016).

Wang, J., Hedar, A-R., Wang, S. and Ma, J. 2012. Rough set and scatter search metaheuristic based feature selection for credit scoring. *Expert Systems with Applications*, 39(6): 6123–6128. DOI: 10.1016/j.eswa.2011.11.011 (Accessed 5 September 2018).

Wang, J., Zhang, L., Liu, G., Liu, Q. and Chen, E. 2014. On top-k closed sequential patterns mining. In: *11th International Conference on Fuzzy Systems and Knowledge Discovery*. Xiamen, China: IEEE. pp. 295–300. Available: DOI: 10.1109/FSKD.2014.6980849 (Accessed 10 September 2016).

Wang, L., Wang, G. and Alexander, C. A. 2015. Big data and visualization: Methods, challenges and technology progress. *Digital Technologies*, 1(1): 33–38. Available: http://pubs.sciepub.com/dt/1/1/7 (Accessed: 19 February 2017).

Ward, M., Grinstein, G. and Keim, D. 2010. *Interactive data visualization: Foundations, techniques, and application*. London, New York: A. K. Peters/CRC Press.

Wei–yong, Y, Huang J., Zhang, Z., Kong, J. 2015. SIBA: A fast frequent item sets mining algorithm based on sampling and improved bat algorithm. In: *IEEE: Chinese Automation Congress (CAC)*. Available: DOI: 10.1109/CAC.2015.7382471 (Accessed 5 September 2018).

Weston, J., Chopra, S. and Adams, K. 2014. #TAGSPACE: Semantic embeddings from hashtags. In: *Proceedings of the 2014 Conference on Empirical Methods in*

Natural Language Processing (EMNLP). Association for Computational Linguistics. 1822–1827. Available: http://emnlp2014.org/papers/pdf/EMNLP2014194.pdf (Accessed: 05-September-2018).

Winkler, W. E. 1999. *The state of record linkage and current research problems: Statistics of income division, internal revenue service publication R99/04*. Bureau of the Census, Washington, DC, USA. Available: http://www.census.gov/srd/www/byname.html (Accessed: 19 February 2017).

Wits University. 2013. Dung beetles follow the milky way: Insects found to use stars for orientation. *ScienceDaily*. Available: https://www.sciencedaily.com/releases/2013/01/130124123203.htm (Accessed 20 March 2017).

Wonner, J., Grosjean, J., Capobianco, A. and Bechmann, D. 2012. Starfish: A selection technique for dense virtual environments. In: *Proceedings of the ACM symposium on virtual reality software and technology*. New York: ACM. 101–104. Available: https://dl.acm.org/citation.cfm?id=2407356 (Accessed 20 March 2017).

Wu, C., Buyya, R. and Ramamohanarao, K. 2016. Big data analytics = machine learning + cloud computing. Available: https://arxiv.org/ftp/arxiv/papers/1601/1601.03115.pdf (Accessed 19 February 2017).

Wu, X., Zhu, X., Wu, G.-Q. and Ding, W. 2014. Data mining with big data. *IEEE Transactions on Knowledge and Data Engineering*. 26(1): 97–107. Available: DOI: 10.1109/TKDE.2013.109 (Accessed 5 September 2018).

Wur, S. Y. and Leu, Y. 1998. An effective boolean algorithm for mining association rules in large databases. In: *Proceedings of the 6th International*

Conference on Database Systems for Advanced Applications. Hsinchu, Taiwan: IEEE. pp. 179–186.

Xue, B., Bing, W. N. and Zhang, M. 2014. Particle swarm optimisation for feature selection in classification: Novel initialisation and updating mechanisms. *Applied Soft Computing*, 18: 261–276. Available: DOI: 10.1016/j.asoc.2013.09.018 (Accessed: 19 February 2017).

Yamany, W., Emary, E. and Hassanien, A. E. 2014. Wolf search algorithm for attribute reduction in classification. In: *Computational Intelligence and Data mining (CIDM): 2014 IEEE Symposium on Computational Intelligence and Data Mining.* pp. 351–358. DOI: 10.1109/CIDM.2014.7008689 (Accessed 5 September 2018).

Yan, G. and Li, C. 2011. An effective refinement artificial bee colony optimization algorithm based on chaotic search and application for PID control tuning. *Journal of Computational Information Systems*, 7(9): 3309–3316. Available: https://pdfs.semanticscholar.org/5266/804e780141929df71af4f75774c10c4c60ad.pdf (Accessed 19 February 2017).

Yan, X., Han, J. and Afshar, R. 2003. CloSpan: Mining closed sequential patterns in large databases. In: *Proc. SIAM Int'l Conf. Data Mining (SDM '03)*. SIAM. 166–177. Available: https://pdfs.semanticscholar.org/e8b7/783db07a89917e8fb1a06227e3a9d64b4242.pdf (Accessed 19 February 2017).

Yang, L., Shi, Z., Xu, L., Liang, F. and Kirsh, I. D. H. 2011. Frequent pattern mining on hadoop using JPA. In: *Proceedings of The International Conference On Granular Computing.* IEEE. pp. 875–878. Available:

http://doi.ieeecomputersociety.org/10.1109/GRC.2011.6122552 (Accessed 19 February 2017).

Yang, T., Lin, Q. and Jin, R. 2015. *Big data analytics: Optimization and randomization*. Sydney, Australia: SIGKDD. Available: https://homepage.cs.uiowa.edu/~tyng/kdd15-tutorial.pdf (Accessed 19 February 2017).

Yang, X. S. 2009. Firefly algorithm, levy flights and global optimization. In: *XXVI Research and Development in Intelligent Systems.* London: Springer. 209–218. Available: DOI: 10.1007/978-1-84882-983-1_15 (Accessed 19 February 2017).

Yang, X.-S. 2008. *Nature-inspired metaheuristic algorithms*. 2nd ed. Cambridge, UK: Luniver Press. Available: https://www.researchgate.net/publication/235979455_Nature-Inspired_Metaheuristic_Algorithms (Accessed 19 February 2017).

Yang, X.-S. 2010. Firefly algorithm, levy flights and global optimization. In: *Research and Development in Intelligent Systems XXVI*. London: Springer. pp. 209–218. Available: DOI: 10.1007/978-1-84882-983-1_15 (Accessed 19 February 2017).

Yang, X.-S. 2013. Bat algorithm: Literature review and applications. *Int. J. Bio-Inspired Computation*, 5(3): 141–149. Available: DOI: 10.1504/IJBIC.2013.055093 (Accessed 9 March 2018).

Ye, F., Wang, Z. J, Zhou, F. C, Wang Y. P. and Zhou, Y. C. 2013. Cloud-based big data mining and analyzing services platform integrating. In: *Proceedings of the international conference on advanced cloud and big data*. Nanjing, China. pp. 147–151. Available:

https://pdfs.semanticscholar.org/a92c/d40e32f48878fa544b2ede0afc8f33368961.pdf (Accessed 19 February 2017).

Yin, J., Zheng, Z., Cao, L., Song, Y. and Wei, W. 2013. Efficiently mining top-k high utility sequential patterns. In: *Proceedings of 2013 IEEE 13th International Conference on Data Mining*. Dallas: IEEE. pp. 1259–1264. Available: DOI: 10.1109/ICDM.2013.148 (Accessed 19 February 2017).

Yongmei, L. and Yong, G. 2008. FP-growth algorithm for application in research of market basket analysis. In: *IEEE 6th International Conference on Computational Cybernetics*. Stara Lesna, Slovakia: IEEE. pp. 269–272. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4721419&tag=1 (Accessed 19 February 2017).

Zaki, M. J. 2001. Parallel sequence mining on shared-memory machines. *Journal of Parallel and Distribution Computing*, 61(3): 401–426. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.150.8178&rep=rep1&type=pdf (Accessed 19 February 2017).

Zar, J. H. 1999. *Biostatistical analysis*. 4th ed. Upper Saddle River: Prentice Hall.

Zhang, L., Liu, L., Yang, X.-S. and Dai, Y. 2016. A novel hybrid firefly algorithm for global optimization. *PLoS ONE*, 11(9): e0163230. Available: https://doi.org/10.1371/journal.pone.0163230 (Accessed 19 February 2017).

Zhao, Y., MacKinnon, D. J. and Gallup, S. P. 2005. Big data and deep learning for understanding DOD data. *Data mining and measurements*, 1–10. Available: https://pdfs.semanticscholar.org/a8d2/9b405aa928a3c0b399f387aa77a04569b67a.pdf (Accessed 19 February 2018).

Zhenxin, Z. and Jiaguo, L. 2009. Closed sequential pattern mining algorithm based positional data. In: *Advanced Technology in Teaching - Proceedings of the 3rd International Conference on Teaching and Computational Science (WTCS)*. Berlin, Heidelberg: Springer. pp. 45–53.

Zyl, A. V. 2013. Kestrels in the Karoo. *African Birdlife*, 36–42. Available: http://www.uct.ac.za/sites/default/files/image_tool/images/275/Publications/semi-popular/2013/AB01%285%2936-42.pdf (Accessed 19 September 2016).

Web references

Apache Hadoop. 2015. Available: http://hadoop.apache.org/ (Accessed 2 February 2015).

Apache Mahout. 2015. Available: http://mahout.apache.org/ (Accessed 2 February 2015).

Apache Storm. 2015. Available: http://storm.apache.org/ (Accessed 2 February 2015).

Appendix 1: Summary of data mining algorithms, advantages and limitations

Author	Algorithm	Mining approach	Approach used	Advantages	Limitations
Aggarwal and Han (2014)		Apriori-based methods			A candidate-generation-and-test strategy produces a large number of candidate sequences and also requires more database scans (Tu and Koh 2010) when there are long patterns
Han, Pei, Mortazavi-Asl, Chen, Dayal and Hsu (2000)		Pattern-growth methods		Compressed database structure that is smaller than original dataset	
Han, Pei, Mortazavi-Asl, Chen, Dayal and Hsu (2000)	FreeSpan	Pattern-growth methods	Sequential patterns by partitioning		
Pei, Han, Mortazavi-Asl, Pinto, Chen, Dayal and Hsu (2001)	PrefixSpan	Pattern-growth methods	Pseudo-projection technique for constructing projected databases	Generates and tests candidate sequences that exist in a projected database	Projected database requires more storage space, and extra time is required to scan the projected database
		Sequential Pattern Mining	Vertical format based methods	Fast computation of support counting	
Zaki (2001)	SPADE	Sequential Pattern Mining	Vertical format based methods; either breadth-first or depth-first manner		Consumes more memory space
Ayres *et al.* (2002)	SPAM	Sequential Pattern Mining	Vertical format based methods; traverses the sequence tree in a depth-first manner.	A vertical bitmap of the database	Consumes more memory space

Agrawal and Srikant (1995)	AprioriAll	Sequential Pattern Mining	Horizontal format based method		Consumes more memory space
		Closed sequential pattern mining		Efficient use of search space pruning, reduced number of patterns, finds more interesting patterns	
Yan *et al*. (2003)	CloSpan	Closed sequential pattern mining	Prefix sequence lattice, post pruning		Huge search space for checking the closure of new patterns
Wang, Han and Li (2007)	BIDE	Closed sequential pattern mining	Depth-first search order; performs closure checking (BIDE)		Multiple database scans, more computational time Without candidate maintenance it does not keep track of historical closed sequential patterns
Huang *et al*. (2006)	COBRA	Closed sequential pattern mining	Bi-phase Reduction Approach, item encoding, pruning methods (LayerPruning and ExtPruning), vertical and horizontal database formats	Reduces searching space	Requires large memory space
Raju and Varma (2015)	ClaSP	Closed sequential pattern mining	Vertical database format, Frequent Closed Candidates, recursive post-pruning (CheckAvoid-able for pruning the search)		Requires more main memory

Han, Wang, Lu and Tzvetkov (2002)		Top-*k* Closed Sequential Pattern Mining	Descending order of support	Minimum support not specified	User must decide the value of *k*; prior knowledge of database required
Hirate *et al*. (2004)	TF^{2}P-growth	Top-*k* Closed Sequential Pattern Mining	Descending order of support	Does not require the user to set any threshold value *k*; output of frequent patterns to user sequentially and in chunks	Time consuming to check all chunk sizes
Wang, Zhang, Liu, Liu and Chen (2014)	BI-TSP	Top-*k* Closed Sequential Pattern Mining	BI-Directional checking scheme, minimum length constraint, dynamically increase support of *k*		
Raju and Verma (2015)	CSpan	Top-*k* Closed Sequential Pattern Mining	Depth-first search, occurrence checking method for early detection of closed sequential patterns, constructs the projected database		Projected database requires more storage space, and extra time is required to scan the projected database

Appendix 2: Summary of the advantages and disadvantages of meta-heuristic search methods

Algorithm	Advantages	Disadvantages
WSA	Can remember previously visited position.	The performance depends heavily on the manually chosen parameter values (Song, Fong and Tang 2016).
PSO	1. The search can be carried out by the speed of the particle. During the development of several generations, only the most optimist particle can transmit information onto the other particles, and the speed of the researching is very fast (Bai 2010). 2. It has a very simple computation process (Bai 2010) 3. There is a limited number of parameters, including only the inertia weight factor and two acceleration coefficients in comparison with other competing heuristic optimization methods (Lee and Park 2006).	The method easily suffers from partial optimism, which causes it to be less exact in the regulation of its speed and direction (Bai 2010).
ACO	Can be used in dynamic applications (adapts to changes such as new distances) (Shekhawat *et al.* Boswal 2009).	1. Time to convergence is uncertain (but convergence is guaranteed) (Shekhawat *et al*. 2009). 2. Probability distribution changes by iteration (Shekhawat *et al*. 2009). 3. Sequences of random decisions (not independent) (Shekhawat *et al*. 2009).
Bat	1. Uses echolocation and frequency tuning to solve problems (Yang 2013). 2. Uses parameter control, which can vary the values of parameters as the iterations proceed. This provides a way to automatically switch from exploration to exploitation when the optimal solution approaches (Yang 2013).	1. How to speed up the convergence of an algorithm (Yang 2013). 2. Lacks best control strategy so as to switch from exploration to exploitation within a right or specified time (Yang 2013).
Bee	Robustness, fast convergence, high flexibility and fewer setting parameters (Yan and Li 2011).	Premature convergence in the later search period, and the accuracy of the optimal value

		sometimes cannot meet the requirements (Yan and Li 2011).
Firefly	Signaling mechanism to communicate with other fireflies. The signaling system consists of rhythmic flash, frequency of flashing light and time period of flashing (Yang 2010).	There is the high probability of being trapped in local optima because they are local search algorithms (Zhang *et al*. 2016).