THE MULTIPLE PHEROMONE ANT CLUSTERING ALGORITHM

VOL 1 OF 1

JAN CHIRCOP

Doctor of Philosophy in Computer Science

ASTON UNIVERSITY

MAY 2014

ASTON UNIVERSITY

Thesis Summary

The Multiple Pheromone Ant Clustering Algorithm

by Jan CHIRCOP

Degree: Doctor of Philosophy

May 2014

Ant Colony Optimisation algorithms mimic the way ants use pheromones for marking paths to important locations. Pheromone traces are followed and reinforced by other ants, but also evaporate over time. As a consequence, optimal paths attract more pheromone, whilst the less useful paths fade away. In the Multiple Pheromone Ant Clustering Algorithm (MPACA), ants detect features of objects represented as nodes within graph space. Each node has one or more ants assigned to each feature. Ants attempt to locate nodes with matching feature values, depositing pheromone traces on the way. This use of multiple pheromone values is a key innovation.

Ants record other ant encounters, keeping a record of the features and colony membership of ants. The recorded values determine when ants should combine their features to look for conjunctions and whether they should merge into colonies. This ability to detect and deposit pheromone representative of feature combinations, and the resulting colony formation, renders the algorithm a powerful clustering tool.

The MPACA operates as follows: (i) initially each node has ants assigned to each feature; (ii) ants roam the graph space searching for nodes with matching features; (iii) when departing matching nodes, ants deposit pheromones to inform other ants that the path goes to a node with the associated feature values; (iv) ant feature encounters are counted each time an ant arrives at a node; (v) if the feature encounters exceed a threshold value, feature combination occurs; (vi) a similar mechanism is used for colony merging.

The model varies from traditional ACO in that: (i) a modified pheromone-driven movement mechanism is used; (ii) ants learn feature combinations and deposit multiple pheromone scents accordingly; (iii) ants merge into colonies, the basis of cluster formation.

The MPACA is evaluated over synthetic and real-world datasets and its performance compares favourably with alternative approaches.

Acknowledgements

Without the patience and assistance of several people this thesis would have never materialised. A thesis which required a considerable amount of commuting, in conjunction with a full time job in an area unrelated to the field of research was harder than I could have ever imagined.

I immeasurably thank my supervisor, Dr. Christopher Buckingham who was happy to dedicate extra hours when required and met with me innumerable times after office hours. Furthermore, I would also like to thank Dr. M. Chli and Prof. F. Guinand, respectively the internal and external examiners for their helpful feedback on the thesis.

Other important people to thank include my family for their unfailing support and lastly, but not the least, Steph who got onto the ride with me at the very start, and by the end, even though the ride got bumpy, was still there to encourage me through it.

Contents

Abstract **2**

Acknowledgements **3**

Contents **4**

List of Figures **8**

List of Tables **9**

Abbreviations **10**

1 Introduction **13**
1.1 Problem of Data Explosion and Motivation . . . 13
1.2 Understanding and Interpreting Data: Top-Down and Bottom-Up approaches . 14
1.3 Swarm Intelligence . . . 15
1.3.1 Self-organisation . . . 16
1.3.2 Stigmergy . . . 16
1.3.3 Positive Feedback . . . 17
1.3.4 ACO in a Nutshell . . . 18
1.4 Objectives of this Thesis . . . 19
1.4.1 The New ACO Model . . . 19
1.4.2 Evaluation Techniques . . . 20
1.4.2.1 Datasets . . . 20
1.4.3 MPACA Overview . . . 20
1.5 Publications . . . 21
1.6 Organisation of Work . . . 21
1.7 Chapter Conclusion . . . 22

2 Literature Review **23**
2.1 Chapter Overview . . . 23
2.2 Clustering and Classification . . . 23
2.2.1 Definition of Clustering . . . 24
2.2.1.1 The Clustering problem as an NP-hard Problem . . . 25
2.2.2 Operands within Clustering Algorithms . . . 25
2.2.2.1 Architectures . . . 25
2.2.2.2 Distance Metrics . . . 25
2.2.2.3 Similarity Metrics - Distance as the Similarity Proxy . . . 27
2.2.2.4 Curse of Dimensionality . . . 29
2.2.3 Comparing Models . . . 29
2.2.3.1 External Evaluation Techniques . . . 29
2.2.4 General Clustering Techniques . . . 31

2.2.4.1 Hierarchical Clustering . . . 31
2.2.4.2 Flat or Partitional Clustering . . . 33
2.2.4.3 Density-based Clustering . . . 35
2.2.4.4 Graph-theoretic Clustering approaches . . . 36
2.2.5 The Relevance of Traditional Clustering Algorithms . . . 37
2.3 Swarm Intelligence . . . 39
2.3.1 General Swarm Intelligence Approaches . . . 39
2.3.2 Particle Swarm Optimisation . . . 40
2.3.3 Bio-Inspired SI Algorithms . . . 41
2.3.3.1 Bee Colony Algorithms . . . 41
2.3.3.2 Ant Algorithms . . . 42
2.3.4 Towards the Ant Colony Optimisation Meta-Heuristic . . . 43
2.3.4.1 Relevance and novelty to the MPACA . . . 47
2.4 Conclusions from SI literature . . . 48
2.5 Ant Algorithms and their Application to Clustering . . . 48
2.5.1 Fundamental Operators Behind the Chosen Models . . . 48
2.5.1.1 Type I - Knowledge Structure Forming Ants . . . 49
2.5.1.2 Type II - Ant Aggregations and Ants' Self-Aggregation . . . 49
2.5.1.3 Type III - Clustering Inspired by the Chemical Recognition System of Ants . . . 51
2.5.1.4 Type IV - Clustering using Ant Colony Optimisation Algorithms 52
2.5.2 Comparison Criteria . . . 54
2.6 Selected Models in Detail . . . 57
2.6.1 Type I - Clustering using Ants' Self-Aggregation . . . 57
2.6.1.1 The AntTree Algorithm . . . 57
2.6.2 Type II - Clustering using Ant Aggregations and Ants' Self-Aggregation 59
2.6.2.1 Standard Ant Clustering Algorithm (SACA) . . . 59
2.6.2.2 Self-Aggregation within a 2D Grid . . . 62
2.6.2.3 Ant Aggregation through Pheromone in a 2D Grid . . . 63
2.6.3 Type III - Clustering Inspired by the Chemical Recognition System of Ants . . . 66
2.6.4 Type IV - Clustering using Ant Colony Optimisation Algorithms . . . 70
2.6.4.1 Multi-Objective Problem Solving . . . 70
2.6.4.2 Multi-Colony and Multi-Pheromone ACO Approaches . . . 71
2.6.4.3 Ant Colony Optimisation (ACO) and its Application to Clustering . . . 73
2.6.4.4 ACO Applied to Graph Partitioning . . . 75
2.6.4.5 Rule Learning Algorithms . . . 82
2.7 Chapter Conclusion and Introduction to the MPACA . . . 86

3 The MPACA Model **87**
3.1 Chapter Overview . . . 87
3.2 Introduction to the MPACA . . . 87
3.2.1 Relationship to the Generic Ant Colony Algorithm . . . 88
3.3 The Main Model Architecture and Processes . . . 88
3.3.1 Partially-Connected Graph Space . . . 90
3.3.2 Connecting Nodes and Measuring Distances . . . 90
3.3.3 Multiple-steps within Edges . . . 92
3.4 Placing Ants on Nodes . . . 92
3.4.1 Assigning Features to Ants . . . 92
3.4.2 Feature Matching a Node . . . 93
3.5 Ant Movement . . . 93

3.5.1 Pheromone . . . 93
3.5.1.1 Ant Deposit State . . . 94
3.5.1.2 Pheromone Quantity Deposited . . . 94
3.5.1.3 Pheromone Evaporation . . . 95
3.5.2 Edge Selection Mechanism . . . 96
3.5.3 Ant Encounters . . . 97
3.5.3.1 Identifying Ant Feature Encounters . . . 98
3.5.3.2 Data Structures for Recording Encounters . . . 98
3.5.4 Merging Features and Colonies . . . 100
3.5.4.1 Merging Features: A Learning and Forgetting Mechanism . . . 100
3.5.4.2 Colony Merging . . . 101
3.6 Overall Operation of the MPACA . . . 102
3.7 The MPACA and Cluster Derivation . . . 105
3.7.1 Mapping Colonies to Clusters . . . 105
3.7.2 Termination Criteria . . . 105
3.7.3 Cluster Membership and Evaluation . . . 105
3.7.3.1 Centroid Cluster Membership Calculation . . . 106
3.8 The MPACA Parameters . . . 106
3.8.1 A Synthetic Dataset for Demonstrating the Parameters' Impact . . . 106
3.8.2 Baseline Analysis . . . 107
3.8.3 Domain Initialisation Analysis . . . 109
3.8.3.1 Maximum Edge Length Parameter . . . 110
3.8.3.2 Step Size Parameter . . . 111
3.8.4 Ant Initialisation Analysis . . . 113
3.8.4.1 Ant Complement Parameter . . . 113
3.8.4.2 Detection Range for Ordinal Dimensions Parameter . . . 114
3.8.5 Pheromone Deposition and Movement Analysis . . . 115
3.8.5.1 Pheromone quantity, maximum coefficient and the evaporation parameters . . . 115
3.8.5.2 Residual Parameter . . . 117
3.8.6 Merging Thresholds . . . 118
3.8.6.1 Feature Merging Threshold Parameter . . . 118
3.8.6.2 Colony Merging Threshold Parameter . . . 120
3.8.6.3 Visibility on Edge Parameter . . . 121
3.8.6.4 Time-window Parameter . . . 122
3.9 The MPACA as a Classifier . . . 123
3.9.1 Training Termination Criteria . . . 124
3.9.2 Evaluation of the MPACA as a Classifier . . . 124
3.10 Novelty and Contribution of the MPACA . . . 125
3.10.1 Distinctive Elements of the MPACA . . . 125
3.10.2 Variations of the MPACA from the Traditional Clustering Algorithms . . . 126
3.10.3 Variations from Ant Based Clustering Literature . . . 127
3.10.4 Advantages of the MPACA Architecture . . . 128
3.10.5 Novelty in Ant Movement . . . 129
3.10.6 Ability to Learn and Acquire Features . . . 130
3.10.7 Multiple Pheromones and Multiple Colonies . . . 131
3.11 Chapter Conclusion . . . 132

4 The MPACA Applied **133**
4.1 Chapter Overview . . . 133
4.2 Evaluation Criteria and Experiment Set-up . . . 133
4.2.1 Synthetic Datasets: the 2D-4C and 10D-10C Datasets . . . 134

4.2.2 Real-world UCI datasets . . . 134
4.2.2.1 Iris dataset . . . 134
4.2.2.2 Wine dataset . . . 134
4.2.2.3 Soya-bean dataset . . . 135
4.2.2.4 Wisconsin Breast Cancer dataset . . . 135
4.2.2.5 Pima Indians Diabetes dataset . . . 135
4.2.2.6 Yeast dataset . . . 135
4.3 Experimentation Framework . . . 136
4.3.1 Basic Set-up . . . 136
4.3.2 Analysis based on a Simulated Annealing Technique . . . 136
4.4 Baseline Experiments . . . 137
4.4.1 Observations from Baseline Experiments . . . 141
4.4.2 Evaluating the MPACA Clustering Performance . . . 142
4.5 Sensitivity Analysis of chosen Parameters . . . 146
4.5.1 Sensitivity Metric . . . 146
4.5.2 Pheromone Driven versus a Random Model . . . 147
4.6 Real-world GRiST and ADVANCE datasets . . . 148
4.6.1 GRiST - Mental health risk assessment . . . 148
4.6.2 ADVANCE - Hub-and-Spoke Logistics Networks . . . 150
4.6.2.1 Predicting Shipments . . . 151
4.7 Discussion of Results Attained . . . 154

5 Conclusion and Future Work **155**
5.1 Summary . . . 156
5.1.1 Unique properties of the MPACA . . . 156
5.1.1.1 Modified Ant Transition Mechanism . . . 156
5.1.1.2 Feature Learning and the Multi-Pheromone Mechanism . . . 156
5.1.1.3 Multi-Colony Clustering via Colony Formation . . . 157
5.1.2 Goals and Objectives met by the MPACA . . . 157
5.2 Alternative Paths . . . 157
5.2.1 Variation in the Ant Types . . . 158
5.2.2 Acquisition of Multiple Features on Each Dimension . . . 158
5.2.3 Feature Merging with No-Forgetting Mechanism . . . 158
5.2.4 Cluster Representation in First Order Logic . . . 158
5.3 Advances of the Algorithm . . . 159
5.4 Current Limitations and Recommended Improvements . . . 160
5.4.1 Parallel versus Non-Parallel . . . 160
5.4.2 Parameters and Parameter Adjustment . . . 161
5.4.3 Termination Criteria . . . 161
5.4.4 Tackling Uneven Datasets . . . 161
5.5 Future work . . . 162
5.5.1 Bayesian Cluster Membership Calculation . . . 162
5.5.2 K-Nearest Neighbourhood Cluster Membership Calculation . . . 163
5.5.3 Ongoing Research . . . 164
5.5.4 Application of the MPACA as a Classifier . . . 165
5.6 Concluding Arguments . . . 165

List of Figures

1.1 Experimental set-up demonstrating the shortest path finding 17

2.1 Dwellings that appertain to the two parishes and possible configurations. 24
2.2 Euclidean versus City block distance problem depiction 26
2.3 Dendrogram representation of the animal kingdom categorisation 31
2.4 Density-reachability and density-connectivity in the DBSCAN 35
2.5 Fire ants coalesce together to form a bridge with their bodies 49
2.6 A sequential clustering task of corpses performed by a real ant colony 50
2.7 Principles of the ANTCLUST 52
2.8 Clustering as an Optimisation Problem 53
2.9 The tournament selection mechanism 79

3.1 A finite state machine representing the changes in ant behaviour as it traverses the graph. 103
3.2 Results of varying the maximum edge length parameter 110
3.3 Results of varying the step size parameter 112
3.4 Results of varying the ant complement parameter 113
3.5 Results of varying the detection range for continuous domains parameter 115
3.6 Results of varying the pheromone related parameters 116
3.7 Results of varying the residual parameter 117
3.8 Results of varying the feature merging parameter 119
3.9 Results of varying the colony merging parameter 120
3.10 Results of varying the visibility range parameter 121
3.11 Results of varying the time-window parameter 123

4.1 Transportation in a multiple hub-and-spoke logistics system. 150
4.2 Fluctuations in the number of pallets each day for a specific depot 151

List of Tables

3.1 The MPACA parameter settings as applied to the Square1 dataset 108

4.1 Summary of the selected datasets 136
4.2 The MPACA as applied to the Square1, 2D-4C, 10D-10C, Iris and Wine datasets. 138
4.3 The MPACA as applied to the Soya-Bean, WBC, Pima and Yeast datasets. . . . 139
4.4 The MPACA performance applied over synthetic datasets 142
4.5 The MPACA performance applied over real-world datasets 143
4.6 Parameter sensitivity analysis as applied to the Iris and Wine dataset. 147
4.7 Pheromone Driven versus a Random Model applied to the Iris and Wine dataset. 148
4.8 Parameter settings for the MPACA as applied to the GRiST dataset 149
4.9 Parameter settings for the MPACA as applied to the ADVANCE dataset 153
4.10 Results of the MPACA as applied to the ADVANCE dataset 154

5.1 Parallel versus Non-Parallel execution over the Iris dataset 160

List of Algorithms

1 ANTCLUST main algorithm 66
2 Ant Colony Optimisation with Different Favour (ACODF) algorithm 80
3 Overview of Ant-Miner 83
4 The MPACA Outline 89
5 Pheromone deposition 94
6 Updating Feature and Colony Encounters 99
7 Feature and Colony Merging 102
8 The core of the MPACA algorithm 104
9 Proximity to Centroid calculation 106
10 Bayesian Approach to cluster membership calculation 163
11 K-Nearest Neighbourhood approach to cluster membership calculation 163

Abbreviations

A^4C	Adaptive artificial ants clustering
AS	Ant system
AACA	Adaptive ant-based clustering algorithm
ABC	Artificial bee colony algorithm
ACO	Ant colony optimisation
ACODF	Ant colony optimisation with different favour
ACS	Ant colony system
AI	Artificial intelligence
AIS	Artificial immune system
AL	Artificial life
ALife	Artificial life
ANTCLUST	Ant clustering based on a modelling of the chemical recognition system of ants
ANTS	Approximate non-deterministic tree search
AntTAG	Ant tree adjunct grammar
APC	Aggregation pheromone density based clustering
APS	Aggregation pheromone system
ASM	Ant sleep model
AST	Absolute sensory threshold
ATTA	Adaptive time dependent transporter ants
ATTA-C	ATTA applied to explicit cluster-retrieval
ATTA-TM	ATTA applied to specific for topographic mapping
BCO	Bee colony optimisation
BM	Basic model
BWAS	Best-worst ant system
CA	Cellular automata
CAS	Colour ant system algorithm
CST	Contrast sensory threshold
DAG	Directed acyclic graph
DBSCAN	Density based spatial clustering of applications with noise
EM	Expectation maximization
FSOM	Fuzzy self-organising map
FOL	First order logic
FP	Foraging pheromone
FP	False positive
FN	False negative
GA	Genetic algorithm
GRiST	The Galatean risk and safety tool
HMACO	Homogeneous MACO
HPP	Hamiltonian path problem
ISBN	International standard book number
IWD	Intelligent water drops
K-NN	K-nearest neighbours
LF	Lumer and Faieta
MACO	Multiple ant colony optimisation
MCL	Markov cluster algorithm
MDL	Minimum description length
MDS	Multi-dimensional space
ML	Maximum likelihood
MMACO	Multi-Object MACO
MMAX	Max-Min ant system
MPACA	Multiple pheromone ant clustering algorithm
MST	Minimum spanning tree
NP-Complete	Non-deterministically polynomial-complete
NP-hard	Non-deterministic polynomial-time
ODUEC	Organisation detection using emergent computing
PABC	Parallel artificial bee colony
PACO	Parallel ACO
PAM	Parallel Ant-Miner on high performance cluster

PAPI	Partially asynchronous parallel implementation
PDF	Probability density function
PSO	Particle swarm optimisation
QAC	Quick ant clustering
RFD	River formation dynamics
SA	Simulated annealing
SACA	Standard ant clustering algorithm
SD	Standard deviation
SDS	Stochastic diffusion search
SI	Swarm intelligence
SSE	Sum of squared errors
SI-ACC	State information-based ant colony clustering
TN	True negative
TP	True positive
TSP	Travelling salesman problem
UCI	UC Irvine
WBC	Wisconsin breast cancer
WEKA	Waikato environment for knowledge analysis
XML	Extensible markup language

Chapter 1

Introduction

1.1 Problem of Data Explosion and Motivation

Ninety percent of the world's data has been generated over the last two years [A, 2013]. Every minute hundreds of hours of video are uploaded to YouTube [YouTube, 2013], each day 175 million tweets are sent, each month seven petabytes of photo content are added on Facebook (statistics till end of 2012) [Pingdom, 2013]. Big data is meaningless unless it can be converted into comprehensible and relevant information. Accurate information has to be separated from noise, whilst existing patterns should be made easily comprehensible to the consuming party. Most importantly, all these activities must happen within a timely fashion. In many cases information is only valid for minute intervals of time. For example, a trade executed on erroneous information in high-frequency trading can have an impact costing billions of pounds [Kablan and Ng, 2010].

The majority of datasets are unstructured, distributed, and massive. This poses a serious challenge to the current categorisation processes. In many cases, human intervention in tagging data is not financially viable. Turning such immense quantities of data into usable information is one of the biggest ongoing challenges for computer scientists [Yadav et al., 2013].

This challenge is taken up in this thesis, where an algorithm that can convert unstructured data into meaningful information is introduced. This algorithm is based on a swarm intelligence technique, more specifically, an ant-inspired one. The Multiple Pheromone Ant Clustering Algorithm (MPACA) is a clustering method that is comparable to other swarm-based techniques, as well as alternative generic clustering approaches. In this thesis MPACA is applied to synthetic and standard datasets, as well as two real-world large datasets, one from the logistics domain and the other for mental health risk assessments.

In literature, two main approaches are used to categorise data; either a structured and supervised top-down tactic or an unstructured, unsupervised bottom-up route, both of which are described below.

1.2 Understanding and Interpreting Data: Top-Down and Bottom-Up approaches

Extracting hidden information from big data is non-trivial [Yadav et al., 2013]. One approach involves analysing the data from the top-down, using existing knowledge structures that in many cases are labelled by human experts in the field. Acquiring such information is expensive, inefficient [Weinstein and Deyo, 2000] and, worst of all, error-prone [James et al., 2008]. Experts in the field can dispute or mislead such tasks [Raykar et al., 2009]. This makes supervised learning approaches infeasible on large, less structured datasets. Thus, algorithms that can learn and adapt are required [Silver, 2011].

An alternative to the above is unsupervised learning where no preconceptions exist and the data is tackled from the bottom-up [Berkhin, 2006]. This eliminates the need for expensive field experts, whilst downsizing the time and cost. What makes this approach even more interesting is that the structure is built around the data, in the manner it is deemed to fit best.

Top-down and bottom-up approaches differ on the level of information known a priori. Referring to the wine dataset [Bache and Lichman, 2013], this contains data elements having 13 descriptive attributes each, where each attribute represents a measure regarding a property of the wine. Each data element has a variable identifying the particular variety of the grape from which the wine is produced, namely "the cultivar". This variable represents the class that each data element belongs to.

In relation to the above, in terms of classification, class information is used as part of the training information. A training set is used to build rules which determine class membership. Accuracy of these rules is verified by determining the success rate of allocating data elements, unused for training, into their correct class membership.

In clustering, on the other hand, the perspective changes, as the required output is unknown. For example the Wine dataset has no structure other than the 13 attributes. The entire dataset is processed, and areas of similarity are defined as the clusters. A successful clustering algorithm splits the data into meaningful clusters, that should in theory correlate to known class separations. This implies that clustering adds structure to the data. In essence, identifying clusters is the task of partitioning datasets into clusters which have a higher common likeness, so that a data

element shares more similarities with its member group rather than any other group [Chircop and Buckingham, 2011b].

There is a correlation between clustering, which is the discovery of patterns from unlabelled datasets, and how a collection of simple entities collectively perform complex tasks in a swarm, or swarm intelligence. Clustering can be seen as an emergent property of particular swarm intelligence activities. Swarm intelligence is purely a bottom-up approach. It is an ideal candidate for tackling the clustering problem since it is distributed, decentralised and operated by the repeated actions of primitive entities.

1.3 Swarm Intelligence

Living within a swarm has its advantages [Kordon, 2010] and it can also prove crucial for survival [Bonabeau et al., 1999]. The power of a swarm comes from the collaboration of decentralised primitive entities, where the whole is greater than the sum of its parts. The entities of a swarm are mostly powerless, with restricted intelligence. It is only via their collective interactions that complex behaviour emerges out of simpler individuals [Zhu and Tang, 2010]. These interactions form the basis of swarm intelligence (SI), which is defined as:

> *"any attempt to design algorithms or distributed problem-solving devices inspired by the collective behaviour of social insects and other animal societies,"* [Bonabeau et al., 1999].

Flocking in birds is a mechanism which inspired Particle Swarm Optimisation (PSO) [Kennedy and Eberhart, 1995]. Bee colony optimisation algorithms (BCO) [Teodorovic et al., 2006] [Teodorović and Dell'Orco, 2005], and ant colony optimisation algorithms (ACO) [Dorigo, 1992], are inspired by the foraging behaviour of honey bees and ant colonies respectively.

A typical example of this swarming in action applied to the clustering problem is inspired by the organisation of cemeteries and larvae in ants [Deneubourg et al., 1990b]. Ants move around randomly in space, while picking or dropping corpses of other ants. When applied collectively at the colony level, and each action is repeated enough times, clusters of dead corpses provide an analogy to data clustering.

Bottom-up analysis is a term often linked with collective or swarm intelligence. This is a mechanism which provides problem-solving abilities and is highly adaptable to dynamic environments. This approach induces complex global behaviours through local interactions among agents [Nakahori, 2000]. Three key terms are linked with understanding SI techniques: *self-organisation*, *stigmergy* and *positive feedback*.

1.3.1 Self-organisation

"Self-organisation is a set of dynamical mechanisms whereby structures appear at the global level of a system from interactions of its lower-level components." [Bonabeau et al., 1999].

A dominant principle in SI is the process of self-organisation, or the attempt to create order out of a disordered system [Prigogine et al., 1984]. From fire ants which form bridges with their bodies [Akbar, 2008], to termites building complex mounds [Worall, 2011], and even the flocking of birds, these are all examples of self-organising activities. The latter is a common example of self-organisation in SI literature [Kennedy and Eberhart, 1995]. Birds flock using three simple rules; separation (collision avoidance), alignment (velocity matching) and cohesion (flock centring). These rules allow birds to form interesting formations, sometimes involving hundreds or thousands of birds.

Self-organisation is interesting for bottom-up analysis, since a set of simple instructions, repeated numerous times, is capable of creating complex structures. In many swarming insects, this self-organisation is aided by the use of scents within the environment, which increase the interaction and collaboration within the swarm. These scents represent stigmergy.

1.3.2 Stigmergy

Pheromones, or hormonal chemicals, are used to relay information to other members of the same swarm. For instance, the typical ant colony has about 10-20 different pheromone types. Differing pheromones play different roles, including alerting the colony of an attack [Powell and Clark, 2004], helping ants to recognise fellow nest mates [Labroche, 2003] and indicating food sources [Deneubourg et al., 1990a].

Stigmergy is defined as the effect of an external influence placed within an environment; pheromone trails, which are marked paths to be followed, represent a prime example of this. These external stimuli manipulate the behaviour of other entities within the same environment without interacting directly with them. In 1959, biologist Pierre-Paul Grassé defined this as the "stimulation of workers by the performance they have achieved." This is derived from stigma (stain) + ergon (work), or work through the process of following signs/stains [Grassé, 1959].

The beauty of stigmergy is that it serves as a mechanism of indirect coordination that promotes spontaneous coordination. It induces the formation of apparently-intelligent structures, without the need for any planning, control, or even any direct communication between the participants. As such, it supports efficient collaboration between extremely simple agents, which can lack memory, intelligence or even awareness of each other. Stigmergy is a catalyst of the actions

performed by the swarm. This self-organisation through stigmergy catalysis is defined as a positive feedback.

1.3.3 Positive Feedback

Initially, ants foraging for food walk in a more-or-less random manner. Whilst doing this, they deposit a pheromone trail. When an ant finds food, it returns home, depositing a stronger trail. As ants have trail-following behaviour, the number of ants which tend to follow such a trail grows. In turn, each ant traversing this trail reinforces it, with this amplification termed as "positive feedback" (auto-catalytic system).

This is demonstrated by Deneubourg *et al.* in their double bridge experiment [Deneubourg et al., 1990a], figure (1.1). In this experiment Argentine ants, *Iridomyrmex humilus*, are allowed to roam the environment searching for nourishing sources. Initially ants tend to randomly choose one of the two bridges. Paths gradually weaken and disappear due to pheromone evaporation, with a proportional relationship existing between distance and time for evaporation. As a result, the shorter paths tend to accumulate more pheromone, which in turn attract more ants in this positive feedback effect.

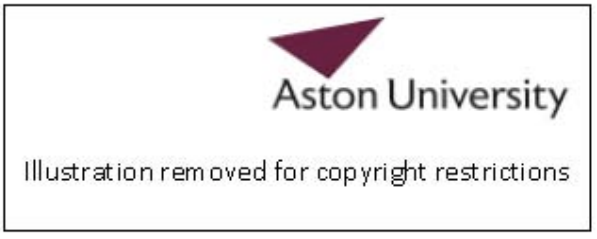

FIGURE 1.1: An experimental set-up demonstrating that the ants are capable of finding the shortest path between the nest and a food source. In figure (a) the bridge is initially closed, figure (b) demonstrates the initial distribution of ants after the bridge is open, and figure (c) shows the distribution of ants after some time has passed since they were allowed to exploit the food source [de Castro, 2007].

Positive feedback is formally defined as a closed loop system, where changes in the output tend to be amplified back within the system, with negative feedback being its antonym, or the stabilising factor [Monmarché et al., 2010]. A practical combination of all three SI terms described previously is found in the ant colony optimisation algorithm.

1.3.4 ACO in a Nutshell

The ability of ants to find the shortest paths, as outlined in the double bridge experiment, forms the basis of ant colony optimisation (ACO). Dorigo applies this mechanism to the travelling salesman problem (TSP) [Dorigo, 1992]. The TSP is used as the common representation for ACO problems. Here the domain space is set up as a graph, with cities being represented by nodes, where n nodes are connected by a number of edges, e. These edges provide possible paths that the ants can traverse. The distance between two nodes is the distance on the $edge(i, j)$. The problem is considered asymmetric when the distance on $edge(i, j) \neq edge(j, i)$. The aim is to find the shortest-possible distance that covers all the nodes in a graph, at least once. This is called a Hamiltonian Path.

In the TSP, each ant constructs a Hamiltonian path by initially moving randomly by departing a node and visiting each and every other node in the sequence it deems best. At each step, the ant probabilistically chooses the edge to traverse next among edges that lead to unvisited nodes. This decision is calculated depending on which edge leads to the node with the highest combination of two-factors; (i) the highest pheromone amount present (trail) and (ii) a heuristic value which is the inverse of the distance function (visibility). Once all the ants have completed their tour, the pheromone on the edges is updated. Over time, the same positive feedback loop which caused ants in the double bridge experiment to choose the shortest path, causes ants to act accordingly over graph space. The process is repeatedly applied until a termination criterion is satisfied. Founding terminologies in ACO include:

- The entity is an *ant*, a primitive being with both limited abilities and memory;
- A collection of ants is a *colony*;
- A *path* is a collection of edges that have been traversed in a sequence;
- When an ant completes a traversal of nodes within a graph, this is called an *ant tour or cycle*;
- Nodes visited by an ant are stored within a *TABU list*, a structure used to ensure that ants visit the same node only once within a tour.

ACO is applied in literature to a multitude of problems; in route finding, [Manjurul Islam et al., 2006], [Agarwal et al., 2005], load-balancing [Bertelle et al., 2007], [Wang and Wang, 2008], scheduling problems [Chen et al., 2007b], [Liu and Chai, 2006], [Kheirkhahzadeh and Barforoush, 2009], constraint satisfaction problems [Shi et al., 2004], [Solnon, 2001], [Merkle et al., 2002], [Sun and Teng, 2002] and quadratic assignment problems [Maniezzo, 1999], [Talbi et al., 1998], [Mouhoub and Wang, 2008], [Tsutsui and Liu, 2007].

In a similar fashion as ACO is applied to finding the minimal tour distance in the TSP, similar approaches can be used for solving the clustering problem. Spatially distributed nodes within a graph are allocated to one cluster over another depending on their inherent composing values and the similarity that exists between them. The edges that connect nodes are additional distorted by pheromone traces increasing the likelihood of certain node connections over others. Eventually, nodes are grouped as sub-graphs, and it is these sub-graphs which are representative of clusters. This forms the basis of clustering via the use of ACO as presented in this thesis. The importance of this ACO technique is that it uses all three of the pivotal elements found in SI; self-organisation, stigmergy and positive feedback.

1.4 Objectives of this Thesis

The objective of this thesis is to introduce and evaluate a new clustering algorithm, the Multiple Pheromone Ant Clustering Algorithm (MPACA). The MPACA is further benchmarked against other clustering algorithms both traditional and nature-inspired. The algorithm uses multiple colonies of decentralised ants, distributed in a problem space for the final aim of clustering objects depending on their composing feature values. Simple ants with limited centralised control, that communicate indirectly with other ants via the use of pheromone traces, catalyse their actions using ACO principles and self-organise to form larger colonies. Whilst doing so, ants also learn about other features present in the given problem space. The final colonies consist of a common description of the surrounding environment within which the ants are found. These colonies are a representation of clusters. The MPACA is therefore a clustering algorithm that is critically reviewed and benchmarked against the state of the art clustering algorithm literature.

1.4.1 The New ACO Model

A number of distinct features are introduced in this clustering algorithm, which collectively are not found in any other ant-driven algorithm in literature, namely:

1. The utilisation of a modified pheromone-driven ant movement mechanism. This mechanism encourages further path searching, and is more stochastic than other methods found in literature.
2. The ability of ants to learn and acquire new features, and deposit pheromone depending on the features being sought by the ant. This utilises a multi-pheromone system unlike any other technique.
3. The utilisation of a multi-colony approach as a bottom-up cluster-forming algorithm, inspired by the parallel, asynchronous and independent activities of ants. This includes the

ability of colonies to vary in size, with ants migrating from colonies and merging into larger ones.

1.4.2 Evaluation Techniques

Benchmarking against a number of standard datasets is key. However, as a well-known side effect of bottom-up clustering, direct mapping of clusters is challenging. Clustering is, in itself, a subjective term. A number of measures are presented which capture the reformulated structured data from the MPACA and are subjected to traditional evaluation mechanisms.

Upon execution of the algorithm, a number of clusters are formed. These clusters are defined by populations of ants, with each population having a number of features. The main technique used to determine cluster membership consists of a centroid calculation and proximity to the centroid value. Subsequently the centroid of each cluster is discovered and data points used for clustering purposes, are allocated to the cluster that is closest to the centroid.

A number of metrics are used for evaluation, further detailed in section (2.2.3), which include Precision, Recall, F-Measure, Rand-Index, and Jaccard Coefficient.

1.4.2.1 Datasets

The MPACA is applied to a number of standard benchmark datasets commonly used in literature, both synthetic datasets including the Square1, 2D-4C, 10D-10C generated by normal distributions, and also real-world datasets. The latter originate from the UCI data repository [Bache and Lichman, 2013], namely the Iris, Wine, Soya-beans, Wisconsin breast cancer, Pima Indians diabetes, and the Yeast datasets. The MPACA is further applied to real world datasets, which are much larger and less structured, namely the Galatean Risk and Safety Tool, GRiST [Buckingham and Adams, 2013] and a hub-and-spoke logistics domain, that is the ADVANCE [adv, 2013] dataset.

1.4.3 MPACA Overview

The MPACA is a clustering algorithm, where a cluster is expressed as a colony of ants. Initially each ant has its own specific colony. It is only through a process of ant interactions that ant colonies merge into bigger colonies, thus clustering. The ants move around in graph space, based on the concept of positive feedback, common to ACO, where stronger paths are reinforced, and weaker paths are allowed to dissipate. Initially, ants are born into a node in graph space, and are imprinted with a feature existing at such point, within a particular dimension. This is the base feature and represents the feature which the ant will set out to look for. The ant traverses the

graph looking for other nodes which match this feature, and also deposits a pheromone value corresponding to this feature.

As the ant continues to traverse the graph, it encounters other ants. Each ant will carry other feature values for other particular dimensions. If the ant keeps on encountering a key feature carried by a number of ants for an exceedingly high number of times, and this feature dimension is not already carried by the ant, this co-occurrence of features is an indicator that the features are related. Therefore, the ant absorbs this other feature, and starts to search for pheromone traces which lead to the combination of all features now carried by the ant. From this point on, the ant deposits and searches for multiple-pheromone combinations. This process is repeated until an ant acquires a feature for each dimension present in the search. This process of feature-acquisition is contained within a time-window, allowing ants to drop the features they acquire. Thus, a forgetting mechanism is inbuilt in the system, which over time clears possible invalid combinations.

As ants acquire more features, the number of nodes they can react to is reduced. This causes the ants to remain localised within an area of graph space, causing them to encounter some ants more frequently than others. Each ant has a label representing the colony it is part of. If a count for a particular colony occurs more than a number of times, that ant migrates to this colony by changing its own label. Therefore, all processing is asynchronous and localised to the ant in scope.

Finally, as stable colonies form, the algorithm terminates, thus producing colonies of ants which represent clusters. The values of these clusters are defined by the feature values being carried by the ants within each colony.

1.5 Publications

The work presented in this thesis has been in part published or accepted for publication. Publications in reverse chronological order, are as follows; [Chircop and Buckingham, 2013], [Chircop and Buckingham, 2014], [Chircop and Buckingham, 2011b], [Chircop and Buckingham, 2011a]. These publication present a concise and concrete representation of the research work carried out.

1.6 Organisation of Work

Chapter (1) introduced the thesis subject matter and outlined the investigation which shall propagate throughout this thesis. This chapter highlighted the importance of dealing with data from

an underlying bottom up approach. Chapter (2) presents the literature overview investigating swarm intelligence and ant algorithms, evaluating the state of the art clustering algorithms, both for traditional clustering and those originating from the ant metaphors, against the MPACA. Chapter (3) introduces the MPACA itself; the data structure necessary, its core operating principles, the MPACA in pseudo-code, and an explanation of the applicable parameters. This chapter serves to highlight the core activities and interacting parameters, and also locates the MPACA within the literature review, concluding by highlighting the novelty and contribution introduced by the MPACA. Chapter (4) applies the algorithm to a number of standard synthetic and readily available UCI repository datasets, a task which serves to determine the sensitivity analysis of the parameters applicable to the MPACA. Finally, the results attained in chapter (4) are benchmarked against values attained by the algorithms introduced in chapter (2). Chapter (5) concludes this work with a critical evaluation of what has been achieved, potential improvements to the algorithm and how this can be applied in future work.

1.7 Chapter Conclusion

This chapter introduced the problem domain that the MPACA strives to tackle. The concepts of top-down and bottom-up approaches, together with key SI and ACO terms, are introduced. Amongst other SI algorithms, the ant algorithm, based on the foraging theory, was chosen. This is mainly due to its use of stigmergy, optimisation ability and its decoupling from supervisory control. Other algorithms utilise similar mechanisms, but none offer the extent of literature as can be found under the umbrella of ant algorithms, thus making comparisons more intuitive. This is not the first ant algorithm applied to clustering, however, the inclusion of a number of variations implemented ensure its uniqueness.

The chapter which follows consists of the literature overview, in which the problem domain is explored into further detail. The chapter reviews traditional clustering algorithms, and evaluation metrics to be used throughout the thesis.

Chapter 2

Literature Review

2.1 Chapter Overview

This chapter reviews the state of the art with regards to algorithms relevant to the MPACA introduced in chapter (3) of this thesis. This review is conceptually split into three parts. Firstly, the clustering problem is defined and with it the classical clustering approaches are introduced. Next, swarm intelligence algorithms are investigated, with special focus on the ant colony optimisation (ACO) algorithm. Finally, a critical review of ant algorithms aimed at solving the clustering problem is undertaken.

2.2 Clustering and Classification

Clustering and classification together form the basis of machine learning [Mitchell, 1997]. The distinction between both lies in the level of previous knowledge. Classification is a supervised learning approach, where a class is known and training sets are used to train algorithms to recognise which data elements correspond to which class values. Classification is based on the Inductive Learning Hypothesis. That is, "any hypothesis found to approximate the target function well over a sufficiently large set of training examples will also approximate the target function well over other unobserved examples" [Mitchell, 1997]. Clustering, on the other hand, represents an unsupervised learning approach, as it allows learning of a hidden data concept. Through clustering it is possible to acquire a knowledge structure without a training set, or any correcting mechanism [Jain et al., 1999].

2.2.1 Definition of Clustering

Clustering is the task of partitioning datasets into groupings of common likeness, so that a data element shares more similarities with its member group rather than any other group. High quality clusters are identified by the high intra-cluster similarity (homogeneity) and the low inter-class similarity (separation). Clustering is often a complex problem because of ambiguous boundaries between classes, for example when does a plant become a tree rather than a shrub? This classic fuzzy problem, combined with uncertainty about what classes one is even expecting from a large dataset, has led to a variety of approaches for optimising clusters, including ones that are modelled on insect behaviour [Chircop and Buckingham, 2011b].

A trivial two cluster problem is depicted in figure (2.1). In this example a two dimensional (2D) map representative of a geographic space is illustrated, in which the location of dwellings belonging to two parishes are plotted. Traditionally village cores are built around a church or some other landmark building. Dwellings mushroom around these central points, where proximity to such points increases dwelling density. Each parish claims a number of dwellings under its territory. A dwelling is more likely to belong to one of the two parishes, depending on the proximity to the particular centre. Arbitrary configurations are possible, a process dependent on the selection criteria applied to the clustering process.

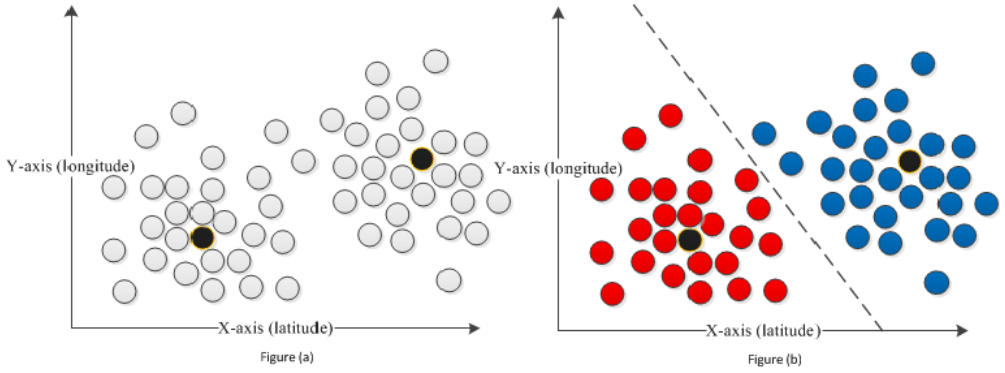

FIGURE 2.1: Figure (a) depicts dwellings that appertain to the two parishes. Figure (b) depicts parish configurations coloured in red and blue, (Author self).

Clustering is mathematically defined as follows; for a given space S, k subsets are created, with each subset represented by a cluster C. This implies that $C = C_1, .., C_k \in S$, such that $S = \bigcup_{i=1}^{k} C_i$ and $C_i \bigcap C_j = 0$ for $i \neq j$. Consequently, any instance in S belongs to exactly one and only one subset [Maimon and Rokach, 2005]. Mathematically this represents the optimal configuration, however the problem arises when dealing with ambiguous situations.

2.2.1.1 The Clustering problem as an NP-hard Problem

In a clustering problem, N data points must be allocated into K clusters, in a process which aims to minimise total distortion. Clustering problems are known to be NP-hard problems [Mettu, 2002]. NP-hard problems are in turn defined by Cook and Karp [Hochbaum, 1997], [Demontis, 2009] as problems which to date cannot be solved in polynomial time. This is because a super-polynomial lower bound has not yet been defined.

2.2.2 Operands within Clustering Algorithms

Clustering algorithms have a number of key operands. Such operands include architectures used, distance and similarity metrics.

2.2.2.1 Architectures

Clustering requires a spatial context in which data elements are located. Three possible architectures exist; grid based, multi-dimensional space and graph based.

In grid based clustering, clustering takes place on a two dimensional (2D) grid. Objects are initially laid down randomly, with the initial spatial positioning of objects irrespective of their underlying properties. Objects are subsequently repositioned according to their similarity, resulting in areas of higher similarity. The final positioning of objects is based on the cluster they belong to. This mechanism is used in ant clustering based on organising bodies in cemeteries.

Unlike grid space, in the multi-dimensional space (MDS) architecture, each data element attribute within an object is expressed in terms of a dimension within Cartesian or geometric space. The location of objects is based on their descriptive properties. It would be infeasible to have entities traverse any point of the MDS for problems with higher dimensionality, such as the subject problems of this thesis, since huge areas of empty space would exist but be of no relevance. A better way to explore this architecture is to exclude the ability of entities to visit these gaps in space, by linking geometric data points with “roads” or edges, hence the use of graph space.

Tightly coupled with the aforementioned architectures is the analysis of the distance metrics. That is, the distance between any two objects within a selected spatial configuration.

2.2.2.2 Distance Metrics

The distance between two data elements represents the degree of similarity. There are a number of metrics which are used to calculate the distance between two points, P and Q. Common

approaches included; (i) Euclidean, and (ii) City block distance.

The Euclidean distance given in equation (2.1) uses the ordinary distance between two points. This is calculated using the Pythagorean formula, where the square of the hypotenuse is equivalent to the sum of the squares of the other two sides. For two positions in space, this distance z, is calculated as the difference on the x and y axis, where $z = \sqrt{x^2 + y^2}$. Figure (2.2) depicts the problem of traversing between two blocks. The shortest distance between two blocks would entail a diagonal route, which would imply flying over city blocks as per figure (b). Walking, on the other hand, would entail a longer route to reach block B from block A, as it is impossible to circumvent buildings, hence the term city block or Manhattan distance, figure (c). This is defined in equation (2.2)

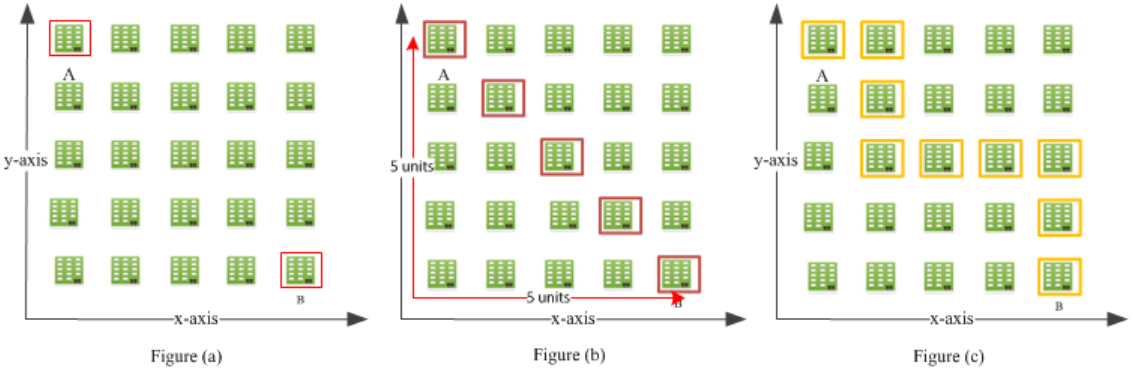

FIGURE 2.2: Figure (a) block A and block B, marked in red; Figure (b) distance between block A and block B using Euclidean distance; Figure (c) distance between block A and block B using City block distance, (author self).

The Minkowski distance [Kruskal, 1964] can be considered as a generalisation of both the Euclidean and Manhattan distances. In this metric, the distance between two vectors is the norm of their difference equation (2.3).

$$d(P,Q) = d(Q,P) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + ... + (q_n - p_n)^2} = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2} \tag{2.1}$$

$$d(P,Q) = \sum_{i=1}^{n} |q_i - p_i| \tag{2.2}$$

$$d(P,Q) = (\sum_{i=1}^{n} |p_i - q_i|^r)^{\frac{1}{r}} \tag{2.3}$$

In the above equations, p and q are Cartesian co-ordinates where $P = (p_1, p_2, ..., p_n)$, $Q = (q_1, q_2, ..., q_n)$, d is the value associated with the selected distance measure, n is the number of dimensions, and r is a parameter which when set to 1, 2 correlates the Minkowski respectively to the Manhattan and the Euclidean distance.

More complex metrics such as the Mahalanobis distance [Maesschalck et al., 2000] provides a way to measure the similarity between a set of conditions. This measure takes into account the covariance amongst variables. The Mahalanobis distance is used to find outliers in a dataset.

Another similarity measure used is the Cosine similarity [Kryszkiewicz, 2013]. It calculates the cosine of the angle between two vectors of an inner product space, and is therefore a measurement of orientation and not magnitude. The cosine between two vectors $\vec{a},\vec{b}$ is calculated as in equation (2.4):

$$\vec{a}\cdot\vec{b} = ||\vec{a}||\,||\vec{b}||cos\theta \;,\; cos\theta = \frac{\vec{a}\cdot\vec{b}}{||\vec{a}||\,||\vec{b}||}, \text{ therefore } \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n}(A_i)^2} \times \sqrt{\sum_{i=1}^{n}(B_i)^2}} \tag{2.4}$$

where $\vec{a} = A$ and $\vec{b} = B$. The resulting similarity ranges from -1, representing a complete mismatch, to 1 representing an exact match. In order to obtain a distance measure, one must subtract the cosine similarity from one; $dist(a,b) = 1 - cos(a,b)$.

Many clustering algorithms operate on ordinal data, allowing distance calculations and comparisons to be more intuitive. However, not all distance measures can be equally applied to every problem domain, this since not all data is ordinal data. Tagging non-ordinal values with a corresponding numeric value in ordinal space introduces a bias. One alternative to determine differences between mix-type datasets, is to sum the difference between matches and mismatches for each dimension. This effectively normalises the data [Maimon and Rokach, 2005].

Another distance measure that operates on strings is the Hamming distance [Hamming, 1950], which represents the number of differing coefficients between two strings of equal length. This returns the minimum number of changes required to convert one string into another, or the minimum error value.

Distance measures are synonymous with dissimilarity, which implies that the inverse of the distance is a measure of similarity.

2.2.2.3 Similarity Metrics - Distance as the Similarity Proxy

The application of distance metrics as similarity proxies is a fundamental approach within clustering algorithms. In grid space, the differences between objects is computed using the distance of their composing attributes. In a spatial context, as in MDS, objects are positioned according to their attributes, thus objects which are closer to each other have higher similarity than objects which are further away.

The standard or z-score: A common mechanism to standardise the granularity of differences for values within dimensions to more discrete intervals is the z-score [Kreyszig, 2000]. For a given data element, x, it returns the absolute value represented in units of standard deviations that this element is above or below the mean. This standard score is calculated as in equation (2.5):

$$z = \frac{x - \mu}{\sigma} \tag{2.5}$$

where μ is the mean and σ is the standard deviation calculated over the selected dimension values. Standardisation mitigates the problem of having an uneven distribution of sample attributes along each dimension.

Weighted Dimensions: In many cases the data elements used do not have an equal number of representations for each dimension. This is especially so when dealing with incomplete datasets, wherein some of the data elements fail to represent on one or more of the relevant dimensions. For example, in some cases dimension α might be represented ten times, whereas dimension β would just be represented five times. Thus, dimensions cannot have the same weight as they do not have the same relevance and this could skew results.

To overcome such a bias, the represented dimensions are weighted. This involves the application of higher or lower weights to reflect more or less the influence of a particular dimension on clustering. This is achieved by adding coefficient weights, w, assigned to each dimension, having a value from $0 < w \leq 1$. The inverse of frequency is used in such a way that, whilst the weight of a variable with low variance is high, the weight of a variable with high variance is low. For a J-dimensional vector, with s_j being the sample standard deviation of the j-th variable, $w_j = 1/s_j^2$, and is the inverse of the j-th variance calculated as in equation (2.6):

$$d(p,q) = \sqrt{\sum_{j=1}^{J} = w_j(p_i - q_j)^2} \tag{2.6}$$

where p and q are Cartesian co-ordinates, where $p = (p_1, p_2, ..., p_n)$ and $q = (q_1, q_2, ..., q_n)$ [Greenacre, 2013]. This is not the only compensation mechanism that exists, but is the method which has been chosen to be applied in the MPACA.

A more intricate problem is that occurring due to the sparsity of data samples, as dimensionality is increased. This is referred to as the curse of dimensionality.

2.2.2.4 Curse of Dimensionality

The "curse of dimensionality" as coined by Bellman [Bellman, 1957], represents a challenge in statistics. The increase in dimensionality increases the volume of space, making data points more sparse. This in turn, makes it less likely that a sample set exists for each dimensionality setting, and the data points tend be unique to a given specific set of dimensions. This prohibits accurate clustering from occurring, as data points do not gather the statistical significance required to form a cluster. This increase in dimensions must be coupled with a substantial increase in observations, which is unlikely to happen, hence dimensionality compression is required. Algorithms such as the MPACA mitigate such a problem by applying normalisation techniques.

2.2.3 Comparing Models

Models are evaluated using internal and external evaluation. Internal evaluation metrics measure the distance between clusters and within the clusters, whilst external evaluation metrics measure the quality of the cluster formed for the known class value. Internal evaluation measures are difficult to benchmark with other existing literature, since different normalisation techniques and other factors manipulate the data and hence are not used in this thesis. External evaluation measures are more consistent, however they usually require some manual intervention, that is matching the cluster to the class value.

2.2.3.1 External Evaluation Techniques

Precision and Recall: The term "Precision", as defined by equation (2.7), is used to measure how many of the correctly classified samples are positive samples, whilst the term "Recall", as defined by equation (2.8), is used to measure how many positive samples are correctly classified [Theodoridis and Koutroumbas, 2006].

$$Precision = \frac{TruePos}{TruePos + FalsePos} \tag{2.7}$$

$$Recall = \frac{TruePos}{TruePos + FalseNeg} \tag{2.8}$$

where TruePos is the number of cases covered by the rule and having the same class as that predicted by the rule, FalsePos is the number of cases covered by the rule and having a different class from that predicted by the rule, FalseNeg is the number of cases that are not covered by the rule, whilst having the class predicted by the rule and TrueNeg is the number of cases that are correctly not assigned to the class.

Rand-Index and Jaccard Coefficient: The Rand-Index determines the degree of similarity with the known correct solution, reflecting its class label (group) and the solution obtained by the clustering algorithm [Theodoridis and Koutroumbas, 2006]. It is defined in equation (2.9):

$$Rand = \frac{SS + DD}{SS + SD + DS + DD} \tag{2.9}$$

where SS, SD, DS, DD represent the number of possible pairs of data points, i and j. In SS both data points belong to the same cluster and the same group, in SD both data points belong to the same cluster but different groups, in DS both data points belong to different clusters but the same group and in DD both data points belong to different clusters and different groups. The value of Rand is in the range $[0,1]$ and the higher the value, the better is the clustering performance.

On the other hand, the Jaccard coefficient (J) which also measures the similarity between sample sets, is defined as the size of the intersection divided by the size of the union of the sample sets as in equation (2.10):

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{2.10}$$

where A and B are two sets of data. In essence the Jaccard coefficient is the same as the Rand-Index, except that it excludes DD, as $|A \cap B| = SS$ and $|A \cup B| = SS + SD + DS$, as defined in equation (2.11):

$$J = \frac{SS}{SS + SD + DS} \tag{2.11}$$

The value of J lies in the interval $[0,1]$. The higher the value of J, the better is the clustering solution.

F-Measure: This is a more elaborate measure of accuracy, which takes into account both the precision and the recall to compute a score. The F-Measure can be interpreted as the weighted harmonic mean of Precision (P) and Recall (R) [equation (2.12)], [van Rijsbergen, 1979]. The harmonic mean H of N positive values $a_1, a_2, a_3, ..., a_N$, is equal to the value N divided by the reciprocal of the summation of the reciprocals of N, or $H(a_1, a_2, a_3, ..., a_N) = \frac{N}{\frac{1}{a_1} + \frac{1}{a_2} + ... + \frac{1}{a_N}}$

$$\text{F-Value} = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{2.12}$$

Further generalisations apply, where a weighted parameter, $\beta > 1$, is added to penalise false negatives more strongly than false positives, as defined in equation (2.13):

$$\text{F-Value} = \frac{(\beta + 1) \times Precision \times Recall}{Precision + Recall} \tag{2.13}$$

In order to comply with the results attained from other studies, β is set to 1. The overall F-value for the partition computed between possible class i and cluster j is given by equation (2.14):

$$\text{F-Measure} = \sum_i \frac{n_i}{n} max_j\{\text{F-Value}(i,j)\} \tag{2.14}$$

where n is the total size of the dataset and the F-Measure is limited to the interval $[0,1]$, which value is to be maximised.

The evaluation measures allow comparisons with a number of classical and swarm based clustering algorithms. The overview undertaken above, relating to the clustering problem, distance metrics, and evaluation techniques serves as a prelude to the review of general clustering techniques discussed below.

2.2.4 General Clustering Techniques

Clustering is an essential tool related to many fields, including statistics [Wu and Ren, 2008], pattern recognition [Baraldi and Blonda, 1999], financial modelling [Wei and Ji, 2011] and data-mining [Kriegel et al., 2009]. The variety of clustering algorithms applied in these areas and their importance to the MPACA are collectively reviewed in section (2.2.5). Traditional clustering techniques are broadly divided into hierarchical and partitioning mechanisms.

2.2.4.1 Hierarchical Clustering

FIGURE 2.3: Dendrogram representation of the animal kingdom categorisation, an adaptation from [1902encyclopedia, 2013].

Hierarchical clustering is subdivided into agglomerative and divisive clustering, differentiated by the direction of clustering. In agglomerative clustering, the direction is bottom-up where various data elements are each in a cluster of their own, and merge into larger clusters. In divisive clustering, each cluster is split into smaller and smaller clusters until there can be no more divisions. Both mechanisms are essentially different sides of the same coin, since they both utilise a structure referred to as a dendrogram [Chen et al., 2009]. The animal kingdom is used as an example in figure (2.3) to illustrate the equivalence of a dendrogram to this categorisation

of animals. A partition of the data elements can be obtained by cutting the dendrogram at the desired level [Grira et al., 2004].

This algorithm operates by linking sets of observations together, until all observations link into the hierarchy. The repeated application of this process causes larger clusters to form. That is, at each step sets of observations are linked together depending on the pairwise distance between observations, using mechanisms called linkage techniques. A number of variations exist for such linkage techniques, including; the single-link, the complete-link, the average-link [Sneath and Sokal, 1973], or a technique which utilises minimum-variance [Ward Jr, 1963], [Murtagh, 1984].

The single-link distance between clusters, C_i and C_j, is the minimum distance between any object in C_i and any object in C_j. The distance is defined by the two most similar objects, equation (2.15):

$$min\{d(a,b) : a \in A, b \in B\} \tag{2.15}$$

The complete-link distance between clusters, C_i and C_j, is the maximum distance between any object in C_i and any object in C_j. The distance is defined by the two most dissimilar objects, equation (2.16):

$$max\{d(a,b) : a \in A, b \in B\} \tag{2.16}$$

The group average distance between clusters, C_i and C_j, is the average distance between any object in C_i and any object in C_j, equation (2.17):

$$\frac{1}{||A||B||} \sum_{a \in A} \sum_{b \in B} d(a,b) \tag{2.17}$$

The distance between clusters, C_i and C_j, is the difference between the total within-cluster sum of squares for the two clusters separately, and the within-cluster sum of squares resulting from merging the two clusters into cluster C_{ij}. This is also known as the Ward minimum criterion as it minimises total within-cluster variance, equation (2.18):

$$d_{ij} = d(\{A_i\}, \{B_j\}) = ||A_i - B_j||^2 \tag{2.18}$$

In all cases, a and b are the observation points in space A and B respectively, and d is the chosen distance metric, as described in section (2.2.2.2).

Hierarchical clustering is typically fast and efficient. The simplicity of the linkage metrics used gives it a high level of flexibility and is applicable across any attribute type. Disadvantages include vagueness of termination criteria and the fact that only one clustering parse takes place.

This implies that constructed clusters are not iteratively improved [Berkhin, 2006].

Other types of algorithms aim to directly obtain a single partition of the collection of items into clusters. This is done by decomposing the dataset into a set of disjoint clusters, namely partitional clustering, which follows.

2.2.4.2 Flat or Partitional Clustering

Flat or partitional clustering attempts to determine a number of partitions that optimise an objective function, or a cluster quality measure. Cluster optimisation is an iterative process [Jain et al., 1999]. Unlike hierarchical methods, where clusters are formed in one parse, partitioning algorithms operate on a gradual improvement mechanism. Depending on the quality of the cluster formed, further iterations are computed, a process which is continued until either a maximum number of iterations has been reached, or until the improvement between iterations is below a specified threshold. This approach returns higher quality clusters, however, is more computationally heavy than the single parse hierarchical approach. There are two main subtypes in the field of partitional clustering, namely centroid-based clustering and probabilistic clustering [Jain and Dubes, 1988].

Centroid-based clustering, K-Means: In centroid-based clustering a number of partitions are generated and for this configuration of partitions an objective function is used, which represents overall cluster quality. Seeds are used as initial cluster placements. The remaining data points are allocated to each "seed point", thus forming a cluster. The most popular clustering algorithm of this type is the K-Means algorithm.

The term "K-Means" was first used by MacQueen [MacQueen, 1967], with the original idea itself being proposed earlier by Steinhaus [Steinhaus, 1956]. In the K-Means, the objective function is to minimise the squared distances from the mean. K centroids are chosen, each representing a seed point, with each seed point being also a cluster centre, thus the name K-Means. For a set of observations, or data points, $x_1, x_2, ..., x_n$, with each observation being a d-dimensional real vector, the nearest cluster centroid is calculated using a distance function. This determines cluster membership. In most cases the distance function is calculated using Euclidean distance. Other variations apply the Minkowski or Mahalanobis metrics, section (2.2.2.2). This process is repeated until all points have been assigned to a centroid. When this occurs, new k-centroids are calculated [Grira et al., 2004]. Thus N data points are converted into K disjoint subsets, S_j, each containing N_j data points in such a way that the sum of squared errors (SSE) is reduced to

a minimum, with the objective function expressed as equation (2.19):

$$J = \sum_{j=1}^{K} \sum_{n \in S_j} ||x_n - \mu_j||^2. \tag{2.19}$$

where x_n is the vector representing the n-th data point and μ_j is the geometric centroid of the data point in S_j.

K-Means is a straightforward algorithm. The way clustering is performed in the K-Means makes the sequence of data entry non-influential to the final clusters formed. Despite this, it suffers from a number of shortcomings, mainly revolving around the pre-selection of the seed points. In the standard approach, random seeds are selected, however numerous literature demonstrates that improved results and quicker convergence is achieved with an appropriate seed selection mechanism (see [Pavan et al., 2012]). Multiple algorithms result in various seed selections, potentially causing a combinatorial explosion problem. The K-Means also requires real-valued data, lacks scalability, is sensitive to outliers, and the objective function can be misleading when contrasted with the entire spatial context [Berkhin, 2006], all of which are considered algorithm deficiencies. Other variations apply different SSE methods, the most popular being the Fuzzy C-Means [Bezdek et al., 1984]. These fuzzy methods tend to be more successful at avoiding local minima.

Other algorithms operate on a similar iterative mechanism. However, the allocation of a data point into a cluster is based on a probability distribution rather than the distance from the mean.

Probabilistic Clustering: In Expectation Maximization (EM) each data point has a probability value of belonging to a cluster [Dempster et al., 1977]. The algorithm presumes that there is a statistical distribution, that is, a probability density function (PDF), that can be approximated over a cluster distribution. The EM algorithm is also an iterative procedure that computes the Maximum Likelihood (ML) estimate in the presence of missing or hidden data. The ML estimates the model parameters which are most likely for the data points presented. Each EM iteration consists of two steps, an expectation step (probabilistic) which assigns points to clusters and a maximisation step, that is estimating model parameters that maximize the likelihood for the given assignment of points.

Convergence is assured since the algorithm is guaranteed to increase the likelihood at each iteration [Jain and Dubes, 1988]. The key lies within the ML estimation, which aim is to find parameters which maximise the probability of finding the PDFs which best describe (approximate) the clusters being sought.

Traditional partitioning clustering algorithms tend to define a cluster by the proximity to the locus of the cluster, which represents a point where each parameter value is the mean of the parameter values of all the points in the cluster. This approach favours recognition of spherical shapes, whilst its weakness is recognition of non-spherical shapes and outliers. Other methodologies exist which compensate for the drawbacks of both the K-Means and EM algorithms, these include density-based algorithms.

2.2.4.3 Density-based Clustering

Clusters can be defined as consisting of denser regions of space, separated by regions of lower object density. This concept is used in the density-based clustering algorithms. The advantage of these algorithms is that they create arbitrary shaped clusters [Jain et al., 1999]. The principle algorithm under this capping is the Density Based Spatial Clustering of Applications with Noise (DBSCAN) [Ester et al., 1996a], later generalised by the OPTICS algorithm [Ankerst et al., 1999].

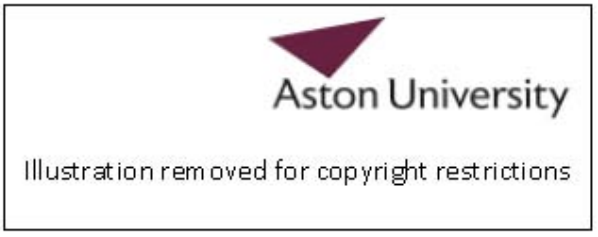

FIGURE 2.4: Figure (a) density-reachability, Figure (b) density-connectivity, [DBS, 2013].

In DBSCAN, new clusters are formed when there is an exceeding number of data points, defined by the $MinPts$, within a certain maximum spatial neighbourhood value, defined as the ε-neighbourhood (eps). A new cluster forms when the critical mass, $MinPts$, occurs within the eps neighbourhood. This is the point that defines the inner part of a cluster, namely a core point. Data points are categorised in three ways:

1. A core point is said to be so if it has more than a specified number of points ($MinPts$) within eps;
2. A border point has fewer points than $MinPts$ within eps, but is in the neighbourhood of a core point;
3. A noise point is any point that is neither of the above.

The algorithm grows its clusters by utilising two mechanisms; density-reachability and density-connectivity. Density-reachability takes two forms; direct and indirect reachability. A point q

is said to be directly density-reachable from a point p, if p is a core point and q is in p's ε-neighbourhood. Subsequently, a point q is said to be indirectly density-reachable from a point p, if there exists a direct density-reachability between q and another common point c, which is itself directly density-reachable to point p. Therefore, a point is indirectly density-reachable if there are chains of associations which are directly density-reachable to each other. Figure (2.4, a) depicts a chain of points, $p_1, ..., p_n$ where $p_1 = q$, and $p_n = p$, such that $p_i + 1$ is directly density-reachable from p_i. A pair of points, p and q, are density-connected if they are commonly density-reachable from a point o, as depicted in figure (2.4, b). This is repeated until all data elements are placed within a cluster.

The advantage of DBSCAN is that clusters can have arbitrary shape and size. The algorithm does not require the number of final clusters, as this is determined automatically. The algorithm works well with surrounding noise and can be supported by spatial index structures. The main disadvantage of DBSCAN is that it is sensitive to parameter settings and is unable to cluster well within large differences in density.

Other algorithms use density to their advantage and partition space into a finite number of cells within a grid [Maimon and Rokach, 2005]. This forms the basis of grid-based clustering.

Grid-based clustering: These are algorithms mainly used for spatial data mining. They are coupled within the same category of density-based algorithms, since density is a key operand [Grira et al., 2004]. Data is converted into a grid by using a density measure, where cells which have higher densities are retained, and isolated data items are removed. Clustering is subsequently applied to this grid. This approach is independent of the number of objects [Han, 2005] and data ordering [Berkhin, 2006]. Algorithms based on this approach include the DenClue [Hinneburg and Keim, 1998] and the CLIQUE [Agrawal et al., 1998].

In the algorithms presented above, the MDS has been used for spatial distribution. The next algorithm introduces the more structured graph-theoretic clustering.

2.2.4.4 Graph-theoretic Clustering approaches

In graph-based clustering, each data point is taken to be a node of a graph. Edges are paths between nodes, and are allocated a weight, which is representative of proximity (similarity). Nodes which are closer to each other offer a higher degree of similarity, versus nodes which are more distant. A cluster here is defined as a sub-graph [Schaeffer, 2007]. A number of mechanisms are applied to the graph in which clusters are formed, being either driven top-down

or bottom-up. In the former, a fully connected graph is used, which is split into smaller sub-graphs. In the latter each node is seen as a new sub-graph, and constructs larger sub-graphs.

In the top-down technique, sets of nodes branch off into new smaller clusters. This is determined by the minimum spanning tree (MST) of the graph [Zahn, 1971], which derives from the spanning tree of the graph. Given an undirected weighted graph, G, the spanning tree of this graph is a sub-graph, T of G, that includes all the nodes within G, and some or all of the edges of G. Thus, T is said to be a tree that spans G. The weight of the spanning tree is calculated as the sum of all weights within it. The minimum spanning tree of G, is the spanning tree which has the least weight [Chazelle, 2000]. Longer edges are eliminated from the graph forming new sub-graphs. By deleting the MST edges with the largest lengths, this in turn generates clusters which are denser to each other. A cluster is a sub-graph that remains connected after the removal of the longest edges of the graph [Jain and Dubes, 1988]. This is repeated until the number of clusters required is found.

In the bottom-up approach, as in the k-nearest neighbour (k-NN), smaller sub-graphs are iteratively merged into larger ones, until the desired number of clusters has been reached. The k-NN graph is defined as a weighted-directed graph, in which every node represents a single cluster and the edges represent pointers to neighbouring clusters. Every node has exactly k edges to the k nearest clusters [Franti et al., 2006]. At first a k-NN graph is created, where all nodes are converted into clusters. The distance between the k-nearest neighbours of every clusters is analysed and the smallest weight (distance) between clusters returns the best pair of clusters to be merged. Let these values be c_a and c_b, which once they merge are set to c_{ab}. Next the k nearest neighbours of this new cluster, c_{ab}, are calculated. This merging process is continued until M clusters are formed [Franti et al., 2006].

The main advantage of graph based methods is simplicity. However, these clustering algorithms exhibit problems including high time complexity, since they require a brute force calculation on all pairwise distances. This requires a $O(N^2)$ search before clustering can even commence.

2.2.5 The Relevance of Traditional Clustering Algorithms

This chapter has so far outlined the state of the art of traditional clustering algorithms, where it has been shown that collectively a valid clustering algorithm must have the following qualities; the ability to scale up to larger datasets with high dimensionality, discover arbitrary shapes without the lack of a given structure even when this data is cluttered with noise, and that the clustering output should not change irrespective of the sequence of data entry. Traditional clustering algorithms describe the principle methodologies by which unsupervised data clustering is

achieved, serving as a foundation upon which the MPACA is developed.

The MPACA is an unsupervised bottom-up approach, which contrasts heavily with the top-down hierarchical techniques described in section (2.2.4.1). Clustering in the MPACA is driven by ants merging into bigger colonies, where the term "cluster" is interchangeable with "colony". Clustering occurs when ants merge into colonies. The clustering process in the MPACA consists of an iterative approach. That is, ants can keep migrating from one colony to another, and merging colonies until a suitable colony for the majority of ants is found, thus sharing similarities with partitional clustering [section (2.2.4.2)]. The MPACA does not use any probability density function (PDF) to determine cluster allocations, distinguishing it from algorithms such as the Expectation Maximization (EM) [Dempster et al., 1977]. The MPACA instead bases its colony formation on ant densities. The more frequently ants encounter each other in a specific area of space, the more likely it is that ants belong to the same colony (cluster). This density driven approach shares similarities with other density based algorithms, section (2.2.4.3).

The MPACA uses a graph architecture to represent the problem domain, where node proximity in space also signifies a higher degree of relevance. That is, the MPACA is a graph-based clustering approach [section (2.2.4.4)]. Ants in the MPACA traverse this structure and lay pheromone to link nodes of higher relevance. The effect of this pheromone causes ants to form higher densities around certain regions at the expense of other regions. Thus, the occupation of certain areas of space by ants represents the clustering process.

Finally, once the cluster representative colonies are formed, the MPACA allocates the data elements to belong to each of these colonies using a weighted approach. This approach operates using cluster centroid values, similar to the K-Means [MacQueen, 1967].

The novelty of the MPACA is that it uses a nature-inspired bottom-up clustering algorithm, an approach which is distributed and decentralised and returns results which are comparable to both traditional and other swarm based approaches. The MPACA introduces another major difference when compared with traditional clustering algorithms. In the MPACA clustering is not achieved over the original objects but by ants within a colony and the collective properties within each colony as defined by the values carried by the ants. The operational differences are easier to appreciate after the MPACA is described in section (3.10.2).

Clustering is a problem which has been tackled by a multitude of algorithms, including those inspired by nature. In many cases nature-inspired algorithms, like swarm based techniques, are decentralised, scalable and are ideally suited for unstructured data. The next section reviews representative techniques under which the MPACA is categorised.

2.3 Swarm Intelligence

Swarm intelligence (SI) as introduced in section (1.3), is the collective problem-solving behaviour of groups of entities, which interact with each other and their environment to create interesting emergent phenomena ([Zhu and Tang, 2010], [Chu et al., 2011], [Blum and Merkle, 2008], [Yang et al., 2013], [Lim and Dehuri, 2012]).

A key aspect of SI is that these are population based approaches. Other population bio-inspired methods exist, with genetic algorithms (GA) being emblematic. GA share key overlapping features with SI. Introduced by Holland [Holland, 1973], these have been applied to a multitude of problems [Thengade and Dondal, 2012]. GA operate on the principle of survival of the fittest, where candidate solutions which offer the best results are given the ability to procreate their solutions further. The key operands are the crossover and mutation functions. The crossover function creates new candidate solutions by selecting the best parents, whilst the mutation function allows the next generation of candidate solutions to randomly adjust the current maximal solution. This process counteracts the problem of local maxima and is a popular technique. Despite being population based and used in a widespread manner, GA do not share all fundamental elements of SI which have been identified and are of interest to this thesis, namely; self-organisation [section (1.3.1)] and stigmergy [section (1.3.2)]. On the other hand, the iterative improvement at each generation be only be considered loosely as a positive feedback in SI terms [section (1.3.3)], as there is no element of reinforced recruitment between agents. Hence, GA are not investigated further.

2.3.1 General Swarm Intelligence Approaches

Many prevailing SI approaches exist, as follows; Bacterial Communities [Flikkema and Leid, 2005], [Eker et al., 2003], Stochastic Diffusion Search (SDS) [Bishop and Torr, 1992] ([Bishop, 1989]), Artificial Immune Systems (AIS) [de Castro and Von Zuben, 1999], [Analoui et al., 2010], [Morrison and Aickelin, 2002] and DNA computing [Adleman, 1994] [Watada and binti Abu Bakar, 2008], [Dong et al., 2009], [Ono, 2009]. SI techniques do not exclude non-biologically inspired approaches, such as the gravitational search algorithm [Hsiao et al., 2005], Intelligent Water Drops (IWD) [Shah-Hosseini, 2007], [Kamkar et al., 2010], [Duan et al., 2008] and River Formation Dynamics (RFD) [Rabanal et al., 2010], [Rabanal et al., 2008],

The focus of this thesis is to explore and contrast the most prevalent algorithms in SI literature, which excludes a deeper analysis of the above mentioned algorithms. Within the vastness of SI literature, the major methods can be categorised under two families of algorithms;

those traditionally considered to be SI techniques like Particle Swarm Optimisation (PSO) [Kennedy and Eberhart, 1995], and Bio-Inspired SI techniques, of which Ant Colony Optimisation (ACO) [Dorigo, 1992] and Bee Colony Optimisation (BCO) [Teodorović and Dell'Orco, 2005], [Teodorovic et al., 2006] are flagship algorithms. Three algorithms that have been chosen to be further explored based on their significance to the MPACA are PSO, BCO and ACO, sections (2.3.2, 2.3.3.1, 2.3.4).

2.3.2 Particle Swarm Optimisation

Particle Swarm Optimisation (PSO) [Kennedy and Eberhart, 1995] is aimed at producing computational intelligence via social interaction, rather than via a purely individual cognitive ability. PSO is influenced by the flocking of birds in search for corn [Heppner and Grenander, 1990].

In PSO candidate solutions called particles, are initially placed in the problem search space, with the population of such particles constituting a swarm. Each particle evaluates an objective function at its current location, this evaluates the quality of the candidate solution generated. The terms "objective function" and "fitness function" are used interchangeably in literature. The search space is represented by a network topology which allows particles to move from one area to another. A particle keeps track of the topological coordinates associated with the best solution (fitness) that it has achieved so far, its personal best, p_{best}.

Since the action of an individual particle is equivalent to random movements in space, there is an inherent difficulty at solving optimisation problems. This necessitates interaction with other particles (the swarm) to gather directional guidance. Particles cooperate by exchanging information on what they have discovered about visited areas. Particles which have achieved best results (optimum particles), known as the global best, g_{best}, represent the best candidate solution found (graded by the fitness function) within the neighbourhood topology of that particle. Particles fly through this topological problem space. Each particle adjusts its position by calculating a bearing, based upon the following considerations; the current position and velocity, the distance between the current position and the p_{best}, the distance between the current position and the g_{best}, together with some other random perturbation. A bearing calculation is based on the following vectors:

1. The $\overrightarrow{x_n}$, or the current position of the particle in a search space;
2. The $\overrightarrow{p_n}$, previous best, or the best solution found so far by the particle; and
3. The $\overrightarrow{v_n}$, velocity gradient, a gradient for which the particle will travel in, if undisturbed.

In the above, n represents the n-th interval, as well as two fitness values, the fitness of the $\overrightarrow{x_n}$, and

the fitness of the $\overrightarrow{p_n}$. At each iteration the solution is evaluated. If the solution for the current problem found is better than what has been found so far, the previous vector ($\overrightarrow{p_n}$) is updated. The value of the best function result so far is stored in a variable, p_{best_n}, which is then compared with later iterations. The aim is to keep finding better positions and update $\overrightarrow{p_n}$ and p_{best_n} accordingly. The particle moves towards a new point by creating a new $\overrightarrow{x_n}$, which is the summation of the current $\overrightarrow{x_n}$ and velocity vector v_n; $\overrightarrow{x_n} = \overrightarrow{x_n} + \overrightarrow{v_n}$. Due to the nature of the topological layout and the manner in which particles interact with their neighbours, progressively improved solutions are achieved. This continues until the minimum error criteria is reached, or the maximum iteration occurs. After a sequence of iterations, the collective and emergent behaviour will guide the swarm towards an optimal fitness function.

PSO is widely accepted as a powerful global optimisation algorithm because of its simplicity in implementation and low constraints on the environment [Poli et al., 2007]. PSO is a benchmark algorithm in SI and has numerous applications in literature, however the interest of this thesis are PSO applications applied to clustering, which include [Niknam et al., 2009], [Sharma and Omlin, 2009], [Hasan et al., 2011], [Chuang et al., 2012], and [Van Der Merwe and Engelbrecht, 2003] amongst others.

Relevance to the MPACA

PSO introduces the notion of a population-wide mechanism applied to problem solving, where each particle is a basic entity, with limited knowledge of its surroundings. Even if the swarm itself is a set of particles with limited top-down coordination, it is still not a purely decentralised technique, since in order to compute the next movement particles need to know not only their current best, but also the global best, and this can only be retrieved by explicit synchronisation with other particles. Another crucial limitation is the lack of the influence of stigmergy in PSO, since direct communication between particles is required through synchronisation.

2.3.3 Bio-Inspired SI Algorithms

The primary focus of Bio-Inspired algorithms are insect based algorithms, rather than particles or actions of automata. The field of Bio-Inspired SI can be further subdivided into two insect algorithms, namely Bee Colony Optimisation (BCO) and Ant Colony Optimisation (ACO).

2.3.3.1 Bee Colony Algorithms

Nobel Laureate von Frisch decoded the language used by honey bees to communicate the location of a food source, coining the term “waggle dances”, and by means of a series of entomological experiments showed how bees have their own colour vision and are attracted by their

hive mates [von Frisch, 1993]. Honey bees, just like their fellow ant cousins, are self-less creatures and are capable of exploring and exploiting large areas simultaneously [Teodorović and Dell'Orco, 2005], [Teodorovic et al., 2006], [Nikolić and Teodorović, 2013].

In bee colonies, scout bees forage for food sources and upon finding such source they return to the hive and perform the waggle dance to share this information, and attempt to recruit other bees in following them. The waggle dance contains information relating to the direction of flower patches, distance from the hive and a quality rating (fitness). Through this dance, other bees direct themselves towards the best food source. Due to the distributed nature of the algorithm, there are always bees which preform random foraging missions. The biological bee colony resembles the biological ant colony in the way they cause a positive feedback to occur. In a bee colony, after bees perform the waggle dance inside the hive, the scout bee returns to the flower patch accompanied by more bees, that repeat this action causing a positive feedback loop to occur.

In computing terms, scout bees are seen as candidate solutions. These candidate solutions are comparatively evaluated against other possible solutions, out of which the candidate solution deemed best is assigned more scouting resources, causing the search effort to focus within a specific result of a candidate solution. The additional random scouts that are still present further avoid stagnation in local maxima, as they are free to seek other solutions without any influence.

BCO is applied to clustering problems [Shanthi and Amalraj, 2012], [Nesamalar and Chandran, 2012], [Oleynik et al., 2010]. Notwithstanding, the lack of stigmergy and the more intricate bee-to-bee communication mechanisms than exist in ants, both BCO and ACO algorithms are composed of multiple homogeneous individuals which interact locally on a set of simple rules. Through self-organisation and positive feedback these algorithms are capable of tackling NP-hard computational tasks. In a similar fashion to PSO and BCO, ACO is population based. However, ACO operates on a fundamentally different metaphor, since it does not make use of direct peer-to-peer communication present in both PSO and BCO. Central to ACO is the indirect communication mechanism that uses pheromones.

2.3.3.2 Ant Algorithms

Omnipresent in Bio-SI literature are models relating to various ant metaphors, ranging from self-aggregating ants [Azzag et al., 2003], to ants organising cemeteries [Deneubourg et al., 1990b], ants using colonial odour recognition [Labroche, 2003], the behaviours of heterogeneous ant types [Admane et al., 2006], pheromone aggregation mechanisms as deposited by ants [Ghosh et al., 2008], and lastly, crucial to this thesis, the ant metaphor based on ant foraging theory.

Many ant species use pheromones to communicate with fellow colony members. This pheromone acts as a trail and marks the path from the nest to the food sources and back. This is the core metaphor behind this thesis.

2.3.4 Towards the Ant Colony Optimisation Meta-Heuristic

The focus of Ant Colony Optimisation (ACO) is the meta-heuristic search, section (1.3.4), a mechanism partly driven by pheromone (the trail) and partly by a heuristic function (the visibility). The latter function is a local optimisation function which aims to improve solution quality at each ant movement. For example, in the travelling salesman problem (TSP) where the overall aim is to minimise total tour distance, the heuristic function is defined as the inverse of the distance function. Dorigo first applied this to the TSP [Dorigo, 1992], where it is experimentally shown that ACO algorithms can solve NP-hard problems [section (2.2.1.1)]. Numerous ACO applications have been introduced. The elements of ACO applications that distinguish their operations from each other are:

1. The mechanism by which an ant moves from one node to the next in graph space, referred to in literature as the ant movement or transition function;
2. The pheromone update rules; and
3. Quantity of pheromone that is deposited.

Further terminologies complement section (1.3.4), as follows:

- An iteration is a step within the algorithm at which all ants perform a tour.
- A traversal of an ant between one node and another over an edge is called a transition.
- Local pheromone updates take place when they occur at each of these traversals, whilst global pheromone updates occur only at the end of the ant tour, thus updating a path which constitutes a collection of traversed edges with pheromone.
- The global best is the ant tour which returns the best result so far in terms of optimisation.
- The iteration best is the ant tour which returns the best result within the iteration.

The **Ant System (AS)**, also known as the ant-cycle, is the first ant colony optimisation algorithm in literature [Dorigo, 1992], [Dorigo et al., 1991b], [Dorigo et al., 1991a], which apply the TSP as the de-facto example. In the ant-cycle pheromone updates are effected at the end of a cycle (global pheromone updates). Within the AS, the heuristic function is expressed in terms of the inverse of the Euclidean distance [section (2.2.2.2)] of the edge. After every iteration, that is after each and every ant has completed its tour consisting of several edge traversals, pheromone values are updated by all the ants. The pheromone quantity to be deposited is calculated as in

equation (2.21), where the constant value Q, is divided by the length of its tour, hence a shorter tour results in a higher value to be deposited, and on the contrary a longer tour results in a lesser value. This value is equally distributed over the tour taken by the ant, with the tour consisting of multiple edge segments. The pheromone value, τ_{ij}, which is associated with the edge joining nodes i and j representing cities i and j, is updated as in equation (2.20). This equation includes the complement of the pheromone value, ρ, which represents the evaporation taking place on the existing edge τ_{ij}. For each ant present in the system, an amount of pheromone is also added to this edge as follows:

$$\tau_{ij} \leftarrow (1-\rho)\cdot\tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}(t) \tag{2.20}$$

where ρ is the evaporation rate, m is the number of ants and $\Delta\tau_{ij}(t)$ is the quantity of pheromone laid on edge (i,j) by ant k at time t, which follows from equation (2.21):

$$\Delta\tau_{ij}(t) = \begin{cases} Q/L_k, & \text{if ant } k \text{ used edge } (i,j) \text{ in its tour} \\ 0, & \text{otherwise} \end{cases} \tag{2.21}$$

where Q is a constant and L_k is the length of the tour constructed by ant k at time t. During the construction of any one solution, ants choose which node to visit next by making use of a transition mechanism which is core to ACO literature. When ant k is in node i and has so far constructed partial solution s^p, the probability of choosing node j is given by equation (2.22):

$$p_{ij}(t) = \begin{cases} \frac{\tau_{ij}^{\alpha}\cdot\eta_{ij}^{\beta}}{\sum_{c_{il}\in N(s^p)}\tau_{ij}^{\alpha}\cdot\eta_{ij}^{\beta}}, \text{if} & c_{ij}\in N(s^p), \\ 0 & \text{otherwise} \end{cases} \tag{2.22}$$

where $N(s^p)$ is the set of feasible components, c, on edges (i,l), where l is a node which has not yet been visited by the ant k at time t. Parameters α and β control the relative importance of the pheromone or trail versus the heuristic function or visibility. The visibility function is calculated by η_{ij}, as in equation (2.23):

$$\eta_{ij} = \frac{1}{d_{ij}} \tag{2.23}$$

where d_{ij} is the Euclidean distance between nodes i and j [Dorigo et al., 2006] or the heuristic component.

As a side note, if the amount of pheromone deposited Q is kept constant, and the amount of ants increase to accommodate larger domains, the influence of the α parameter can be adversely impacted by this scaling increase, as the quantity of pheromone deposited would be larger.

This algorithm allows ants to deposit pheromones only at the end of each tour, whilst also allowing an unrestricted quantity of pheromone to be deposited. This positive reinforcement, whilst required for the general scope of the algorithm, can cause local maxima to occur, as ants would repeatedly reinforce a specific path. This raises two considerations; should all ants be allowed to reinforce paths by pheromone deposition, and if so, what is the maximum amount of pheromone that should be allowed on each edge? In such a case a capping on the amount of pheromone deposited would be more coherent. These considerations led to improvements based on the concept of the "best ant".

The best ant, that is the ant which performs the tour which returns the best results. The elitist strategy [Dorigo et al., 1996] allows this best performing ant to add an additional amount of pheromone, therefore adding a bias towards the best solution. An extension to the elitist strategy is the AS_{Rank} [Bullnheimer et al., 1999], where the tour quality produced by the ants is ranked, and only the n th best ants together with the overall global best are allowed to deposit pheromone on the traversed paths. A further extension is presented by the Best-Worst Ant System (BWAS) algorithm [Maniezzo, 1999], in which only the best-so-far ant is allowed to deposit pheromone, whilst the worst solution is subject to further pheromone decrease.

The **Ant Colony System (ACS)** is a further contribution in literature in which a local pheromone update is also included. This differs from global pheromone update which is applied only once the ant tour has been completed. Local pheromone updates occur whilst the ant is moving from one node to another, at each constructed step. This is given by equation (2.24):

$$\tau_{ij} = (1-\varphi)\cdot\tau_{ij} + \varphi\cdot\tau_0, \tag{2.24}$$

where $\varphi \in (0,1]$ is the pheromone decay coefficient and τ_0 is the initial value of the pheromone.

The main goal of the local update within the ACS is to increase the diversity of the search performed during an iteration. A decrease in pheromone concentration on the edge which has just been traversed encourages other ants to choose other edges, and thus increases the variation, a measure which reduces early convergence [Dorigo and Gambardella, 1997b], [Dorigo and Gambardella, 1997a], [Gambardella and Dorigo, 1996].

The process behind the ACS consists of each and every ant constructing a solution by making stochastic decisions, in response to a deposited pheromone. The colony then reinforces decisions in the construction process according to its successes by adding pheromone. This pheromone

can also decay (evaporate) to mitigate against poorer decisions.

$$p_{ij}(k) = \begin{cases} argmax_{c_{il} \in N(S^p)}\{\tau_{il}\eta_{il}^b\}, \text{if} \quad q \leq q_0 \\ \text{equation} \quad (2.22) \qquad \text{otherwise} \end{cases} \tag{2.25}$$

Another important difference between the ACS and the AS is in the decision rule used by the ants during the construction process. The transition function, or ant movement, in the ACS uses what is called as pseudo proportional rule, determining the probability of going from node i to node j. This depends on the random variable q, uniformly distributed over $[0,1]$, and a parameter q_0 as per equation (2.25). The ACS, does not include an element of capping the quantity of pheromone on each edge. The next algorithm reverses this limitation.

The **MAX-MIN Ant System (*MM*AS)** introduced by Stützle and Hoos, is considered to be the state of the art ACO algorithm [Stützle and Hoos, 2000]. The *MM*AS increases the search capability of the standard algorithm by combining exploitation with exploration of the search space, and by imposing bounds to the pheromone update level, thus helping to avoid premature stagnation. The term premature stagnation means that the edges connecting nodes have such a high amount of pheromone, that ants will most likely always choose the same edges. In doing so they create a sub-optimal path, which might lead to a local maxima.

The *MM*AS differs from the ACS/AS as follows:

1. Only the best ant tour is allowed to updates the pheromone trails;
2. The value of the pheromone is bound within a minimum and maximum update range $[\tau_{min}, \tau_{max}]$, a process which avoids stagnation;
3. The pheromone trails are deliberately initialised to τ_{max}, achieving higher exploration of solutions at the start of the algorithm.

In the *MMAS* the pheromone update rule is implemented as in equation (2.26):

$$\tau_{ij} \leftarrow [(1-\rho)\cdot\tau_{ij} + \Delta\tau_{ij}^{best}]_{\tau_{min}}^{\tau_{max}}, \tag{2.26}$$

where ρ represents the evaporation rate at each iteration and τ_{max} and τ_{min} are respectively the upper and lower bounds imposed on the pheromone quantity that can be deposited. $\Delta\tau_{ij}^{best}$ is defined in equation (2.27):

$$\Delta\tau_{ij}^{best} = \begin{cases} 1/L_{best} & \text{if } (i,j) \text{ belongs to the best tour,} \\ 0 & \text{otherwise,} \end{cases} \tag{2.27}$$

where L_{best} is the length of the tour of the best ant, which can be either the best tour found in the current iteration, known as the *iteration best*, L_{ib}, or the best global solution found, known as the best so far, L_{bs}. Guidelines exist on choosing suitable τ_{min} and τ_{max} [Stützle and Hoos, 2000].

2.3.4.1 Relevance and novelty to the MPACA

The previous section has outlined the standard ant algorithms and referenced the state of the art ACO algorithm. The AS and its subsequent derivations share the key movement operator with the MPACA. The core operator is the collective action of ants moving from one node to another within a graph, together with the laying of pheromones used to strengthen edges of higher relevance over other edges. The relevance of the connection is dependent on a similarity function. Key variations between the methods reviewed so far revolve around the mechanism which the ants in the MPACA use to move around in graph space, tackled further in chapter (3).

Typical ACO implementations use TABU lists, an exclusive list which keeps track of the sequence of nodes that have been visited within each ant tour. This TABU list ensures that within each solution the ant visits a node just once. This is the general ACO approach, and variations exist in literature (e.g. [Bertelle et al., 2006]) which do not use such a strict TABU list. The MPACA offers yet another variation; whilst it still uses a TABU list, this list is restricted to size one, since there is no concept of ant performing tours. The MPACA stores only the last node which has been visited, naturally limiting the ant from visiting the node it just visited. Ants in the MPACA do not perform any tours, and respectively there is no fitness function to determine the amount of pheromone that should be laid down. Pheromone is laid down in a way that edges which connect interesting nodes are reinforced. The absence of a fitness function enables the MPACA to be considered truly decentralised and distributed.

The MPACA employs a more innovative mechanism for ant movement. This movement mechanism does not require any foresight about the potential nodes that can be visited, further enhancing decentralisation. It removes the need to use a meta-heuristic function as per equation (2.22), which requires both pheromones on edges (trail) and a heuristic function (visibility). The leading contributor to the selection process is the pheromone present on each edge, which is used in conjunction with a weighted random selection. Therefore, the MPACA is part-random, part-pheromone driven. The probabilistic process means that an ant is most likely to choose the path with the most pheromone matching its features. Any edge can be selected, yet the edges with higher pheromone levels are more likely to be selected than others.

All the above mentioned changes and their subsequent benefits towards the decentralisation of

the process, lead to the primary benefit of the MPACA. That is, that there is no longer the need to implement a direct fitness function; it is the action of the collective behaviour of ants which determines this action. In the MPACA pheromone is always deposited at the local level where there is no centralised optimisation controller, with pheromone being deposited as soon as ants traverse the edge. This coupled with the lack of a fitness function, implies that the algorithm is much more decentralised than other ACO implementations.

2.4 Conclusions from SI literature

Bio-SI algorithms have been introduced and defined as population based, bottom-up approaches which allow unsupervised learning without any preconceptions, being distributed and asynchronous, and allowing scalability. The extended population base allows redundancy, thus the overall robustness and resilience of the solution is improved. This makes them ideal to be applied to the clustering domain.

2.5 Ant Algorithms and their Application to Clustering

2.5.1 Fundamental Operators Behind the Chosen Models

A review of ant algorithms and their application is given by Jafar and Sivakumar [Jafar and Sivakumar, 2010], from which it is apparent that there are numerous confusing ant models and terminologies. This section describes ant clustering algorithms by categorising them into four types. Comparisons between the different types and also with the MPACA is made on a set of criteria defining each model. This makes it easier to show how they are similar or different and, in particular, how the MPACA is positioned within them. These operational rationales are subdivided as follows:

- Type I - Knowledge Structure Forming Ants, section (2.6.1);
- Type II - Ant Aggregations and Ants' Self-Aggregation, section (2.6.2), where clustering is achieved by ants as they aggregate objects or themselves as representative of objects, within a 2D grid;
- Type III - Chemical Recognition System of Ants, section (2.6.3), where ants determine colony membership by exchanging pheromones within a virtual architecture;
- Type IV - Ant Colony Optimisation Algorithms, section (2.6.4), which is also the category which the MPACA is best categorised under, where ACO inspired techniques are used within graph space to formulate clustering solutions.

2.5.1.1 Type I - Knowledge Structure Forming Ants

Various ant metaphors are linked to self-organisation. Mechanisms which found their way in clustering algorithms include; cemetery and larvae sorting [Deneubourg et al., 1990b], colonial odour [Labroche, 2003] and ant foraging [Dorigo, 1992]. This typology applies to another biologically observed behaviour in real ants, that is the ability of ants to build mechanical structures using self-assembly, such as the building of chains of ants with their bodies in order to link leaves together, as depicted in figure (2.5), [Akbar, 2008]. In nature different types of chains are observed, from crossing an empty space to building nests [Lioni et al., 2001].

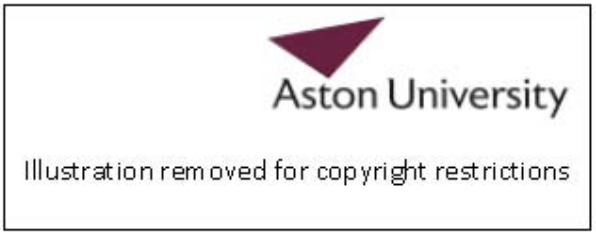

FIGURE 2.5: Fire ants coalesce together to form a bridge with their bodies to reach a distant leaf in Jakarta, Indonesia, [Akbar, 2008].

This structure-forming metaphor in ants is used to partition data, by allowing ants to mould around data giving it more structure. This structure forms a hierarchical clustering technique. Algorithms falling under this category include the AntTree algorithm [Azzag et al., 2003]. Each ant represents a data element that is to be classified. Ants subsequently fix themselves on supports, with other ants attaching themselves to the already fixed ants, thus allowing the creation of bridges. Artificial ants build a tree according to the similarity between the data they represent. This mechanism is used to build a hierarchical knowledge structure through these ant connections.

2.5.1.2 Type II - Ant Aggregations and Ants' Self-Aggregation

The ability of ants to aggregate objects and the alternative ability of ants to aggregate themselves serve as key analogies to the clustering process defined next. This concept originates from the work of Chrétien who investigates the *Lasius niger* ant and the organisation of cemeteries [Chrétien, 1996]. Similar experiments are also undertaken by Deneubourg using the *Pheidole pallidula* ant, figure (2.6) [Deneubourg et al., 1990b]. Another biological analogy is that of brood sorting, as observed by Franks and Sendova, in the ant *Leptothorax Unifasciatus*, where workers of this species gather the larvae according to their size [Sendova-Franks and Franks, 1995].

FIGURE 2.6: From (a) to (d), a sequential clustering task of corpses performed by a real ant colony. 1500 corpses are randomly located in a circular arena with radius = 25 cm, where Messor Sancta workers are present. The figure shows the initial state (a), 2 hours (b), 6 hours (c) and 26 hours (d) after the beginning of the experiment, [Chrétien, 1996].

Deneubourg coins the clustering methodology inspired by such activity as the Basic Model (BM). In the BM, a population of ants moving randomly on a grid, pick up or drop off corpses (data points) so as to cluster them. These drops occur only when the similarity with objects in the immediate neighbourhood exceeds a certain threshold. The ants operate according to local rules and have only local perceptual capacities. Notwithstanding this limitation, and the decentralised nature of the ants, the ant colony itself demonstrates global and coordinated control. Distinguishing this typology is the 2D architecture in which ants operate, and the ability of ants to aggregate either objects or themselves within a particular sub-space. Various models follow this typology, such as the eponymous Lumer and Faieta (LF) model [Lumer and Faieta, 1994]. This sub-category is called the **Standard Ant Clustering Algorithm (SACA)**.

This typology can be extended to include other ant driven self-aggregation metaphors, such as the search for safety in the Ant Sleeping Model (ASM), later implemented as the Adaptive Artificial Ant Clustering (A^4C) algorithm [Chen et al., 2004]. This is inspired by the aggregation of ants towards safer areas in a given space. A safe area represents an area where the neighbouring ants share many similarities. Ants depart unsafe locations, in search of safe places where to sleep. The expression "birds of the same feather flock together" is a fitting statement for such aggregations. Algorithms classified under this type utilise the ants themselves to generate cluster information. This sub-category is called **Self-Aggregation within a 2D Grid**.

The spatial transition probabilities in the SACA and the ASM allow ants to needlessly explore regions without interest. Further extensions include the incorporation of bio-inspired spatial transition probabilities to compensate for such a lack of orientation by the use of stigmergy [see section (1.3.2)] [Chialvo and Millonas, 1995]. This latter sub-categorisation revolves around the

aggregation of pheromones that causes the conglomeration of ants, or clustering behaviour, in a species and brings individuals into closer proximity. The ACLUSTER algorithm [Ramos and Pina, 2002], [Ramos and Merelo, 2002] is a prime example. In this algorithm pheromones are deposited into a structure accessible by all ants within the immediate proximity. This avoids the need of short-term memory based strategies, the use of several heterogeneous ants running at different speeds and other mechanisms frequently associated with the SACA inspired models. The ant transition probabilities depend on the spatial distribution of pheromone across the environment. The ant determines its location depending on the change in pheromone from its current cell towards the most likely cell it can travel to, combining both factors of geo-location together with the pheromone present in the environment. Thus, areas with higher pheromone are explored more frequently than areas with lower pheromone. This pheromone orientation allows ants to avoid roaming in areas void of any objects, a measure which enhances the exploration capabilities of the ants, and reduces time consuming movements.

Building upon the ACLUSTER, one finds other more recent algorithms in literature, these include the Aggregation Pheromone Systems (APS) and the Aggregation Pheromone Clustering (APC) [Ghosh et al., 2008], [Tsutsui et al., 2005]. In these algorithms, ants are used as proxies of objects that move closer towards each other depending on the pheromone being distributed. This sub-category is called **Ant Aggregation through Pheromone in a 2D Grid**.

Other ant metaphors use additional pheromones to exchange information, therefore not relying solely on location information, including pheromones that determine nest belonging.

2.5.1.3 Type III - Clustering Inspired by the Chemical Recognition System of Ants

Labroche *et al.* [Labroche, 2003], [Labroche et al., 2002a], [Antoine et al., 2008], [Labroche et al., 2002b] introduce the ANTCLUST algorithm, which builds around the idea of colonial or "Gestalt" odour theory. This theory exploits the ability of ants to use a colony wide odour to discriminate between nest-mates and non-nest-mates, a recognition based on phenotype matching. The ANTCLUST algorithm models this chemical recognition in such a way that the ants of the same nest share a similar odour, and gather to form a class [Labroche, 2003].

Type III differs from Type II as in this mechanism ants only operate within a virtual environment, without the need of a physical operating space. Ants exchange knowledge information on colony belonging by the interaction of pheromone traces. In this typology, colony membership is key to the clustering process.

There are two key terms to be noted; the label and the template. The label is the individual odour

FIGURE 2.7: Principles of the ANTCLUST. Labels and Templates are represented in a 2D-space for a better understanding. In figure (a) ants have no Label and are just described by their Genetic odour. In figure (b) the first labels have been computed by the algorithm. In figure (c) the final classification groups in the same nest, the ants that share a similar Label, [Labroche et al., 2002a].

which the ant is imprinted with, whilst the template is the odour which the ant is interested in. The template indicates the odour that nest mates should have on their cuticle for a positive match to occur. Over time, labels and templates change accordingly. Ants gather around a finite number of nests, which are representative of the final classification groups. Each nest represents ants that share a similar label [Labroche et al., 2002b], and are effectively ants which share a higher similarity to each other, figure (2.7), thus resulting in a collective clustering algorithm.

The use of pheromones which is most relevant to this thesis, is that inspired from the ant foraging mechanism. This is the founding mechanism behind ACO.

2.5.1.4 Type IV - Clustering using Ant Colony Optimisation Algorithms

Algorithms categorised under this type involve ants that lay down pheromone onto a graph structure to encourage other ants to increasingly choose edges with higher pheromone. In this typology, graph space is crucial and is used to represent the problem which is unlike previous types, where data elements move around in 2D grid space. There are a number of approaches which fall under this category, including multi-objective, multi-colony and multi-pheromone implementations using ACO.

In the field of ACO based clustering, two predominant approaches exist; either (i) the conversion of the clustering problem into an optimal assignment problem, alternatively (ii) the application of ACO for spatial clustering within a graph.

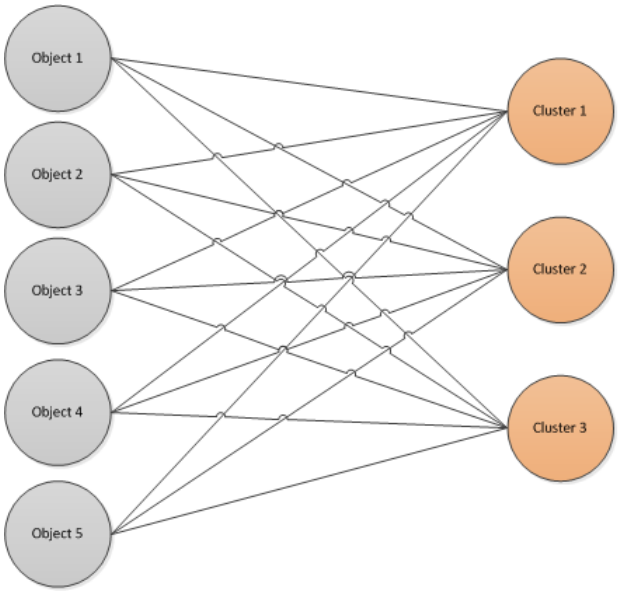

FIGURE 2.8: Objects are placed into clusters 1,2, or 3 by ant tours. Depending on the overall placement quality, edges between objects and clusters are updated, (author self).

In the former, nodes are allocated to clusters depending on ant traversals between nodes representing objects and nodes representing cluster centres [figure (2.8)]. Thus the problem is converted into graph space by having a sequence of objects, N, each having edges that reach any cluster K. Ants are used to allocate objects to clusters, with the number of clusters being known, but the value of the cluster centre still being unknown. Clustering is achieved via an iterative improvement process, where at each iteration potential cluster configurations, or solutions, are generated and are evaluated through a fitness function. Depending on the overall fitness of the solution, improved or degenerated, an amount of pheromone is deposited on edges connecting nodes, reinforcing certain cluster configurations over others. The fitness function aims to minimise the total cluster error [Shelokar et al., 2004b], [Shelokar et al., 2004a]. This process eventually leads to increased likelihood of an improved overall solution.

The alternative approach builds on the notion of spatial graph-based clustering [section (2.2.4.4)]. Spatially distributed nodes within a graph are allocated to one cluster over another depending on their inherent composing values and the similarity that exists between them, the key difference being that this is achieved without the need of an overall fitness function. That is, nodes are re-grouped and form sub-graphs which are effectively the clusters sought. The edges that connect nodes are additional distorted by pheromone traces increasing the likelihood of certain node connections over others. That is, nodes within the graph itself are set to belong to one cluster or another. This mechanism is further separated into two approaches. The first approach uses multiple ant colonies competing for colouring nodes within a graph, where each colony has a specific colour, and nodes get coloured by the population of ants present on them. Nodes

adjust by changing colour [Bertelle et al., 2006], and the final settlement forms the clusters. The second approach uses ants to populate graph space in a discriminatory manner, that allows certain cluster areas to form according to node distribution and proximity. These algorithms assume that the spatial distribution of nodes is directly built on the properties of the data points. Therefore, spatial configurations are indicative of cluster formations. Ants populate specific areas of the graph in various densities, a process which further expedites the formation of clusters [Tsai et al., 2004].

A further application of ACO is in the field of classification and data mining, with its flagship algorithm being the Ant-Miner [Parpinelli et al., 2001]. Graph space is once more used, with the added difference that in this configuration nodes do not represent objects to be clustered, but instead each node represents categorical attributes within a larger sequence or logical combination. The logical combinations are used to represent rules for a specific condition. For every initial condition a number of rules are generated. This is achieved by having ants traverse nodes by selecting an attribute value amongst a possible set of attribute values. The choice of attribute value is driven by the standard ACO meta-heuristic, which considers the quality of the next move (heuristic) and the amount of pheromone present on the edge connecting it (trail). The sequence of traversed attribute value pairs constitutes partial rules. On completion of a number of traversals, rules are pruned depending on their successful approximation and quality of the initial condition. The rule which generates the best result has the nodes within its attribute values and the edges that link these respective nodes incremented with further pheromone. Thus, in principle a classification rule is discovered via ant traversals over a rule domain.

A number of ant algorithms belonging to each of the mentioned typologies are used to highlight the variations between such algorithms and the MPACA. To standardise the way this evaluation is executed, the following criteria are used.

2.5.2 Comparison Criteria

Problem Space (architecture): Section (2.2.2.1) introduces three spatial architectures; the 2D grid, the MDS and graph space. Graph space does not always directly correlate to MDS, as is the case in standard ACO optimisation clustering [type IV, (i)], where graph space is used to contain the solution space. The spatial arrangement used is irrespective of its underlying properties, where edges are created between objects and possible cluster locations, as in figure (2.8).

Ant Movement: Spatial arrangements allow ants to execute movement (ant transition). Two mechanisms are most relevant to this research; random movement, where ants are not influenced by external peer-to-peer or environmental factors, and meta-heuristic movement, in which external factors contribute to the ants' decision on movement selection.

Interpretation of clusters: Clustering algorithms reorganise the given dataset into subdivisions of higher similarity. Interpreting the final cluster can be somewhat challenging. In a hierarchical structure, as in the AntTree, clusters are defined as the sub-trees which are connected to the primary node. The subsequent datasets within each sub-tree, thus contain the separate cluster values.

In some cases the clustering process is not fully completed by the action undertaken by the ants. A case in point is clustering within a grid, where ants either aggregate themselves or aggregate data elements into clusters. In this case, even if the 2D space returns a clear spatial separation amongst clusters, there still needs to be a mechanism to extract the cluster value, and include or eliminate outliers. In these circumstances, additional algorithmic parses are performed, as the K-Means [Kuntz et al., 1998], and pattern recognition mechanisms are employed to identify the size of clusters within the grid [Martin et al., 2002]. Other mechanisms include agglomerative clustering applied over the resultant grid.

Other cluster interpretations include the following:

- In ant algorithms where colony membership is used as a discriminant, as in the ANTCLUST, the final colony is used to define the cluster. In such a case, a colony is synonymous with a cluster.
- In ant algorithms where ACO is used as an optimisation tool, the cluster is defined as the configuration of data elements within the known clusters, which minimises the Error Sum of Squares (SSE).
- In graph spatial clustering, sub-graphs are formed when edges are cut-off from the main graph. This edge connectivity is determined by a process where ants reinforce edges between nodes and create areas of higher pheromone intensity on edges which link closer (similar) nodes. Depending on a population average applied to all edge weights, edges with a weight lower than a threshold are cut off from the main graph.

Colony usage: Although the term colony is frequently used in ant literature, it does not necessarily imply that the colony itself has any computational significance, since in many cases colonies only serve as the operating basin collecting ants together. Evident cases falling under this fold are all ant algorithms which use a single colony. On the other hand colonies, in

multi-colony algorithms, can be split into two categories:

1. Distinct ant colonies, possibly in distinct architectures, having limited inter-colony interaction at specific synchronisation points;
2. Multiple ant colonies interacting within the same architecture, heavily influencing each others' actions, where pheromones used by a colony directly influence other colonies. In addition to this category, ants may opt to form new colonies in reaction to the ongoing processing. Thus, the final set of colonies is the emergent behaviour.

Distributed versus Centralised: In any multiple entity algorithm, coordination and communication exchange mechanisms are crucial. Information exchange determines the level of centralisation, where lesser coordination and more asynchronous communication leads to a higher level of decentralisation and vice-versa. The level of centralisation is dependent on the overall controlling mechanism and the co-ordination it exerts.

Communication between Ants: Centralisation, or the lack of it, is intertwined with the ability of ants to interact with their peers. This is considered to be explicit when direct ant communication is used, and implicit if indirect communication exists between ants. The latter is achieved through pheromone deposition, with a combination of both also being possible. Pheromone use is further categorised into ants which use just one pheromone type, and ants which use multiple pheromone types.

The Ant Entity, Abilities and Properties: In its simplest form, the ant is a primitive entity with a variety of limitations:

- It can access only local information and perform only local evaluations;
- It is unaware of the global solution, being aware only of its own processing;
- It does not keep track of previous solutions;
- It possess restricted sensory abilities. That is the ant is not allowed to view internal states of other ants which are not in its immediate proximity;
- It is only capable of constrained interaction with other ants and its surrounding environment.

One of the aims of this thesis is to conform with the cardinal rule of biological swarm intelligence, which forbids giving entities (in this case ants) additional abilities to circumvent the above mentioned limitations, especially centralised global control, unlike some ant algorithm variations which fail to do this.

Application Area: A further selection criteria is the area of applicability (domain) of the algorithm. Whilst clustering and classification algorithms tend to be coupled together, there is a difference in the way they operate, as outlined in section in (2.2). Clustering and the analysis of clusters formed is usually a prelude to classification and respective learning algorithms. This discriminant is used to distinguish between clustering and classification applications, and when the algorithm overlaps into both domains.

2.6 Selected Models in Detail

This section analyses in further detail the most pertinent algorithms in literature and contrasts the key operating mechanisms against the MPACA. These algorithms have been chosen as they highlight the core elements of each of the four types of algorithms listed in section (2.5.1), contrasted using the comparison criteria as per section (2.5.2).

2.6.1 Type I - Clustering using Ants' Self-Aggregation

2.6.1.1 The AntTree Algorithm

This consists of an algorithm inspired by the "chains of ants", where ants of the species *Linepithema humiles Argentina* and the *Oecophylla longinoda*, become fixed to one another to build structures in order to fill gaps between two points. In the AntTree, artificial ants build a tree depending on the similarity between nodes [Azzag et al., 2003]. Each ant represents a node of the tree being assembled or the data being clustered. Ants add themselves onto an initial node, called the support, then successively onto the ants fixed to this node and so on, until all ants are attached to the structure. This creates fixings or connections. The structure grows according to the interactions performed by ants traversing it. Ants can disconnect themselves from the structure, with ants on the leaf nodes being more prone to exit than ones in the middle of the structure.

The basic principle involves the ant, a_i, moving downward in the tree by following the path of maximum similarity. If at a given level along this path, a_i is sufficiently dissimilar to the children of a_{pos}, then a_i is connected to a_{pos} in order to create a new sub-category. When a_i meets a leaf, it is connected to this leaf. The threshold for similarity is equal to S_{mean}, taken to be the arithmetic mean of S_{min} and S_{max}, which are respectively the minimum and maximum similarities. In this case the similarity measure is denoted by $Sim(d_i, d_j)$, represented by the Euclidean distance for numeric values and the Hamming distance for symbolic values, with all results being normalised within $[0,1]$ [Norouziy et al.]. Additionally, once detached, ant a_i can

be placed back on the support node, moving along the tree until it finds a better place to connect itself to. When all ants are connected within the structure, the algorithm stops, resulting in data partitioning. The emergent property is the creation of a tree structure, similar to hierarchical clustering presented earlier.

Evaluation Criteria

The architecture used in the AntTree is that of a tree (graph). Ant movement is driven by the particular similarity measures that exist between nodes (or other ants). Therefore, ants traverse paths which are most similar to them. The interpretation of clustering information is entirely based on the structure formed, and the sub-trees linked to the primary support. This sub-division of ants towards each of these clusters is heavily dependent on the initialisation sequence. The first connections to the support influence tree formation heavily. A number of adjustments are used, from random sequencing, to applying a decreasing order of similarity when linking ants to the support. Irrespective of claims by its authors, the ordering of data elements still remains a present limitation of the AntTree algorithm. The ant itself is a primitive entity, which only applies a similarity measure on nodes. There is no concept of colony, and ants do not make use of any pheromones. Furthermore, the collective behaviour and positive feedback action of ants is severely restricted. Ants interact with each other quite physically in the immediate vicinity. Despite not having a core controlling mechanism, the algorithm is still not totally decentralised, as ants have to traverse paths commencing from a central node, going down specific paths. This top-down structure construction makes it difficult for ants to execute traversals that are independent of each other. This since ants need a structure to traverse, which is built from other ants, hence an ant only builds on the structure of other ants. The application area is within the field of hierarchical knowledge formation.

Relevance to the MPACA

Its relevance to the MPACA is due to its similarity in the phenomenon of self-assembly. The MPACA does not form any structures in the way the AntTree does. Ants in the MPACA learn features from other ant encounters, a mechanism which also leads to ants clustering around certain areas and merging into bigger colonies. In the MPACA ants get engulfed by some colonies and dropped by others. This dynamic structure occurs within the “colony” formation. Clusters generated by the AntTree algorithm are contrasted with those achieved by the MPACA in the results section.

2.6.2 Type II - Clustering using Ant Aggregations and Ants' Self-Aggregation

Following from section (2.5.1), there are three main sub-divisions within this typology; The Standard Ant Clustering Algorithm (SACA), Self-Aggregation within a 2D Grid, and Ant Aggregation through Pheromone in a 2D Grid.

2.6.2.1 Standard Ant Clustering Algorithm (SACA)

In the LF or SACA [Lumer and Faieta, 1994], an extended measure of similarity between two data objects is used over the BM. Ants are given the capability to sense objects in their surrounding region, s, however unable to communicate in any way with each other. The notion of short-term memory is introduced, where each ant is capable of remembering a finite number of locations where it has successfully dropped off an item. Hence, when picking a new item this memory is consulted in order to bias the direction the ant moves towards. It follows that the ant tends to move towards the location where it last dropped a similar item. Lumer and Faieta define picking and dropping probabilities in equations (2.28, 2.29, 2.30). The function, f, represents the similarity density dependent function of the objects in the neighbourhood, r, located by the ant. A value is assigned to α, a parameter that is applied to scale dissimilarity. In this function, s represents the size of the selected local neighbourhood around the ant's current position. For the given object, o_i, the function calculates the difference between this object and the objects in its neighbourhood, $o_j \in Neigh_{s\times s(r)}$. This is implemented as the summation of each of these distances. The Euclidean distance, d, between two objects o_i, o_j, is divided by α, and subtracted from the fixed value one, equation (2.28).

$$f(o_i) = max\left\{0, \frac{1}{s^2}\sum_{o_j \in Neigh_{s\times s(r)}}[1 - \frac{d(o_i,o_j)}{\alpha}]\right\} \tag{2.28}$$

Thus, the difference between two objects is expressed as in terms of the Euclidean distance within an n-dimensional space, limiting its ability to handle non-ordinal objects.

The pick-up and drop-off probabilities are calculated by applying the f value and contrasting them against k_1 and k_2, which are two arbitrary values, respectively in equations (2.29, 2.30).

$$P_{pick}(o_i) = \left(\frac{k_1}{k_1 + f(o_i)}\right)^2 \tag{2.29}$$

$$P_{drop}(o_i) = \begin{cases} 2f(o_i), \text{if} \quad f(o_i) < k_2 \\ 1, \qquad \text{if} \quad f(o_i) \geq k_2. \end{cases} \tag{2.30}$$

A number of features have been added to the SACA to improve performance, namely ants having different moving speeds and a short-term memory as well as behavioural switches. The latter implies that ants can start to destroy clusters if they have not performed any pick-up or drop-off actions for a given number of time steps [Bin and Zhongzhi, 2001]. Other research includes Bonabeau who explores the influence of various weighting functions, especially those with short-term activation and long-term inhibition [Bonabeau et al., 1999].

Evaluation criteria for the BM and the SACA approaches

The architecture used is that of a 2D grid, where spatial positioning of objects is irrespective of their underlying properties. Each data object represents a multi-dimensional pattern. The ant movement is either random or guided by a visibility range driven by the entropy of the surrounding neighbourhood, where ants move towards areas of higher interest. This is all powered by a similarity function, whereby the ants attempt to move items into closer clusters of similar items. The emergent property of these actions is that objects are subsequently repositioned according to their similarity, into places of higher similarity, effectively resulting in clustering. Clusters are interpreted by the final aggregation of ants, and the features which these ants represent. The colony is non-influential. A collection of ants is used, but the colony gives no extra information on the formation of clusters. The mechanism is decentralised as ants execute actions in isolation, however apply a degree of visibility which includes an ability to comprehend the surrounding environment. Ants do not communicate with other ants directly, or indirectly, but are aware of their spatial positioning. There is a limited element of stigmergy, as ants locate themselves over the grid influencing the movement and actions of other ants. The ant is a primitive entity with two states, active or dormant. The ant has limited memory skills. Depending on the variation of the SACA, the ant itself has an element of memory, visibility and other abilities. The application area of this algorithm is that of clustering.

Relevance to the MPACA

Numerous ant clustering algorithms use this SACA (or part-SACA) approach. The SACA approach is considered by many as the de-facto ant clustering algorithm. Besides the fact that the approach is ant-based, heavily decentralised, and applied to the clustering domain, the MPACA shares few similarities with the SACA. In the MPACA, data elements (objects) are not aggregated into clusters. These data elements are used to build a spatial sub-representation within a graph, where each data element is represented by a node, connected with edges between nodes. The edges make it possible for ants to traverse from one node to the next. In the MPACA ants deposit pheromone whilst traversing such edges, and are also influenced by this pheromone when choosing which edge to traverse next. The SACA approach does not make use of any graph

structure or pheromone deposition.

There exists a resemblance between the way in which ants in both models respectively calculate similarity. Unlike the SACA, in the MPACA the similarity function is not applicable to the entire object, but merely to the features the ant is interested in. Therefore, the ant does not react to all attribute values, but only to a selected subset of the possible permutations. This subset is learnt within the MPACA during ant interaction, yet another crucial difference as the SACA has no concept of learning, in that ants do not change as the execution progresses. Ants in the MPACA acquire (learn) new features, and lose (forget) other less relevant features. Thus, ants in the MPACA have more adaptability throughout various stages of execution. In the MPACA ant colonies are clusters, whilst in the SACA colonies are effectively irrelevant.

A number of algorithms are found in literature that compensate for the deficiencies of the SACA. Since the core operand remains the SACA model, these are not critically evaluated further, and only key variations/extensions are outlined.

Extensions to the SACA model

Gutowitz proposes complexity seeking ants [Gutowitz, 1995], where ants can see local complexity and perform the actions of picking-up and dropping-off items in the regions of highest complexity. In his work, it is claimed that this complexity seeking attribute increases the effectiveness of sorting and the clustering process. Monmarché *et al.* propose a hybridisation of the SACA by including the K-Means, called the AntClass algorithm [Monmarché, 1999], [Monmarché et al., 1999b]. The approach differs from the SACA since the algorithm allows an ant to drop more than one object in the same cell, forming heaps of objects. It makes use of hierarchical clustering, allowing ants to carry an entire heap of objects. This process can be described in four steps:

1. Application of the ant-based algorithm; this is used for clustering objects in order to generate a number of accurate clusters, however this will also have some obvious classification errors.
2. Application of the K-Means algorithm; this uses the initial partition provided by the ants, and filters out the classification errors at low level.
3. Application of the Ant-based clustering; this is carried out on previously found heaps.
4. Application of the K-Means algorithm; once again performed on the clustered objects.

Monmarché *et al.* provide empirical results, with the AntClass algorithm being successfully tested over several databases, including ones presented later on in this thesis [Monmarché et al.,

1999b]. The use of the K-Means algorithm greatly increases the convergence rate of this algorithm, whilst the use of a heterogeneous population of ants avoids complex parameter settings, which would otherwise require expert domain knowledge.

Handl *et al.* introduce the Adaptive Time Dependent Transporter Ants (ATTA), incorporating adaptive heterogeneous ants, a time-dependent transporting activity and a method that transforms the spatial embedding produced by the algorithm into an explicit partitioning [Handl et al., 2006]. They demonstrate that their proposed modifications yield significant improvements in terms of quality and speed of the solution generated.

The ATTA is yet another SACA inspired approach, which does not offer much further insight into the variations from the MPACA. Despite this, it offers a very compelling benchmark for measuring ant clustering algorithms. Handl *et al.* involve metrics to measure the efficiency of clustering algorithms, and possibly provide the most compelling experimental evaluation of an ant-based algorithm [Handl et al., 2003b]. The results are further evaluated against the MPACA.

2.6.2.2 Self-Aggregation within a 2D Grid

The Ant Sleeping Model (ASM) and Adaptive Artificial Ant Clustering

The ASM algorithm originates from a metaphor which is not fully an ant inspired one, but more of a Cellular Automata (CA) approach, based on the principle of ants aggregating in the quest of seeking a secure habitat. Data elements in the form of ants are deployed in a 2D grid environment. This method uses the metaphor that ants are continuously seeking a safe and comfortable environment where to sleep, upon which Chen *et al.* base the ant sleeping model (ASM) [Chen et al., 2004].

The modus operandi surrounding it is very similar to SACA, with the notable difference being that rather than ants moving objects within grid space, ants are themselves representative of objects, and hence aggregate themselves. The algorithm starts off with a number of ants randomly scattered over the 2D grid, topologically equivalent to a sphere grid. The search for safe areas is thus represented by the search for areas of higher similarity. At each iteration the ants move to a new location, and calculate a fitness, f, for the relative position. The fitness of the position occupied by a particular ant is directly linked to the similarity of the neighbourhood around it. The ant uses a probability, p_a, so as to decide whether it needs to continue moving, or if it has located a safe place and becomes dormant (sleep state). If the current probability (p_a) is small, there is a lesser chance that the ant continues moving, and therefore a higher chance the ant stays put. If the area surrounding the dormant ant changes substantially, the ant will revert to a wake

state and attempt to relocate to a safer location. The collective movement of ants (which are representative of objects) in search for safety causes agglomeration of ants (objects) to occur, thus clustering, much in the same way as in SACA.

The algorithm uses a fitness function that makes use of local information, implying that the entire ant group, dynamically and independently from each other, self-organises into aggregation areas, within which highly similar ants are closely connected. As part of their collective behaviour, clusters are eventually formed, even if agents exert only a sphere of influence on the area surrounding them.

Building on the ASM, the Adaptive Artificial Ants Clustering (A^4C) algorithm introduces self-adjustable parameters which minimise the information of the surrounding neighbourhood [Chen et al., 2004]. Self-adaptive adjusting parameters represent fewer restrictions and require less computational cost, whilst giving an overall improved clustering quality over standard SACA methods.

Evaluation Criteria

The only variation between SACA and ASM-type algorithms is the self-aggregation of ants, rather than the aggregation of objects. This since SACA moves objects by similarity, whilst ASM-type algorithms move themselves. Excluding this, the same critique applies.

Relevance to the MPACA

The relevance of this algorithm is in the way ants agglomerate to form a cluster. This is similar to the MPACA, where groups of ants which continually encounter each other agglomerate into the same group. Ants in the MPACA do not represent data elements in a fixed place, but a mechanism which reacts to features present on data elements, which are nodes in a graph.

The aforementioned algorithms focus on the collective ability of a colony of ants to move objects or proxies for objects around to create aggregation of objects within a 2D spatial arrangement, which eventually form cluster definitions. These are valid clustering algorithms in their own right, however they lack a mechanism which is crucial to the MPACA. That is the use of pheromone.

2.6.2.3 Ant Aggregation through Pheromone in a 2D Grid

The SACA based algorithms lack the benefits of cooperative ant actions, tantamount with ant colonies. The next algorithm eliminates this limitation by allowing ants to deposit pheromone traces. The pheromone traces enhance the gathering of ants within particular spatial positions

within the environment, with the behaviour produced also being a stigmergic one.

Aggregation Pheromone Clustering (APC)

The APC builds on the principle that pheromone causes clumping or clustering behaviour which brings individuals of the same species closer to each other [Ghosh et al., 2008], [Tsutsui et al., 2005]. In this model, ants are representative of data elements and are allowed to move in the search space looking for points with higher pheromone density. Ant movement is controlled by the pheromone intensity around the ants' location. The higher the accumulation of pheromone at a point, the more likely it is that ants will move towards this point. This eventually results in the formation of homogeneous groups of data. Since the number of clusters formed could exceed the number of clusters present in the problem, a second agglomerative average linkage algorithm is used.

The APC operates on a dataset of n patterns, $x_1, x_2, x_3, x_4, ..., x_n$, and the population of n ants, $a_1, a_2, a_3, a_4, .., a_n$, where an ant a_i represents the data pattern x_i. Each ant emits pheromone within its neighbourhood, which intensity of pheromone by ant a_i located at x_i decreases with its distance from x_i. The pheromone intensity at a point closer to x_i, exceeds that of points farther away from it. The pheromone intensity deposited by ant a_i at point x, is $\Delta\tau(a_i, x)$. This pheromone spread mechanism is modelled by a Gaussian distribution, a mechanism which also compensates for the similarity between x and x_i, where the smaller the Euclidean distance between the two points, the higher the pheromone concentration is, and vice-versa, as per equation (2.31):

$$\Delta\tau(a_i, x) = exp^{-\frac{d(x_i,x)^2}{2\delta^2}} \tag{2.31}$$

where δ denotes the spread of Gaussian function and $d(x_i, x)$ is the Euclidean distance between x_i and x. The total aggregation pheromone density at x, deposited by the entire population of n ants is computed using equation (2.32):

$$\Delta\tau(x) = \sum_{i=1}^{n} exp^{-\frac{d(x_i,x)^2}{2\delta^2}} \tag{2.32}$$

Ant a_i, initially located at x_i, moves to location x_i' if the total aggregation pheromone density at x_i' is greater than that at x_i. This movement is defined by equations (2.33, 2.34):

$$x_i' = x_i + \eta . \frac{Next(a_i)}{n} \tag{2.33}$$

where:

$$Next(a_i) = \sum_{j=1}^{n} (x_j - x_i) . exp^{-\frac{d(x_j,x_i)^2}{2\delta^2}} \tag{2.34}$$

with η being a proportionality constant at each step size.

The process of determining new locations is continued until the ant finds a location where the total aggregation of pheromone is higher than its neighbouring points. When this occurs the point x_i' for ant a_i is assumed to be a new potential cluster centre, Z_j, where $j = 1,2,...,C$, with C being the number of clusters, and the data point with which the ant was associated earlier, x_i, is assigned to the cluster formed with centre Z_j.

Any other data points which are within the distance $\delta/2$ from Z_j are assigned to this new cluster. If the distance between x_i' and the existing cluster centre, Z_j, is less than 2δ and the ratio of their pheromone densities is greater than the predefined threshold density, the data point x_i is allocated to the cluster having cluster centre at Z_j. Higher density values indicate that data points should belong to the same cluster.

The APC is first applied to generate a number of clusters. The next step involves determining which clusters are maintained, and which ones are too small and therefore engulfed into larger ones. In order to achieve this, the use of an agglomerative hierarchical clustering algorithm [average linkage, section (2.2.4.1)] is used. This renders the APC a two-stepped approach. Research by Ghosh *et al.* validate their results against a number of metrics, which are further compared against the MPACA algorithm in the evaluation section [Ghosh et al., 2008].

Evaluation Criteria

The architecture used is that of a 2D grid, where the initial spatial allocations are irrespective to the attributes of the object. Each data element (object) is represented by an ant, and placed within this 2D grid. The ants move with the aim of creating homogeneous groups of data. Ant movement is guided by pheromone intensity, and is governed by the intensity of aggregation pheromone deposited by all other ants at a point. This gradual pheromone build up and movement results in the formation of ant aggregations. The final agglomeration of ants, where ants are proxies for data elements, represents the clusters formed. The colony is non-influential, and only represents the collection of ants used. It returns no extra information on the formation of clusters. The mechanism is decentralised, as the ant executes actions in isolation. There is also full use of stigmergy, as pheromone traces are key elements for movement and other ant actions. The ant is primitive, as it does not have any additional functionalities. The application area is that of clustering.

Relevance to the MPACA

The importance of the APC within this literature review is that it introduces the idea of clustering by pheromone attraction and aggregation. This approach is closer to the MPACA than any of the

other methods introduced earlier. Albeit, the pheromone usage is dissimilar to the way it is used by the MPACA. The complexity of pheromone usage in the MPACA is by far superior to that in the APC. In the MPACA, there are pheromone scents which are specific to each feature or feature combination. The APC only uses a generic pheromone as an indication for ant density. There is no learning mechanism that takes place within the ants, only aggregation.

The algorithms discussed so far have introduced the use of pheromone scents and the ability to catalyse the coordination of ant movements. Pheromone scents are not solely used to transfer information about some position or locale, but they also serve to transfer other colony level information, resulting in a shared learning mechanism. An algorithm which makes use of such mechanisms is discussed next.

Unlike other pheromone driven algorithms presented later in type IV, the initial distribution of objects within space is not significant. If the learning process was to be run again, one would hope to get the same clustering of ants but not necessarily in the same areas of the 2D grid.

2.6.3 Type III - Clustering Inspired by the Chemical Recognition System of Ants

Labroche *et al.* transpose the idea of colonial closure to the clustering domain [Labroche, 2003], [Labroche et al., 2002a], [Antoine et al., 2008], [Labroche et al., 2002b]. That is, the allocation of ants into a colony and the recognition mechanism for this to occur is based on a set of comparisons of the perceived label to its template and the emerging behavioural rules from this reaction. The ANTCLUST assigns an ant for each data element. Initially, ants have no label and are described solely by their genetic odour. As ant encounters take place, these labels are adjusted depending on encounters with other ants, modelled by the selection of multiple pairs of ants at each iteration. This clustering behaviour is outlined in algorithm (1).

Algorithm 1 ANTCLUST main algorithm

(1) Initialize the ants:
(2) $Genetic_i \leftarrow i^{th}$ objects of the dataset
(3) $Label_i \leftarrow 0$
(4) $Template_i$ is initialised
(5) $M_i \leftarrow 0, M_i^+ \leftarrow 0, A_i \leftarrow 0$
(6) Simulate Nb_{ITER} iterations during which two ants, that are randomly chosen, meet
(7) Delete nests with less than $P \times n (P \ll 1)$ ants
(8) Re-assign each ant having no more nest to the nest of the most similar ant found that have a nest [Labroche et al., 2002a].

A similarity function is used during ant encounters, this allowing ants to adjust their labels. The similarity function operates on data elements, i and j, and outputs a value, $Sim(i, j)$. When $Sim(i, j) = 0$, the two elements are dissimilar, whereas when $Sim(i, j) = 1$, the two elements are

identical. For n-dimensional numerical data, x_i and x_j, equation (2.35) is used:

$$Sim(i,j) = \frac{1}{n}\sum_{k=1}^{n}\left(1 - \frac{|x_i^k - x_j^k|}{|\max x^k - \min x^k|}\right) \tag{2.35}$$

This process of adjustment operates as follows, where for each ant i:

1. The Label, $Label_i$, is determined by the nest which the ant i belongs to, coded by a number representative of the nest. Initially, all ants have $Label_i$ equals 0. This value adjusts over time, until each ant finds its best nest.
2. The Template is evenly split into components, the first being the genetic odour, $Genetic_i$, which corresponds to the object within the dataset and does not change, whilst the other second half is derived by an acceptance threshold, $Template_i$. This latter component is learnt during the initialisation phase. Therefore, the Template threshold, $Template_i$, is composed of the function of all the similarities observed during this period. This renders the acceptance threshold, $Template_i$, dynamic.
3. M_i measures the success of ant encounters. Initially, at time $t = 0$, an ant which has no encounters has $M_i = 0$. M_i is increased when ant i meets another ant with the same Label and decreased when the opposite occurs. M_i also estimates the size of the nest to which i belongs to.
4. The estimator, M_i^+, measures how well accepted an ant i is within its nest. This measure of acceptance is increased when ant i meets another ant with the same Label and when both ants accept each other. The opposite also holds, that is the measure is decreased when ants of different labels encounter each other.
5. Age is defined by A_i, which at the beginning equals 0, and is used when updating the acceptance threshold.
6. The maximum similarity formula is $Max(Sim(i,\cdot)$ and mean similarity $\overline{Sim}(i,\cdot)$ which is observed during its meeting with other ants.

The ANTCLUST consists of two operators, the initialisation of young ants and the ability of ants to resolve label and nest belonging. In the first operator, the template, $Template_i$, for each ant i, is learned during meetings. At the end of this period, the ant i possesses values of mean and maximal similarities defined in equation (2.36):

$$Template_i \leftarrow \frac{\overline{Sim}(i,\cdot) + Max(Sim(i,\cdot))}{2} \tag{2.36}$$

The second operator is the ant meeting resolution. The label and acceptance threshold are changed according to the behavioural rules in equation (2.37).

$$Acceptance(i,j) \Leftrightarrow (Sim(i,j) > Template_i) \wedge (Sim(i,j) > Template_j) \tag{2.37}$$

The behavioural rules follow:

- R_1, new nest creation:
  If $(Label_i = Label_j = 0)$ and $Acceptance(i,j)$, then create a new Label $Label_{NEW}$ and $Label_i \leftarrow Label_{NEW}$, $Label_j \leftarrow Label_{NEW}$. If Acceptance is false then rule R_6 is applied.
- R_2, adding an ant with no Label to an existing nest:
  If $(Label_i = 0 \wedge Label_j \neq 0)$ and $Acceptance(i,j)$, then $Label_i \leftarrow Label_j$. Similarly, the case $(Label_j = 0 \wedge Label_i \neq 0)$ is handled in the same way.
- R_3, positive meeting between two nest mates:
  If $(Label_i = Label_j) \wedge (Label_i \neq 0) \wedge (Label_j \neq 0)$ and $Acceptance(i,j)$ then
  Increase M_i, M_j, M_i^+, M_j^+. This is achieved by increasing equations (2.38, 2.39):

$$x \leftarrow (1-\alpha) \times x + \alpha \tag{2.38}$$

$$x \leftarrow (1-\alpha) \times x \tag{2.39}$$

- R_4, negative meetings between two nest mates:
  If $(Label_i = Label_j) \wedge (Label_i \neq 0) \wedge (Label_j \neq 0)$ and $Acceptance(i,j) = False$ then
  Increase, M_i, M_j and Decrease M_i^+, M_j^+.
  The ant $x\,(x = i, x = j)$ which has the worst integration in the nest $(x|M_x^+ = Min_{k\in[i,j]}M_k)$ loses its Label.
  Following which there is more nest $(Label_x \leftarrow 0, M_x \leftarrow 0$ and $M_x^+ \leftarrow 0)$.
- R_5, meeting between two ants of different nests:
  If $(Label_i \neq Label_j)$ and $Acceptance(i,j)$ then decrease M_i and M_j.
  The ant x with the lowest M_x, therefore belonging to the smallest nest changes its nest value to that of the encountered ant.
- R_6, default, if no other rule applies, nothing happens.

This interaction allows ants to resolve meetings and adjust the labels and templates. Indirectly, ants migrate from one colony to another, a process which builds bigger colonies, and erases smaller ones. In the end, this results in a restricted number of colony values, where nest mates are most similar to each other than the ants of other colonies, providing a partition of the set of objects. This represents the core of the clustering mechanism.

Labroche *et al.* contrast their work with the K-Means algorithm [Labroche et al., 2002a] and demonstrate that on the contrary to other clustering algorithms, the approach does not impose any assumptions on the given dataset and does not require initial partitioning or a known initial number of classes. This method is easily applicable to many fields including web mining and other unsupervised domains. There are many cases of ANTCLUST implementations [Labroche et al., 2003a], [Labroche et al., 2003b], including variants that have been shown to separate noise from data [Zaharie and Zamfirache], credit evaluation of small enterprises [Xue-chun et al., 2007], and web-mining [Inbarani and Thangavel, 2006].

Evaluation criteria

The ANTCLUST uses a pairing comparison mechanism, where each ant is matched against any other ant. This excludes the need for a spatial architecture. Clusters are interpreted by the final colony position the ants are in, which is their label. This renders the use of multiple colonies in the ANTCLUST essential. In the ANTCLUST pheromones are not used to guide movement, but to transfer knowledge between ants. This information gives a sense of belonging into a colony more than any other colony. The mechanism is distributed, since ants adjust their label at the local level depending on ant encounters. The ant itself is a primitive entity, which aim is solely to distribute this pheromone odour within a colony. The application of ANTCLUST is within the clustering domain.

Relevance to the MPACA

The ANTCLUST shares a number of similar emergent properties, mainly the colony/cluster forming nature, and the ability of the individual ant to learn the odours belonging to a colony. Like the MPACA, the ANTCLUST allows ants to form new colonies. Operationally, this is slightly different since in the MPACA all ants initially belong to a colony, and they eventually merge into bigger colonies, whilst in the ANTCLUST initially there are no colonies, and these form only as the execution progresses. Both the ANTCLUST and the MPACA allow ants to shift colonies depending on ant encounters, albeit using a different operating mechanism.

There is also the mechanism of transferring knowledge between ants. In the ANTCLUST, ants adjust the label and template according to the odour of their nest mates. In the MPACA, ants learn to acquire new features which are carried by other ants, depending on the frequency of encounters. Hence, in both instances, this knowledge transfer happens at the ant level. The difference between both is that ants in the MPACA learn an additional feature at each merge, features describing nodes, whilst in the ANTCLUST ants learn to recognise other ants as belonging to the colony or not. The MPACA delves deeper into the properties of the objects,

since objects are effectively collections of features, whilst the ANTCLUST considers objects as distinct items to be clustered. Additionally, the similarity function applied by the ANTCLUST returns a function of difference between two objects, whilst in the MPACA there are functions which determine the difference between features, mainly if a feature "A" fits into feature "B" and so on.

A crucial difference between the ANTCLUST and the MPACA is in the way pheromone is used. Whilst the ANTCLUST uses pheromone for information exchange for colony belonging purposes only, as it operates only within a virtual environment and there is no real domain over which ants move. The MPACA on the other hand, uses the pheromone also as a guidance mechanism, where pheromone scents represent either single features or combinations of features which are being sought by the ant. The MPACA also allows a much larger element of self-adjustment to occur, as ants modify themselves according to the other ants they encounter.

The aforementioned algorithms bring us closer to the final and most important set of algorithms, those that are inspired by ant foraging metaphor. This subset of algorithms is traditionally grouped under the term Ant Colony Optimisation algorithms.

2.6.4 Type IV - Clustering using Ant Colony Optimisation Algorithms

Crucial to the understanding of the MPACA is the domain within which it operates and the principles it builds upon, introduced in section (2.3.4). The MPACA has its core an ACO algorithm which operates over graph space. The use of graph space makes it dissimilar from types I-III above. The prevalence of this research is towards ACO algorithms applied to the clustering problem.

In this typology, objects are converted into nodes and are spatially arranged depending on their underlying values. The next section is subdivided into the two segments; the multi-objective and multi-colony implementations which follow from classical ACO literature, and ACO algorithms which are specifically applied to the clustering problem.

2.6.4.1 Multi-Objective Problem Solving

Clustering problems are not commonly identified as multi-objective problems. Despite this, the utilisation of multiple pheromones in seeking different feature combinations can be seen as a multi-objective task. The MPACA is influenced by the work of Montgomery [Montgomery, 2005], where he introduces a framework around higher order pheromones. This framework is applied to multi-objective constraint searches. A multi-objective optimisation problem involves several conflicting objectives and has a set of Pareto optimal solutions. The term Pareto set or

front, was coined by the economist Vilfredo Pareto [Bruni, 2002]. A Pareto efficient situation is one in which states of current resource allocation cannot improve for any member without making another member worse off. Zhou *et al.* survey the state of the art within this field including ant algorithms [Zhou et al., 2011].

Relevance to the MPACA

In the MPACA, the ant clusters which form are dependent on the routes taken by the ants. Throughout its operation, the MPACA combines the features being sought, effectively having multiple search criteria to match. The following occurrence can manifest itself: three groups of ants, the first searching for shape, the second group searching for colour, and the final group searching for a combination of both colour and shape. The pheromone laid for each of the three differs. Even if these are not conflicting in terms of a proper Pareto front, it still remains a multi-objective search. Consequently, despite the clustering problem not being traditionally considered a multi-objective search, the transformations carried out within the MPACA allude at this being so.

The ability of different ant groups to have specific pheromone values, or a collection of pheromone values, brings us closer to the MPACA, and to another important aspect of that MPACA, which is its multi-colonial and multi-pheromone mechanism.

2.6.4.2 Multi-Colony and Multi-Pheromone ACO Approaches

The combination of the two properties, the multi-colony and the multi-pheromone aspects of ACO, and their utilisation helps to highlight a crucial novelty within the MPACA. Many ACO instances are considered to be multi-colony, however this is only a term for the collection of independent ACO instances operating on the same problem and at the same time. This is dissimilar from the interlinked ants within colonies in the way they are used in the MPACA. This offers three aspects of relevance to this thesis: (i) the multi-colony mechanism and how pheromone features in such a configuration, (ii) the polymorphism of ants within a colony and their adaptability within the search, and (iii) the exchange mechanism used to transfer information between colonies.

The multi-colony ACO consists of two main approaches, either a number of smaller ACO colonies (sub-instances) are used in an attempt to solve sub-problems within a larger problem domain, or alternatively different ACO colonies are used in attempting to solve the entire problem and compare the results attained by each. Respectively represented by the Multi-Object MACO (MMACO), and the Homogeneous MACO (HMACO) [Middendorf et al., 2002],

[Guntsch and Middendorf, 2001], [Guntsch, 2004], [Guntsch and Middendorf, 2002]. The term multi-pheromone differs from its usage within the MPACA, in MMACO/HMACO each colony has its own single pheromone type. Hence, the term multi-colony and single pheromone is more appropriate.

In MMACO ants from various colonies interact with each other whilst attempting to solve the problem, thus they adapt to the new search criteria, which is why ants in the MMACO are termed as polymorphic ants. Whilst in HMACO, ant colonies are considered homogeneous as they do not adapt during the search and all ant sub-colonies are provided with the same search criteria, and results are merged via exchange of information between sub-colonies at the end of each iteration. By exchange of information this can alternatively represent intermediate results, pheromone structures or any other information which is pertinent to the overall search. The exchange strategy is the timing at which the various ant colonies interact with each other. In MMACO each colony executes in isolation and exchange only occurs at the end, whilst in HMACO exchanges occur within specific iterations. The information exchange between colonies is described by Middendorf *et al.* as being categorised in four main exchange strategies: exchange of globally best solutions, circular exchange of locally best solutions, circular exchange of migrants, and circular exchange of locally best solutions plus migrants [Middendorf et al., 2000].

Kruger *et al.* also show that in the communication between colonies, it is better to only exchange pheromone matrices for the best found solutions, and transmit these to the rest of the local pheromone matrices [Krüger and Merkle, 1998].

By their nature of implementing a series of ACO instances, multi-colony ACO are shown to be generally superior to single colony ACO. This since at its worst case, the solution of a multi-Colony ACO tends to be equivalent to that achieved by a single ACO. Numerous publications demonstrate the effectiveness of multi-colony ACO ([Chen et al., 2007a], [Li and Bai, 2010], [Zhang and Lin, 2010], [Zong et al., 2010], [Chen and Liu, 2009], [Ling and Wei, 2009], [Viet et al., 2008], [Hao et al., 2009]).

Multi-Pheromone Approaches: Few truly multi-pheromone approaches within the same colony can be found in literature. Ngenkaew *et al.* provide an exception, but even so, this approach is restricted to only two pheromone types; a trailing pheromone and a foraging pheromone [Ngenkaew et al., 2008b], [Ngenkaew et al., 2008a]. The trailing pheromone (TP) is used to lead ants to return towards a nest or lead ants towards clusters of other ants, whilst the foraging pheromone (FP) is used to help ants locate new food sources and explore unknown areas. The

MPACA has a much higher number of pheromones. In the MPACA, the pheromones do not belong to one colony or another, but they belong to the features and the feature combinations that have formed. Hence, depending on the features ants are seeking, this determines which pheromone values are used, rendering pheromone traces totally distinct from colonies. This makes colony formation only a secondary product of the pheromone traces which are deposited, and the colony itself does not control or co-ordinate in any way the pheromone traces which are laid down.

Relevance to the MPACA

The MPACA is different from standard multi-colony and multi-pheromone implementations. Standard approaches do not offer a "complex pheromone guidance" system, which is a core activity within the MPACA. They are mostly restricted to applying different variations of the ant algorithm from a variety of angles. In fact the connection between colonies is limited to the strategic exchange points. In the MPACA there is no exchange of information between various colonies, as each colony is only a collection of ants within the larger basin of ants within the system. There is no optimisation which occurs within any specific colony, the interaction occurs between all ants in the system. Notwithstanding the interest in the above mentioned approach, this still does not satisfy the multi-pheromone concept present in the MPACA. The approach is dissimilar from the MPACA, where ants from each colony interact with other ants from any other colony via pheromones deposition. ACO by its distributed nature, is an ideal algorithm for parallelisation. Multi-colony and/or multi-pheromone approaches share many similarities and are interlinked with parallel ACO approaches [Bullnheimer and Strauss, 1997], [Stützle, 1998], [Talbi et al., 1998], [Middendorf et al., 2000], and [Chu et al., 2003].

So far the literature review has evaluated ant algorithms which use aggregations of other ants to from knowledge structures, ants which cluster objects by aggregating them based on similarity, ants which use pheromone to transfer knowledge between other ants and form new colonies, and finally ACO and its applications to multi-objective, multi-colony and multi-pheromone implementations. These algorithms lead to the ACO algorithms as applied the clustering domain.

2.6.4.3 Ant Colony Optimisation (ACO) and its Application to Clustering

Shelokar apply a direct ACO implementation to clustering, which brings us closer to the main MPACA concept, that is, clustering directly via standard ACO using auto-catalytic path reinforcement [Shelokar et al., 2004b], [Shelokar et al., 2004a].

Multiple ants attempt to solve a clustering problem by determining an optimal assignment,

which is iteratively improved, after the evaluation of a fitness function. The fitness function is the sum of the squared Euclidean differences between each object and the cluster centre of the cluster it belongs to (SSE). Given N objects in $\Re^n$, objects are assigned to one of the K clusters, such that the SSE is minimized, figure (2.8). The algorithm considers R ants to build solutions. An ant starts with an empty solution, string S of length N, where each element of the string corresponds to one of the test samples. The value assigned to an element of solution string S represents the cluster number to which the test sample is assigned in S. For a given N objects, where $N = 8$, with these being labelled a, b, c, d, e, f, g, h and three possible clusters, i.e. $K = 3$, with these having labels as c_1, c_2, c_3, solution S_1 is defined as;

$$S_1 = \{(a \rightarrow c_2), (b \rightarrow c_1), (c \rightarrow c_3), (d \rightarrow c_2), (e \rightarrow c_2), (f \rightarrow c_3), (g \rightarrow c_2), (h \rightarrow c_1)\}$$
$$\equiv \{c_1 \text{ has objects}(b,h), c_2 \text{ has objects}(a,d,e,g), c_3 \text{ has objects}(c,f)\}$$

Each of these solutions is created by each ant individually. To construct a solution, the ant uses the pheromone trail information to allocate each element of string S to an appropriate cluster label. At the start of the algorithm, the pheromone matrix, τ, is initialised to some small value, τ_0. The trail value, τ_{ij} at location (i, j), represents the pheromone concentration of element i associated with cluster j. For the problem of separating N samples into K clusters, the pheromone matrix is of size $N \times K$. Therefore, every sample has K pheromone concentrations associated with it.

At every iteration, ants will develop trial solutions via the pheromone driven communication in an attempt to obtain a near-optimal partition of the given N test samples into K groups, satisfying the defined objective. Once a population of R trial solutions has been generated, a local search is performed to further improve fitness of these solutions. Following this, in line with the standard ACO approach, pheromone trails are updated within the matrix. This step is repeated a number of cycles until convergence to a near-optimal solution occurs.

Evaluation criteria

In this approach a graph architecture is used, where ant movement follows the transition probability presented in standard ACO, being fully Bio-Inspired. The fitness function determines the quality of the clusters formed, and also the amount of pheromone to be deposited on the edges between nodes. Pheromone plays a crucial aspect in the ant movement function. This is limited to just one pheromone type. Like in many other cases preceding this algorithm, the colony is non-influential, and there is no formation of new colonies, a crucial mechanism in the MPACA. The final cluster positioning is evaluated according to the highest weights existing from each object to each cluster set. The ant entity utilises functionalities which are in-line with the ACO

transition mechanism, with the ant being able to choose which edge to visit next out of the possible edges, and can see beforehand how much pheromone each edge has, together with being able to apply a heuristic function which determines the quality of such a move before it happens.

Relevance to the MPACA

The principle similarity between this approach and the MPACA is related to path following and reinforcement of pheromone. The above mentioned algorithms, rather than minimising tour distance as in the TSP, minimise the total error of a cluster centre. Ants traverse multiple paths corresponding to nodes, in an attempt to generate combinations of "object-to-clusters" that result in lower total error differences between clusters. Despite the application of a pheromone driven approach, the similarity to the MPACA does not extended any further. Unlike the MPACA, ants do not merge into different colonies, ants do not carry any features, merge any features or adapt to the circumstances found in the dataset. There is no notion of self-assembly into bigger structures, that is, "colonies", which form the basis of the clusters in the MPACA. There also exists a difference in the way clustering is achieved. In this method, the problem at hand is converted into an "optimisation" problem rather than a graph partitioning problem.

The next set of algorithms apply ant based clustering mechanisms over graph distributions, which in many cases are a form of graph partitioning.

2.6.4.4 ACO Applied to Graph Partitioning

The first algorithm in this sequence uses competing colonies of ants to colour graph nodes. This colonisation of nodes is also a form of clustering, as it subdivides the graph into regions of similarity by proximity.

Organisation Detection Using Emergent Computing: Organisation Detection Using Emergent Computing (ODUEC), is an algorithm suited at handling balanced graph partitioning problems, by subdividing a given graph G, into k disjoint partitions [Bertelle et al., 2006]. The goal is to obtain a balanced number of nodes in each partition, whilst minimising the number of edges which are cut [Andreev and Räcke, 2004]. In this algorithm, ants are allowed to travel within graph space and detect possible organisational layouts, by means of an auto-organising mechanism. Multiple ant colonies are used, with each colony represented by a distinct colour.

A coloured graph, G, consists of N nodes, with each node being characterised by a colour at time t, with time being the synchronising mechanism. Edges connecting nodes are characterised by a weight, $|e| \in \mathbb{N}^+$, which represents the level of importance between the nodes at each edge end, e, and a quantity of pheromone representative of each colour. $C(t)$ is the set of colours

representing ant colonies at time t. In this construct, ω is the sum of all weights on each edge.

In this method ants do not perform an optimisation process as in standard ACO. That is, the solution generated by the ants is not subjected to a fitness function, which determines the amount of pheromone laid down. Ants from various colonies travel along the graph and lay pheromone corresponding to the colour of their colony. The partitioning mechanism comes into play as ants are repulsed by different pheromones of other ants, and are attracted towards their own pheromone type. At each iteration, the ant chooses the node to visit next depending on two parameters, A and T. A is the value controlling the competition amongst ants; when an ant detects a node with a proportion of similar colour less than A, this is deemed as a hostile node and the ant proceeds to move towards another node. T is a resting time value, and acts as a stabilising factor, limiting the fleeing capabilities of ants. In ODUEC the actual edge selection mechanism is still based on the ACO mechanism as per equation (2.22).

The algorithm commences with two colonies, and more colonies are added as the solution progresses. For each node of the graph a colour is computed. This node colour is said to have stabilised if this is larger than a selected threshold. Once this occurs, it is possible to compute the proportion for all nodes within the graph and determine global stability. This value is used to determine if it is possible to add more colonies or not. If the addition of another colony gives a stability that cannot reach the threshold, then this is the point where no more colonies are to be added.

Over time this action causes ant colonies to colonise areas of graph space, forming the effective partitions. In other words, these ant colonies compete in order to colonise separate subsections of the graph space. Organisational information emerges from the actions of such global behaviour of decentralised ants. This results in the nodes on the graph being coloured accordingly. The colour of the node is retrievable depending on the colour which is represented with the highest pheromone proportion. This algorithm is applied to a number of datasets, amongst which is the amazon.com book seller database. The ISBN of the book is used to find all other books bought by customers at the same time, resulting in similar topics being closely matched together.

A similar auto-organising mechanism is also presented in the Colour Ant System algorithm (CAS) [Bertelle et al., 2003] and $AntCO^2$ [Cardon et al., 2006], both of which are applied to the problem of load balancing processor loads, whilst minimising communication overhead. This differs from a traditional clustering problem, however the underlying principle remains the same. The load is transferred amongst a number of processors, and in order for the communication overhead to remain low, processes in proximity of each other operate on the same underlying processor.

Each colony consists of a processing unit, which is given a colour. Initially, it is allocated nodes with this same colour, and in a similar mechanism to competing colonies, ants allocate pheromone to deter competing ants from other colonies, and attract ants from the same colony. The colour of the node is obtained from the main colour of its incident edges. From an operational perspective, this colonisation of nodes is effectively once more equivalent to the clustering problem.

Evaluation Criteria

Graph space architecture is used, with ant movement being fully Bio-Inspired, following on from ACO movement functions. This methodology makes use of different "competing pheromone scents". An evaluation function is used to determine how much pheromone is to be deposited on each edge. The mechanism by which ants flee for safety is similar to the ASM introduced earlier. Two key differences between this approach and the ASM are the use of graph space rather than 2D space, and the use of multiple pheromones to encourage stigmergy.

Multiple ant colonies are used, which are in competing mode, and these are essential to the formation of clusters. The colonisation effect happens at the individual node, which is influenced by a population of ants. This requires a contained level of interaction with other ants. However, whilst coordination between ants is asynchronous, the inclusion of new colonies is centrally controlled. Communication between ants happens only indirectly through the use of pheromone. The ant entity is a primitive one, with no access to memory or other information. The area of applicability is that of colouring graph nodes, which can be interpreted as colonised areas of graph space, or a spatial clustering process.

Relevance to the MPACA

The utilisation of various interacting pheromones originating from different colonies and the mechanism by which spatial graph colouring is achieved, brings us a step closer to the way the MPACA operates. Like the MPACA, ODUEC does not make use of a centralised mechanical evaluation or fitness function, and the solution is built from the bottom-up. This makes both algorithms much more decentralised than many other ant algorithms found in literature. Rather than being a pure clustering approach, ODUEC is more representative of a mapping mechanism. Despite this, the emergent properties of its collective actions induce a substantial level of similarity.

A number of dissimilarities exist. The pheromone values utilised by ODUEC are not fine grained or adjusted throughout its operations. This differs from the MPACA where pheromone values represent features values and conjunctions of feature values which can change over time. In

ODUEC the number of colonies is operator controlled. ODUEC uses a common pheromone value which is common all throughout the colony. It is true that the colours of the nodes change according to ant populations, but the colonies themselves do not adapt to change. Ant colonies retain a static population of ants, so no new ants come in or depart the colony, and this introduces no variation in the pheromone phenotype. The colonies are identified by one pheromone value and ants only react positively to this scent and negatively to other scents from other colonies. Despite this being a powerful approach, the action of multiple scents is subdued when compared to the MPACA. The MPACA on the other hand uses multiple pheromones, with pheromones being distinct from colonies, and only linked to the features and feature combinations that are being carried by the ant.

It is apparent that the mechanism by which spatial colouring of nodes is achieved is primarily driven by proximity within the graph. Hence, the physical layout of the graph is crucial to the formation of coloured areas. This means that only feature values which are used in the graph setup have any significance in the search. This is a further variation from the MPACA, where even nominal values (even if not used for the graph setup itself) are capable of shifting and distorting the formation of clusters depending on their values. In the MPACA, pheromones are used to distort space between nodes to increase their level of similarity. This is not the case in ODUEC.

This colonisation and partitioning of graph space by a number of colonies serves to bring us closer to the model which is most similar to the MPACA in literature. That is, the Ant Colony Optimisation with Different Favour (ACODF) [Tsai et al., 2004]. This algorithm follows on the preceding ones and uses a set of ant colonies to partition graph space into clusters of higher similarity.

Ant Colony Optimisation with Different Favour (ACODF)

ACODF shares a number of similarities with the MPACA. It utilises graph space, where ants are allowed to roam the nodes within the graph by traversing the edges. In both algorithms, ants are given the ability to form sub-colonies by using spatial exploitation. That is, ants roam a particular area more frequently than any other, and this causes drifting off of particular subsets. Clustering is expressed in ACODF as the disconnection of the sub-graphs from the main graph. As in the ODUEC, and equally in the MPACA, ACO does not use a direct fitness optimisation function.

In ACODF once a problem is converted into graph space, each node, n, is representative of the data element being clustered, where the spatial arrangement of nodes is dependent on the

attributes within these data elements. Edges, e, are used to connect nodes.

The algorithm consists of three main variations from standard ACO coupled with a consolidation mechanism. The latter uses the pheromone traces present on edges to determine which edges can be cut-off, therefore forming new sub-graphs. The consolidation mechanism also allows the merge of smaller clusters with their nearest neighbouring cluster. The three variations are outlined as follows:

1. Ants do not visit the entire set of nodes within the graph space. Instead they visit only a reduced subset which typically consists of $1/10$th of the graph size. This subset of nodes is chosen depending on proximity, hence the term "favourable" positions, as the ant selects nodes only from this subset. Given that ant movement is partly pheromone driven, this increases the chances of ants remaining localised within a particular area of space.
2. The number of nodes which an ant visits is decreased at each iteration. This enhances the positive feedback, where ants are further encouraged to not only choose edges with higher pheromone, but also edges that lead to nodes which are in the immediate proximity. When nodes are further away from each other, trail intensity will continue to decrease until its effect is relatively nullified. Finally, a number of groupings (clusters) are built.
3. The transition function, or ant movement, differs from the traditional Roulette Wheel Selection [Dawson and Stewart, 2013]. Rather than this standard technique for proportionate meta-heuristic edge selection, the ACODF algorithm uses a tournament selection, which is also claimed to be more powerful, figure (2.9).

A cycle is a unit of time required for an ant to select a node to traverse to and execute the traversal. An iteration on the other hand is a number of time-cycles needed for all ants to traverse the entire list of nodes which are to be traversed. As the number of iterations increase, the trail intensity between closer nodes will be much higher than nodes which are further apart. Over time, ants will favour visiting the closer nodes, reinforcing such paths by depositing further pheromone. Eventually, clusters are built by dividing the pheromone that was laid on the edge between the data points.

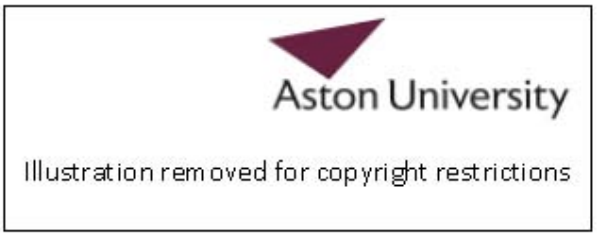

FIGURE 2.9: The tournament selection mechanism, depicting the case of randomly selecting three lines amongst five possible lines, [Tsai et al., 2004].

Two equations determine how many cycles are to be visited in the next iteration (2.40, 2.41).

$$ns(t+1) = ns(t) \times T \tag{2.40}$$

where ns is the number of visiting nodes of ants during T_0, and $ns(t+1)$ denotes the current number of visiting nodes of ants, $ns(t)$ represents the number of visiting nodes of ants at last time (cycle), T is a constant, implemented at $(T = 0.95)$. This formula causes ants to visit a decreasing number of nodes at each iteration. Equation (2.41) shows the relationship between $nf(t+1)$ and $ns(t)$, where nf is the number of visiting nodes of ants during T_1 function. $nf(t+1)$ denotes the current number of visiting nodes of the ants, whilst $nf(t)$ represents the number of visiting nodes of the ants at the last time cycle, where $run = 2$, $i \in \{1,2\}$.

$$nf(t+1) = \frac{2 \cdot ns(t)}{3} - \frac{i \cdot ns(t)}{run \cdot 3} \tag{2.41}$$

The tournament selection technique used for a proportionate selection mechanism and the selection of new nodes is based on a random selection of an edge amongst other available edges. This is followed by the selection of the shortest edge amongst the previous randomly selected edges, as in figure (2.9). The previous selected edge is prohibited, implying a TABU list of size one.

Algorithm 2 Ant Colony Optimisation with Different Favour (ACODF) algorithm

Step 1: Initialisation: n datasets are entered and m ants are assigned randomly to m nodes, where n represents the number of nodes, m denotes the number of ants, and m is equal to $n/2$.
Step 2: Compute the number of nodes next time for ants to visit.
Step 3: Compute the number of nodes for ants to visit as Step 2.
Step 4: Select n trails randomly and find the pheromone trail with high quantity, and then select this trail to visit.
Step 5: Update pheromone quantity of every trail.
Step 6: Repeat Step 2 through Step 5 until all trails of pheromone quantity are in a stable state.
Step 7: Perform clustering using the value of pheromone quantity.

Given 200 nodes, each ant needs only visit $1/10$ of this search space, hence 20 nodes. Ants only need to visit 19 nodes at the next iteration, since $(20 \times T = 0.95) = 19$. This progressively decreases the number of nodes to be visited after each iteration. Trail intensity increases for nodes which are in close proximity of each other. Ants will eventually "favour" nodes with higher pheromone and those closest to each other.

The final step in this approach is the consolidation mechanism. In this mechanism, visible and non-visible trails are set apart, where visible trails are those with a pheromone amount greater than the average of all the pheromone quantities on the trails, where non-visible edges

are removed. Clusters are defined as sub-graph areas which are still connected. This process can cause a number of small clusters to occur. To compensate for this, the nearest distance between all clusters is calculated and the smaller clusters are merged with their nearest cluster. Thus, a number of clusters finally form. Therefore, the ACODF can be seen more as a graph density partitioning algorithm. According to the results presented by Tsai *et al.* over a set of databases, the computation cost and error rate of this are much lower than in other powerful methods such as FSOM+K-Means and genetic K-Means algorithm [Tsai et al., 2004]. A more recent application of the ACODF is applied to clustering of MIT-BIH arrhythmias [Korürek and Nizam, 2008].

Evaluation Criteria

There are a number of similarities between the MPACA and ACODF. Graph space is used, where the data elements are plotted into nodes and edges are created between nodes. Ants move by using a stochastic mechanism, which takes into consideration both distance and pheromone quantity. In the work presented by Tsai *et al.* this is claimed to be influenced by elements of simulated annealing. This mechanism further reduces the number of edges at each iteration by allowing the ant to visit only a reduced subset of the graph space. The cluster that forms is dependent on two steps. Firstly, sub-graphs form after the removal of edges with non-visible pheromone, and secondly smaller sub-clusters merge based on a function of proximity. The final cluster constitutes a sub-graph in the given graph space.

All ants belong to the same colony, and all ants share the same pheromone. The algorithm is heavily decentralised and ants act independently of each other. The ant entity is a primitive one, which has limited access to graph space. Communication between ants happens in response to the pheromone deposited onto the nodes.

Relevance to the MPACA

The main similarity between ACODF and the MPACA is the approach by which clustering is achieved, using the ants' ability to form sub-colonies by using spatial exploitation and partitioning of spatially distributed nodes, via the collective behaviour of ant movement. Ant movement is in turn driven by pheromone trails. The MPACA differs from ACODF, since the former is a multi-colony and multi-pheromone approach, whilst the latter uses just one colony type and one pheromone type. The MPACA also uses more elaborate ant interactions, which allow learning to take place. Ants in the MPACA are able learn which features to react to, and move from one colony, or merge into another colony depending on ant encountering frequencies. In the

MPACA, ants recognise if the node present is of interest to them or not, making this a more powerful mechanism. Whilst as in ACODF proximity of nodes is important, ants do not only focus on the proximity of nodes, but also determine the quality of the node in relation to their search. Ant algorithms are not solely applied to clustering. The same path optimisation metaphor in ants can be used to discover rules within a dataset. This is presented in the Ant-Miner algorithm.

2.6.4.5 Rule Learning Algorithms

A final crucial family of ant algorithms is that pertaining to the Ant-Miner. The Ant-Miner algorithm uses a sequential covering approach to discover a list of classification rules from a given dataset [Parpinelli et al., 2001]. These rules are added to the list of discovered rules, and the training cases that are covered correctly by these rules are removed from the training set. The algorithm covers all, or almost all, the training cases. Ant-Miner searches for a rule list in an incremental fashion. At every iteration the ant colony discovers one rule, which is subsequently added to the end of the list of discovered rules, following which the cases covered by the rule are removed from the training set. This process is repeatedly called to discover other rules from the remaining training cases, until a stop criterion is met. Finally, the algorithm discovers a list of classification rules covering almost all the training cases. The main loop of the Ant-Miner consists of three key steps, namely; rule construction, rule pruning and pheromone updating. In these three steps, the algorithm defines a number of basic formulas, such as the heuristic function based on information theory, the transition probability based on ACO, and the quality measure of the rule and pheromone updating. The Ant-Miner comprises four key steps:

1. The first function is a problem-dependent heuristic function, η, which measures the quality of items that can be added to the current partial solution. The heuristic function remains unchanged during the algorithm execution;
2. A rule for pheromone updating, specifying how to modify the pheromone trail τ;
3. A probabilistic transition rule based on the value of the heuristic function and on the contents of the pheromone trail that is used to iteratively construct a solution; and
4. A pruning function to eliminate low quality rules.

Each rule condition is a term, so that the rule antecedent is a logical conjunction of terms in the form: IF $term1$ AND $term2$ AND ... Each term is a triple tuple $< attribute, operator, value >$, such as $< Gender = female >$.

The rule consequent (THEN part) specifies the class predicted for cases whose predictor attributes satisfy all the terms specified in the rule antecedent. The final output of the Ant-Miner algorithm are the classification rules for a given dataset or problem. Besides the benefits of using

an ant based approach, where a flexible, robust and bottom-up approach is used to learn classifications, Ant-Miner also has the advantage that data representation is intuitively comprehensible to the user. Algorithm (3) expresses the interaction of a number of key components:

1. Rule construction;
2. Heuristic Function Value;
3. Rule Pruning; and
4. Pheromone Rule Update.

Algorithm 3 Overview of Ant-Miner

```
TrainingSet = {all training cases};
DiscoveredRuleList = [];
while (TrainingSet ≥ Max_Uncovered_Cases) do
  i = 1;
  No_Ants_Converg = 1;
  Initialize all trails with the same amount of pheromone;
  repeat
    Ant_i starts with an empty rule and incrementally constructs a classification rule R_i, by
    adding one term at a time to the current rule;
    Prune rule R_i;
    Update the pheromone of all trails, by increasing pheromone in the trail followed by
    Ant_i (in proportion to the quality of R_i) and decreasing pheromone in the other trails
    (simulating pheromone evaporation);
    if (R_i is equal to R_{i-1}) then
      No_Ants_Converge = No_Ants_Converge + 1;
    else
      No_Ants_Converge = 1;
    end if
    i = i + 1;
  until (i ≥ No_of_Ants) or (No_Ants_Converge ≥ No_Rules_Converge)
  Choose the best rule R_best among all rules R_i constructed by all the ants;
  Add rule R_best to DiscoveredRuleList;
  TrainingSet = TrainingSet - {set of cases correctly covered by R_best};
end while(see [Parpinelli et al., 2001]).
```

Rule Construction: Each rule in Ant-Miner contains a condition part as the antecedent and a predicted class. The condition part is a conjunction of attribute-operator-value tuples. Given the rule condition, such that $term_{ij}A_i = V_{ij}$, where A_i is the i-th attribute and V_{ij} is the j-th value in the domain of A_i, the probability that this condition is added to the current partial rule that the ant is constructing, is as in equation (2.42).

$$P_{ij}(t) = \frac{\tau_{ij}(t).\eta_{ij}}{\sum_i^a \sum_j^{b_i} \tau_{ij}(t).\eta_{ij}, \forall_i \in I} \tag{2.42}$$

where a is the total number of attributes, b_i is the total number of values on i domain, I are the attributes i not yet used by the ant, η_{ij} is a problem dependent heuristic value for $term_{ij}$, and τ_{ij}

is the amount of pheromone currently available for time t on the connection between attribute i.

Heuristic Function Value: The heuristic value is an information theoretic measure for the quality of the term to be added to the rule, which is measured in terms of the entropy based on the preference of this term over other terms, as defined in equations (2.43, 2.44):

$$\eta_{ij} = \frac{log_2(k) - InfoT_{ij}}{\sum_i^a \sum_j^{b_i} log_2(k) - InfoT_{ij}} \tag{2.43}$$

$$InfoT_{ij} = -\sum_{w=1}^{k} \left[\frac{FreqT_{ij}^w}{|T_{ij}|}\right] \times log_2 \left[\frac{FreqT_{ij}^w}{|T_{ij}|}\right] \tag{2.44}$$

where along with the notations described earlier, k is the number of classes, $|T_{ij}|$ is the total number of cases in partition T_{ij}, $FreqT_{ij}^w$ is the number of cases in partition T_{ij} with class w, and the higher the value of $InfoT_{ij}$, the less likely that the ant will choose $term_{ij}$.

Rule pruning: The rule pruning procedure iteratively removes the term that causes the maximum increase in the quality of the rule. The quality of the rule is measured using equation (2.45). This procedure induces the discovery of more comprehensible rules, and helps to avoid the over-fitting of rules to the training dataset. This procedure is repeated until the quality of the rule can no longer improve further.

$$Q = \left(\frac{Truepos}{Truepos + Falseneg}\right) \times \left(\frac{Truepos}{Falsepos + Trueneg}\right) \tag{2.45}$$

with Truepos, Falsepos, Falseneg, Trueneg having the same definition as given in section (2.2.3.1).

Pheromone Rule Update: After each ant completes the construction of its rule, pheromone updating is carried out following equation (2.46):

$$\tau_{ij}(t+1) = \tau_{ij}(t) + \tau_{ij}(t) \times Q, \quad \forall i | j \in \text{the rule} \tag{2.46}$$

where $\tau_{ij}(t)$ is the amount of pheromone on a path at time t, and Q is the amount of pheromone to be added.

To simulate the phenomenon of pheromone evaporation in the real ant colony system, the amount of pheromone associated with each $term_{ij}$, which does not occur in the constructed rule, must be decreased. The reduction of pheromone over an unused term is performed by dividing the value of each τ_{ij} by the summation of all τ_{ij}.

Results attained by Parpinelli *et al.* show that Ant-Miner has good classification performance on test datasets [Parpinelli et al., 2002b], [Parpinelli et al., 2001]. The algorithm is able to achieve

good predicative accuracy and also contemporaneously reduce the number of rules. Numerous results are given in literature of Ant-Miner (including derivatives) and its applications, amongst which are the following; [Parpinelli et al., 2002a], [Parpinelli et al., 2002c], [Parpinelli et al., 2002b], [Parpinelli et al., 2005], [Martens et al., 2010], [Martens et al., 2008], [Cumps et al., 2009], [Vandecruys et al., 2008], [Thangavel et al., 2005], [Wu and Sun, 2012], and [Jin et al., 2006].

The Ant-Miner, despite its success as a classification method, has a number of drawbacks:

- During the rule construction phase many unfit terms are unnecessarily added and pruned at a later stage, a process that heavily increases computational complexity and cost of rule construction [Thangavel and Jaganathan, 2007].
- If the quality measure Q is very small, then the evolutionary process may stagnate.
- Ant-Miner cannot cope with continuous attributes [Otero et al., 2009].
- State transition rule computation is very complex and lacks proper balancing between exploration and exploitation [Jiang et al., 2005].
- Lacks of a continuous attribute coping mechanism [Otero et al., 2009], [Otero et al., 2008].

Evaluation Criteria

The most important consideration is that the Ant-Miner is applied to rule learning and classification and not clustering. The architecture used is that of graph space. Nodes are representative of attribute values and attribute value pairs are generated to represent rules. In Ant-Miner, ants are homogeneous primitive entities and they determine the node to move towards to next by using the standard ACO movement function, as per equation (2.42). Ants move independently of each other and depend on pheromone as a guidance mechanism. The algorithm is centrally controlled, since any pheromone update only occur after being processed by a fitness function. This is also coupled by a process of pruning of lesser quality rules, once more a centralised operation.

Relevance to the MPACA

Despite Ant-Miner being a classifier rather than a clustering tool, experimentally a number of results are presented in literature which are useful for comparative reasons. The Ant-Miner demonstrates how ant traversals within a graph is used to form rules. In the Ant-Miner, as with its sub-derivatives, a number of rules are formed, which correspond to a particular class. Besides the supervised versus unsupervised comparison of the Ant-Miner and the MPACA, there are a number of other variations which are identified in the Ant-Miner. The MPACA uses the following approaches which are not found in Ant-Miner:

1. Use of multiple-colonies;
2. Use of multiple-pheromone type;
3. Topological information is kept in the graph space; and
4. Self-assembly of ant colonies.

Ant-Miner also suffers from excessive rule generation, which rules are eventually pruned at a later stage. The possible curse of dimensionality can effectively hinder rule formation and classification results. In the MPACA, due to its distributed nature, such events are better handled.

2.7 Chapter Conclusion and Introduction to the MPACA

This chapter has introduced the clustering problem and various ways of tackling it, both via classical approaches and ant based techniques. Classical algorithms which are important, not only for their crucial benchmark value, but also because they introduce key notions in the MPACA, namely the bottom-up principle, the concept of density and graph based clustering. This chapter explored ant methods for analysing data without having prior knowledge of its inherent structure, which primarily rely on independent entities (ants) with minimal intelligence and limited coordination. Therefore, it is the collective behaviour of the ant population or colony as a whole provides the emergent properties that represent pattern recognition information in data. Various ant algorithms as highlighted in this chapter, are applied to the clustering problem, where specific models falling under each of the above types have already been compared individually for their similarity with the MPACA.

Recapping what has been introduced so far, the MPACA is an ant colony driven technique, chosen since it combines the benefits of self-organisation, stigmergy and positive feedback. It consists of distributed ants with no centralised controlling mechanism, which only interact with each other indirectly via pheromone traces. This stigmergic effect allows decentralisation and asynchronous communication between ants. Key ant algorithms their groupings into typologies and how the MPACA fits in the type IV more than the others is also explored. These typologies explored a number of algorithms which their collective difference from MPACA is highlighted further in section (3.10.3).

The next chapter explores the details of the MPACA, which has so far been only presented in part for comparison purposes. The presented literature overview provides a background setting for the MPACA algorithm introduced in chapter (3), which also provides an adequate test bed for result comparisons presented in later chapters.

Chapter 3

The MPACA Model

3.1 Chapter Overview

This chapter explains and specifies the Multiple Pheromone Ant Clustering Algorithm (MPACA). It commences with an analysis of the domain architecture, its initialisation, and pre-processing activities such as data normalisation. It then explains how ants are initialised on nodes, and how features within the domain are assigned to ants so that they can recognise nodes of interest to them. Also discussed is the pheromone driven ant movement mechanism, which determines those edges that are selected for traversal by the ants.

An important phenomenon of the MPACA is its ability to detect feature combinations. How ants contribute to this is by how feature encounters are counted, which leads to the main operators of the MPACA: the feature and colony merging mechanisms. Each step of the algorithm is detailed. The chapter then investigates the internal interactions of the MPACA and its key parameters as applied to a standard synthetic dataset. It concludes by comparing the MPACA with other algorithms reviewed in chapter (2), highlighting their similarities and differences.

3.2 Introduction to the MPACA

The MPACA is a multi-pheromone and multi-colony ant algorithm. It operates by converting data elements pertaining to a dataset into nodes connected by edges. Initially, every node has one or more ants assigned to each feature present on it. The ants attempt to locate nodes with matching values, by depositing pheromone traces linking these nodes. The process of adding pheromone to edges increases their likelihood of being selected for traversal by other ants, causing a positive feedback to occur. Each feature has a specific pheromone representation, thus introducing multiple pheromones, which is a key innovation of this algorithm.

The interaction of ants with pheromones and together with other ants, introduces to the two fundamental operators; feature and colony merging. Feature merging occurs when an ant which repeatedly detects co-occurring features, merges these features based on the conjecture that such features are related. After which point the ant will seek the conjunction of these features. Colony merging occurs when an ant encounters ants from other colonies an exceeding number of times, which indicates that the ant should migrate into such colony. It is these colonies which represent cluster definitions. This process is outlined in algorithm (4).

Throughout this text, the term colony and cluster are considered synonymous. Those properties of the MPACA that vary from generic ant colony algorithms (ACO) will be highlighted to show how they make the MPACA more applicable to the clustering domain.

3.2.1 Relationship to the Generic Ant Colony Algorithm

There are cardinal differences between the MPACA and ACO. In many cases an optimisation process takes place in ACO, where ants perform a visit to all nodes in the given graph space and then lay down pheromone depending on solution quality, thereby making use of a fitness function to determine quality. This mechanism is globally applied, therefore requiring centralisation. ACO is additionally coupled with a meta-heuristic function which uses domain knowledge during ant movement, as explained in section (2.3.4). The MPACA differs, as there is no overall fitness mechanism; ants do not need to visit the entire graph space, or perform any tours. More importantly ants move from one node to the next solely by detecting pheromone trails which are of interest to them, without any global input.

3.3 The Main Model Architecture and Processes

The MPACA uses graph space architecture. It converts the given raw dataset (of objects) into nodes, where ordinal features (continuous or discrete) for each data element dimension are used as Cartesian points to plot these nodes on the graph, reflecting a spatial arrangement. Nominal (non-ordinal) features are ignored for the purposes of graph space creation, since these have no implied order. However, nominal features that influence cluster formation (feature selection is outside the remit of this thesis) are still present at the node and influence cluster formation by being part of the pheromone trails. When an ant matches its own feature value, irrespective of whether it is nominal or ordinal, it lays a pheromone trail. The impact of the trails is to strengthen connections between nodes with matching feature values rather than moving the nodes closer together, which some ACO algorithms do. Edge distances are calculated and expressed as the Euclidean distance rather than, say, the city-block distance that is sometimes used.

Algorithm 4 The MPACA Outline

1. A given dataset is normalised and converted into graph space.
2. A number of ants are created for each feature of each node, a number controlled by the ant complement parameter.
3. Ants are initially imprinted a feature value for the feature of the node they are created in, called the instinct or base feature, and this feature stays within the ant all throughout execution. From this point onwards ants react to any other nodes which match this feature.
4. The ant deposits pheromone traces for the feature or the feature combinations, which it is carrying:
   (a) This continues until it keeps reaching nodes which match the feature combinations it is carrying;
   (b) Once this condition no longer holds, pheromone deposition ceases;
   (c) This process is only restarted once an ant finds a node which matches the feature combinations it is seeking.
5. Ants traverse the graph by selecting edges to visit depending on a stochastic mechanism influenced by the pheromone quantity representative for each feature value. That is, ants detect pheromone traces deposited by other ants on edges. If these pheromones represent feature values which are of interest to them, this process increases the likelihood of ants choosing these edges over others.
6. Ants further reinforce interesting paths by depositing pheromone, causing the positive feedback to occur. Pheromones evaporate and this evaporation serves to eliminate paths which are not adequately reinforced.
7. Two crucial operators are feature merging and colony merging, both having separate thresholds, but both of which gather information at the same points when applying the ant feature encounter counting mechanism.
8. The ant feature encounter counting mechanism used consists of a number of events:
   (a) The ant records those features being carried by other ants;
   (b) The ant records the Id of the encountered ant;
   (c) The ant records the ant deposit state (active/de-active) the detected ants are in;
   (d) The ant records the colony Ids the detected ants are in;
   (e) All of the above include a time-stamp when this encounter took place.

   Only ants that are within the same vicinity are recorded, defined by the visibility parameter, or a number of steps away.
9. Feature merging occurs when a repeated number of features are encountered by the same ant within the same time-window. There is an inherent inbuilt forgetting mechanism for merged features. These features are dropped when ants no longer detect the same features frequently enough within the same time-window. Only one exception applies, that is the feature that the ant is initially imprinted with is never dropped.
10. Colony merging occurs by ants determining which colony they should merge into. This depends on the colony Id which has been detected most frequently amongst the encounters in deposit mode, which encounters exceed the colony merging threshold. If more than one colony exists, the ant is set to belong into the colony which has the highest ant population amongst the detected ants.
11. Once ants reach a stable dynamic equilibrium in the colonies they form, the algorithm terminates.
12. When this occurs the initial data points are compared against the centroids of the features which are carried by the ants present in each colony (i.e. cluster), and this value is used to determine which cluster is best representative for such a data point.

Before the MPACA sets its ants loose on the graph space to find clusters, the domain has to be initialised. The following considerations governed the mechanisms used:

1. the environment must be scalable;
2. nodes must be linked by pathways along with ants can move;
3. pheromones must be laid by ants, which must also be able to detect and follow particular ones.

3.3.1 Partially-Connected Graph Space

In an ideal world with no computational constraints, the multi-dimensional space (MDS) could be represented by hypercubes for each point and ants would be able to reach every possible point in the space. In reality this is not feasible. The sheer number of possible movement positions would make convergence slow, if not impossible. To overcome this limitation, graph space is used. This restricts ant movement to only those edges connecting nodes, therefore rendering the algorithm more scalable.

Another scalability issue concerns the number of connecting edges. In a fully connected bi-directional graph, the number of edges is dependent on the number of nodes present. The MPACA adds a node in hyper-space for each existing data element and a large data sample would result in a correspondingly large graph and thus scalability problems. Excluding the node it is on, and the node it just arrived from, the ant is allowed to travel towards any other node in space. Therefore, the number of possible edges from any single node is $n-2$, where n represents the number of elements present. This will be a very large number for large datasets and, again limits convergence because ants can be scattered too widely. This is apart from the time taken to process so many paths and possibilities. Hence, the MPACA does not utilise a fully-connected graph. Instead, its truncation is part of the pre-processing phase for setting up the graph and edge distances.

3.3.2 Connecting Nodes and Measuring Distances

Although attributes used for the graph initialisation process are limited to ordinal ones, their variety of distributions creates its own challenges. The values for each dimension need to be standardised, in such a way that the magnitude of different dimensions are scaled to a common representation. Additionally, when non-uniformly distributes ranges are considered, like ages ranging from 10 to 15 and from 45 to 55, these values are either re-enumerated as ordinal values, or considered as nominal features. This choice is at the discretion of the MPACA operator.

This normalisation process eliminates the unit of measurement by transforming the data to scores of standard deviations (SD) from the mean. It is applied to each and every ordinal data element dimension so that the unit of each dimension becomes comparable to any other dimension. It is the usual normalisation process of converting values to SDs from the mean, known as z values, as per equation (3.1), where x is the original value and μ is the mean:

$$z = \frac{(x-\mu)}{SD}, \tag{3.1}$$

The distance between nodes is denoted in terms of the Euclidean distance, E, between each pair of nodes (a,b) with the dimensions now measured on a comparable scale of z values. However, each dimension is further converted into a number of steps, where each step equates to the distance moved by an ant in each time cycle. A **step size parameter** is calculated on the basis that four SDs above and below the mean adequately encompasses all the population apart from outliers. It is generally accepted that 95% of data points are covered within four SDs [Al-Saleh and Yousif, 2009].

The initial assumption is that dividing one SD unit into 10 steps gives an adequate granularity, so each step is 0.1 of a SD. This means 40 steps will cover the majority of the population as discussed, although there is no actual limit to the number of steps for a dimension: the outliers are still included in the graph.

Having converted dimensions into steps, the Euclidean distance, E, between two nodes is the square root of the sum of squared number of steps along each dimension. However, as the number of dimensions in the hyper-dimensional space increases, the corresponding length of the edges will also increase: this effect is known as the curse of dimensionality [section (2.2.2.4)]. To compensate for it, an adjusted Euclidean step size, U, is used, which is the Euclidean distance for one step along each dimension. Two points are separated by one Euclidean step, U, if they are one step apart along each dimension. This makes U the square root of the number of dimensions, D where $U = \sqrt{D}$. The number of steps along the edge, ES, is calculated using the normal Euclidean equation and then divided by U, as per equation (3.2):

$$ES = \frac{E}{U} = \frac{E}{\sqrt{D}} \tag{3.2}$$

where E is the length of the edge.

A second problem with the domain initialisation comes with too many data points. If every one is connected to every other one, then there will be too many edges. The **maximum edge length parameter** is used to reduce the number by specifying the maximum length of an edge:

any nodes further apart than this are not connected by an edge, as explained per section (3.3.1). Experimentation will be needed to determine the recommended settings for accurate cluster definitions.

3.3.3 Multiple-steps within Edges

In the MPACA an edge is divided into a number of steps that an ant takes to traverse it. This concept is a key variation between the MPACA and other ant colony algorithms, where edges are traversed in single atomic units. This has been identified as a weak point in other algorithms because, irrespective of how distant nodes are from each other, it takes an ant just one step to get from one node to another. This creates a topological structure for the graph rather than one that conserves distances.

In the MPACA, longer edges have more steps and traversals take more time and pheromone has longer to evaporate before the path is completed. This reinforces the inverse-relationship of length to pheromone quantity. The position of the ant along an edge is determined by the system time, where every time increment represents one step along the edge until its full length has been traversed. Thus, this process does not delay the clustering process, on the contrary it adds further emphasis on the distance between nodes.

For the algorithm to operate, a number of ants need to be initialised at each node of the graph, where the number of ants is proportional to the problem domain size.

3.4 Placing Ants on Nodes

Similarity of nodes is determined by their attribute values and ants need to communicate this information to generate similarity connections. The MPACA commences by allocating at least one ant for each feature of each node, with the number of ants per feature controlled by the **ant complement parameter**. The value of this parameter determines the total number of ants in the system and thus how well the population can cover the entire search space with enough ants to form clusters. The minimum is one ant per feature within each node. Any increase on this is applied across all nodes and features. The actual number used will be domain dependent, but the population of ants will obviously increase with the number of features per data element and the number of elements.

3.4.1 Assigning Features to Ants

During ant initialisation, each ant is assigned a particular feature value, referred to as the **base or instinct feature**. This feature is never lost by the ant throughout the execution run. Ants

are placed on every feature, both dimensional (that are used to set up the problem space) and nominal (e.g. colour). The ant takes on the feature value and responds to other nodes that match it. For nominal features, node values match on a one-to-one basis (e.g. blue with blue). For ordinal dimensions, a range of values around the ant's own value will match, determined by a **detection range parameter**.

Section (3.3.2) explained how edges are converted into numbers of steps. When an ant is assigned a value of a dimension, the **detection range parameter** will determine how many steps above or below this value will be considered a match. This detection range parameter applies uniformly to all dimension values and is necessary because continuous values will rarely match exactly. If the threshold is too wide, too many features will match, whereas if too narrow, very few features will match: the actual parameter setting needs to be empirically determined.

3.4.2 Feature Matching a Node

In the MPACA, each ant starts with the feature it is initialised to, but feature combinations [section (3.5.4.1)] mean that ants can actually end up with a combination of features to match. Either way, the matching process is the same for each one and all must match if the ant matches the node. The match applies the detection range parameter to the node's value and if it is within the range, it is considered a match.

3.5 Ant Movement

Ants start-off by being placed onto graph nodes. Subsequently, ants depart their home node and choose an edge to traverse, depending on the pheromone levels on each edge. Initially this is a random process since edges will not yet have any pheromone. When leaving a node that matches the ant's one or more features, the ant will deposit pheromone equating to its features. Hence, pheromone deposition is dependent on the features which are present on the node, and if these features are of interest to the ant. The next section outlines the pheromone definition, deposition state, quantity of pheromone deposited and its evaporation.

3.5.1 Pheromone

Pheromones are central to the whole movement process within the MPACA. In this mechanism each feature present, irrespective whether ordinal or nominal, has a representative pheromone value. For example a nominal feature like colour could have pheromone indicating "red", which differs from pheromone indicating "blue". A similar mechanism applies to normalised ordinal value features: separate pheromones are associated with each feature value on each edge and

ants will match any pheromone for that dimension that occurs within its own range. When ants acquire (learn) new features, they no longer deposit pheromone for a single feature value, but deposit pheromones for combined features together.

The point at which pheromone deposition takes place is a core consideration in ACO literature. Does pheromone deposition take place once all ants traverse the entire graph, or does the pheromone deposition take place at each edge traversal, or is it a combination of both? [see section (2.3.4)]. In order to keep in check the decentralisation aspect of the algorithm, the MPACA only applies pheromone deposition at the local ant movement level.

3.5.1.1 Ant Deposit State

Ants only lay a pheromone trail when departing nodes which have feature values that match ones being sought by the ant. The ant ceases to deposit pheromone once it arrives at a node where the features present are uninteresting (i.e. do not match). Pheromone deposition is only restarted once the ants reach a new interesting node, outlined in algorithm (5).

Algorithm 5 Pheromone deposition

```
Comment: Each time increment represents a single step moved by an ant
while (MPACA is in running mode) do
  if (Ant at a node) then
    Ant deposit mode is set to non-deposit
    if (Feature set on the node matches ant's carried features list) then
      Set ant to deposit mode
    end if
    Ant chooses an edge in accordance with the edge-selection process
    Ant executes the first step on the edge
  end if
  if (Ant is on an edge) then
    Ant moves one step forward onto the edge it is traversing
    if (Ant is in deposit mode) then
      Ant deposits a unit of pheromone signalling the features it is carrying
    end if
  end if
end while
```

The MPACA can potentially add a pheromone representation on each connecting edge for each feature value in the system. Thus, the dimensionality of the problem and the dataset size determine the number of pheromone types that can be laid down.

3.5.1.2 Pheromone Quantity Deposited

The MAX-MIN Ant System (*MMAS*) is one of the more effective ACO implementations, section (2.3.4), and gains much of its power from the fact that it limits the amount of pheromone

that can be deposited. The upper bound limits the excessive build-up of pheromone on any one edge. It is important to note that if unconstrained, the auto-catalytic effect of pheromone can precipitate stagnation at local maxima. Following from the *MMAS*, the MPACA uses a similar mechanism that increases exploration and limits the likelihood of stagnation via an inbuilt maximum pheromone value.

Each time pheromone is laid down, the amount deposited is compared against a maximum co-efficient of the standard pheromone deposition value. The upper ceiling is controlled by the **maximum coefficient parameter**, τ_{max}. τ_{max} serves as a co-efficient of τ_Q. That is, when $\tau(t) \geq (\tau_Q \times \tau_{max})$, than the amount of pheromone deposited is capped and set to $\tau_Q \times \tau_{max}$.

There is no fitness function in the MPACA which determines the quantity of pheromone that is to be added. Instead, ants deposit pheromone equally on all edges they are traversing for each feature value they are carrying. This makes the MPACA more decentralised than other approaches. The amount of pheromone at a step is expressed by equation (3.3):

$$\tau(t) = \tau(t-1) + (\tau_Q) \text{ where } \tau(t) \leq \tau_Q \times \tau_{max} \tag{3.3}$$

where $\tau(t)$ is the amount of pheromone to be deposited at time t, $\tau(t-1)$ is the current amount of pheromone present at this same step $t-1$, and τ_Q is the amount pheromone that can be deposited.

Coupled with the maximum amount of pheromone, there is also an additional **minimum tolerance parameter**, τ_{min}, which serves as a pheromone cleansing mechanism. If the amount of pheromone is lower than τ_{min}, then this value is removed from the system. Thus, there are three key considerations for pheromone deposition:

- The "pheromone quantity being deposited", τ_Q;
- The "maximum ceiling value" , τ_{max}, or the maximum coefficient parameter; and
- The "minimum clearing value", τ_{min} or the minimum tolerance parameter.

3.5.1.3 Pheromone Evaporation

The available literature presents various ways of how pheromone evaporation occurs. In most cases deposition and evaporation are combined into one process. This cannot be done in the MPACA, since pheromone deposition does not occur simultaneously on all edges. In the MPACA, evaporation is applied to each and every pheromone scent present at every given interval in

$time \rightarrow t$, via a percentage reduction expressed by equation (3.4):

$$\tau(t) = [\tau(t-1) \times (1-\rho)] \tag{3.4}$$

where $\tau(t)$ is the amount of pheromone present at a point in time t, $\tau(t-1)$ is the current amount of pheromone present at the same point previously at time $t-1$, and ρ is the pheromone evaporation rate.

Since a percentage reduction applied to any value can never reach zero, the algorithm filters out values below τ_{min}. This is a variation from the *MMAS*, which applies a lower bound on each edge, τ_{min}, where each edge always has a minimum pheromone value.

The quantity of pheromone deposited is interlinked with the size of the graph. The **pheromone evaporation parameter** controls the amount of pheromone that can exist on any edge. It follows that there must be a balance between the amount of pheromone deposited and the pheromone evaporation rate itself. If the pheromone quantity is Q, and evaporation rate is E, then E should consist of a value per unit time substantially less than Q, otherwise the Q value would be nullified. Ideally, these values are chosen by the operator of the algorithm or extracted empirically via experimentation. Irrespective of the approach, the MPACA cannot at this point self-adjust these values for optimal performance.

If the graph space is overly-compressed, higher pheromone evaporation is required in order to ensure that stagnation does not occur, as smaller evaporation intervals would not manage to decrement enough pheromone and the solution could at some point have equal maximum amounts of pheromone on each edge.

3.5.2 Edge Selection Mechanism

Ants at a node have a choice of edges to select from. Their choice is driven by the levels of pheromone on each edge that matches one or more features of interest to the ant. A probabilistic selection process is used that means an ant is most likely to choose the edge with the most pheromone matching its features. This differs from that used in other ant algorithms, as described in equations (2.22, 2.25). The MPACA movement mechanism does not require any foresight about the potential nodes that can be visited, further enhancing decentralisation.

It utilises a limited TABU list of size one [section (1.3.4)], unlike most other ACO algorithms. This minimal TABU list is only to ensure the ant cannot go back to the node from which it just departed. However, ants are allowed to visit the same nodes multiple times, with triangulation being possible, which is why a probabilistic edge-selection mechanism is used.

Every edge is given the same residual pheromone, r, that matches the ant's features so that all edges are open to the possibility of traversal. This is to prevent stagnation and local maxima, where a better global solution is masked by the local situation. If a node has n potential edges, where each edge, i, has a matching pheromone scent, s_i, then the probability of selecting a particular edge, e, is governed by the amount s_i on the edge plus the residual pheromone amount, r compared to the amounts on all other edges, as given in equation (3.5):

$$P(e) = \frac{s_e + r}{\sum_{i=1}^{E} s_i + (r \times E)} \tag{3.5}$$

where $P(e)$ is the probability of selecting edge e, E is the total number of edges.

The **residual parameter** is therefore defined as a percentage of the total pheromone on the edges leading out of the node. A zero value neutralises this parameter and a high r value creates random choices. Thus it is possible to check how it is helping prevent local maxima or having no effect.

Implementation of the probabilistic selection is by assigning each edge an accumulating numeric range, which is calculated using the probability calculation in equation (3.5). If one edge has half the total pheromone on the edges and the other two each have a quarter, then the edges are assigned a number range of 1 to 50, 51 to 75, and 76 to 100 respectively. If the random number outputs 81, then the third edge is chosen by the ant.

3.5.3 Ant Encounters

A fundamental concept underlying the MPACA is the ants' ability to acquire features from other ants, and append them in their carried feature list, effectively learning new feature combinations. For example, if ants searching for feature "blue", keep encountering other ants carrying feature "large", where such feature combinations have been encountered a number of times which exceeds the **feature merging threshold parameter**, then these two distinct features are coupled.

Colony merging represents another crucial operator. If ants co-occur within the same region of space, then this makes it ever more likely that such ants should belong to the same colony. This occurs since ants keep a record of other ants they have encountered and their respective colony Id. When the number of encounters with a specific colony Id exceeds the **colony merge threshold parameter**, the ant will set its colony identifier to this other colony Id. Therefore, an ant colony is a collection of distinct ants, which are collectively labelled with the same colony Id. Both feature and colony merging are executed asynchronously by the ants and independently of each other.

3.5.3.1 Identifying Ant Feature Encounters

Encounters with other ants are recorded for both feature and colony merging purposes. To help clarity of the explanation, the ant whose feature list will be updated is termed the focus ant. Its encounters with other ants are only monitored and recorded when it arrives at a node and only with ants within its visibility. This is controlled by the **visibility parameter**, which is the number of steps along an edge an ant can see.

Ants remain true to their nature of being distributed and decentralised, as there is no communication between ants apart from when they analyse each other because they are within the visibility range. The focus ant is able to see what features other ants are carrying and it maintains a queue for recording them during encounters. This data addition is implemented at the ant level of the focus ant, hence no other ant is influenced by this change.

If the focus ant is in deposit mode, ants at the same node or departing the same node it is in which are also in deposit mode, have their features added to the encountered features queue. This is due to the fact that ants departing such a node and are still in deposit mode, are so as they found this node interesting to them. Likewise, ants which are travelling towards a node, irrespective of the deposition mode they are in, have their features added to the encountered features queue. This since, if ants are travelling towards a node, they must be heading towards this node in believing that the node is also of interest to them. If the focus ant is not in deposit mode, then only ants coming towards its current node which are in deposit mode have their features added to the encountered features queue. This is because they are showing interest in the node they have left and this is the node chosen by the focus ant due to the pheromones on the edge.

This mechanism is operationalised by algorithm (6). The result is that ants can detect nodes with combinations of features and that can merge with other ants into the same colony.

3.5.3.2 Data Structures for Recording Encounters

The MPACA uses collections of carried and detected features, that is the carried feature list, and the recorded ant feature encounters. The latter also includes a reference to the colony Id the seen ant was detected in. The carried feature list structure has the following properties:

- *Feature_Id*, the feature identifier, common to all ants and created during the set-up and normalisation process;
- *Value*, the value within the feature itself, e.g. colour=blue or height=1.56;
- *Feature_Dimension*, the dimension value for the current feature; and

Algorithm 6 Updating Feature and Colony Encounters

```
if (Ant is in deposit mode) then
    Let α = Ants that are in deposit mode at or travelling away from the node
    for (ant ∈ α) do
        Feature-encounters ← record encounter
    end for
    Let β = Ants coming towards the node whether or not they are in deposit mode
    for (ant ∈ β) do
        Feature-encounters ← record encounter
    end for
else
    Let γ = ants coming towards the focus ant which are in deposit mode
    for (ant ∈ γ) do
        Feature-encounters ← record encounter
    end for
end if
```

- *Time*, the time instance when this feature has been acquired;

The mechanism which stores feature encounters is a first-in first-out queue, this allows the oldest encounters to be removed first. It consists of the following:

- *Feature Id*, which corresponds to a carried feature Id extracted from the encountered ant;
- *Ant Id*, representing the ant Id of the encountered ant;
- *Colony Id*, a value pertaining to the ant it has been derived from;
- *Deposit Mode*, representing the deposition state the encountered ant was in at that specific time; and
- *Time*, a time-stamp value, that is the time instance when this feature is detected, taken from system time.

The operation of the MPACA requires that each ant is initialised as an entity with the following properties:

- *Id*, the unique identification of the ant within the entire system;
- *Node At*, the current node the ant is on. When traversing an edge this is set to null;
- *Traversing Edge*, the current edge the ant is traversing. When none this is set to null;
- *Tabu_List*, a single element list that stores the node the ant has just visited;
- *Deposit Mode*, identifies the pheromone deposit state the ant is in (active/dormant);
- *Colony Id*, the colony the ant is currently a member of;
- *Carried Features List*, a collection of features including the base/instinct feature and other acquired features; and
- *Encountered Features Queue*, a recorded instance of all the features the ant detects, being carried by other ants which are within its visibility radius.

3.5.4 Merging Features and Colonies

Feature and colony merging are triggered by the number of ant feature encounters and colony Id recordings. Feature merging occurs if an ant keeps repeatedly seeing the same features carried by other ants. These features will be acquired by the ant, and from then on the ant will search for the combination of such features, which is called the carried features list. On the other hand colony formation occurs when the colony count of encountered ants goes over the colony threshold. The ant joins the colony with the highest ant population from the colony Ids which have been detected more than the threshold number of times.

These merging activities are linked, but they are not synonymous with each other, since ants do not need to share the same features to belong to the same colony. A cluster is expressed in terms of an ant colony and the features being searched for by each ant member of that colony.

Both feature and colony merging operators are limited by a **time-window parameter**. Therefore, only encounters which occur within the last number of steps represented by this time-window are taken into consideration. The MPACA is synchronised via an incremental system time process. At each time increment, each and every ant moves one step. If the ant reaches a node, the ant will update its ant feature encounters queue. When the time-stamp of the ant feature encounter exceeds the selected time-window, this is removed from the queue. The time-window keeps the size of the ant feature structure in check, and also ensures that only the most recent encounters are validated, as more time distant encounters are ignored.

3.5.4.1 Merging Features: A Learning and Forgetting Mechanism

The MPACA combines the features being sought in an asynchronous manner. It is important to note that this merging is applicable only at each distinct feature dimension. This limit inhibits the focus ant from having a combinatorial explosion of carried features. A feature vector has at most only a singular representation for each of the features such as weight, height, colour and shape.

When the ant merges features, the number of nodes matching the ant's feature detectors decreases, concurrently reducing the pheromone laying chances of the ant. This consequently leads to a natural check on the combination process, since specialising the detection reduces the number of matching nodes and the probability of an ant being in deposit mode. This results in ants becoming more localised to a particular area of graph space. When this happens, the chances of ants encountering the same subset of ants increases. This subsequently increases the likelihood of ants belonging to the same colony.

Both merging processes are implemented in algorithm (7). A list of features is produced from counting the frequencies for each encountered feature value within the given time-window, and where counts exceed the **feature merging threshold**. For each feature in this list, as long as the feature dimension is not already being carried by the ant, its carried feature list is augmented with this new feature. Thus, a feature merge occurs.

When ants become more localised, this reinforces both feature and colony merging. This is the correct functionality behind the algorithm. However, there might be instances, especially in the opening phases of the algorithm, where feature merging should not occur so rapidly. This is rectified by a forgetting mechanism, which is implemented as part of the time-window concept. The time-window is also used as a forgetting mechanism, this clears carried features within an ant which are no longer relevant. It helps prevent matching static features, enabling the ants to learn new feature combinations for different areas of space. Hence, feature merging is temporary, and this continues until an equilibrium is established, where ants stabilise the features they are carrying. A crucial aspect is that ants cannot drop the instinct feature that they were imprinted with at initialisation, since this is time independent.

The process of feature merging is distinct from colony formation for a number of reasons. Most prominently, this mechanism allows for various levels of feature merging to occur within the same colony. This means that a colony can consists of various ants which have combined over multiple but not all dimensions present. This allows various distributions of ants, having distinct feature combinations to belong to the same colony. More importantly, the separate feature merging allows the MPACA to detect interactions between variables and encode them for non-linear separability such as the XOR problem, which draws on the relationship between values of separate features occurring together.

Another aspect is that the mechanism allows for colony formation that can handle missing data. Once more, as not all ants need to merge on all dimensions, missing data for particular feature dimension values can be compensated for.

3.5.4.2 Colony Merging

As the algorithm begins execution, each ant is assigned a colony Id, representative of the initial node that it is born on which is consists of an increasing numerical sequence. During the MPACA execution, ants encounter other ants, and determine the colony Id which they carry. This colony Id is used for colony membership.

Colony merging is achieved as in algorithm (7), by using the same structure which stores the ant

Algorithm 7 Feature and Colony Merging

Let α be the frequency grouping of encountered ants by feature Ids, tuple(f,n), where f and n represent the feature Id and its respective count.
Apply a filter where only values when ($n >$ *feature merging threshold* parameter) are kept.
Let β be the frequency grouping of encountered ants **only in deposit mode** by colony Ids, tuple(c,n), where c and n represent the colony Id and its respective count.
Apply a filter where only values when ($n >$ *colony merging threshold* parameter) are kept.
for (all features, $f \in \alpha$) **do**
  if ($f \notin$ *carriedFeatures* of the ant) **then**
    Feature carried by ant $\leftarrow$ include feature f
  end if
end for
Let γ be all carried features for the focus ant
for (all carried features, *carriedFeature* $\in \gamma$) **do**
  Let $\delta =$ frequency of occurrence for the current carried feature
  if (carried feature == base/instinct feature) **then**
    Do nothing
  else
    if ($\delta <$ *feature merging threshold* within *time-window* parameter) **then**
      Drop carried feature from carried feature list for the focus ant
    end if
  end if
end for
Let ε be a repository for all ant Ids
for (all colony Ids, $c \in \beta$) **do**
  Locate ant Ids within the encountered ants that are **in deposit mode** which correspond to c
  Update $\varepsilon \leftarrow$ with value c
end for
Locate colony Id from, $c \in \beta$ which has the highest number of ants
Set ant's colony Id $\leftarrow$ to colony Id to this value

feature encounters. The only additional filter applied is that only ant encounters of ants in deposit mode are considered, as only such encounters represent ants which were satisfied to be in that particular area of graph space. The colony Ids grouped by their frequency, and frequency counts, which exceed the **colony merging threshold** are used. This will return the colony Id, amongst the colonies within which the other ants seen belong to, which has the highest population of ants. The final step adjusts the colony Id value of the focus ant. Thus, merging is done at the ant level by adjusting the colony Id.

3.6 Overall Operation of the MPACA

Central to the algorithm is the ant itself, since all respective emergent behaviours are ant induced. The ant has two main states, deposit and movement, and their subsequent combinations, which are expressed in the finite state machine depicted in figure (3.1).

Ants roam the graph as per algorithm (8). The ant moves from one node to another in ant

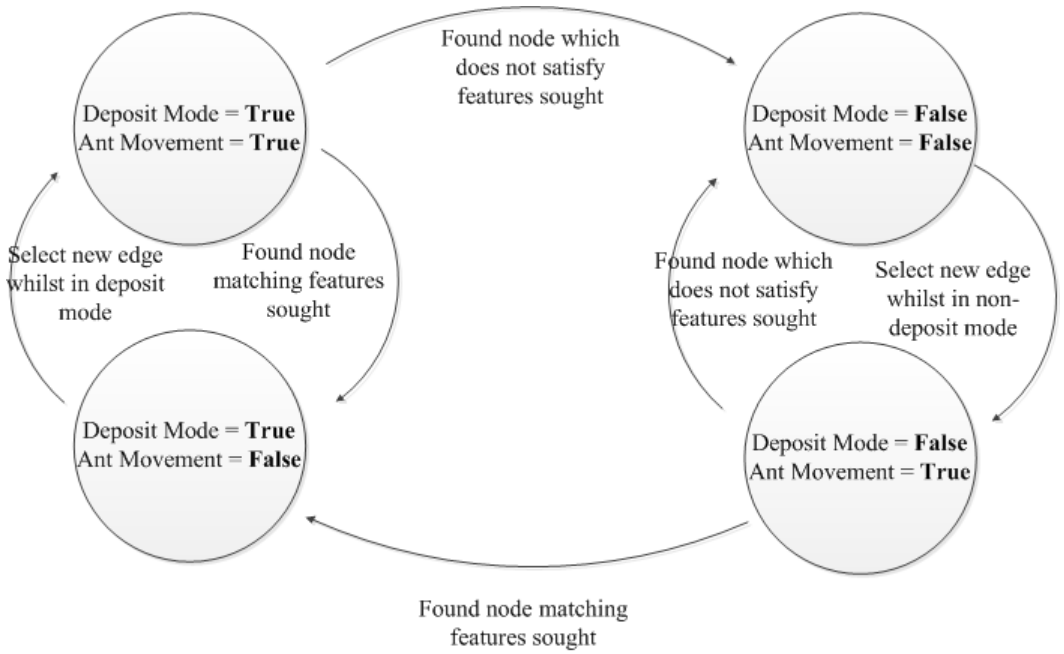

FIGURE 3.1: A finite state machine representing the changes in ant behaviour as it traverses the graph.

traversals, depositing a pheromone trail on the way. The ant selects an edge to traverse in accordance to the path-selection equation (3.5). Each time the ant arrives at a node, it determines if the node features match the carried feature list it is seeking, and if so resumes or continues pheromone deposition. On the contrary, if the node does not match the carried feature list, pheromone deposition ceases. Initially each ant carries an instinct feature as its sole feature within a carried feature list. Over time, this list increases to reflect feature information that the ant learns during other ant encounters.

The MPACA exploits the ability of ants to combine into bigger colonies, with each ant in the colony carrying a particular feature vector, which forms the basis of clusters. Therefore, the cluster is represented by a representative sample of an ant colony, with the cluster itself being a collection of the features carried by the individual ants.

The colony is a virtual structure, this since a colony is a collection of ants, which is completely distributed. Ants do not need to know which other ants belong to the same colony they are in, or any other colony. Ants do not need to know how large the colony is, for it is enough for the ants to know that amongst the colonies which they have seen this is the most popular, and its popularity exceeds the threshold to merge into such a colony. An ant easily migrates the colony it is in by adjusting the "ColonyId", with no need for a supervisor mechanism.

It is thus the ant itself that determines with which other ant/s it should merge features. It is also the ant itself that determines which colonies it should combine into.

There are a number of synchronisation considerations which need mentioning. A characteristic of the MPACA is that since ants traverse edges in steps, some ants might be traversing an edge whilst others would have already arrived at a node. Unlike other ant colony algorithms, where ant movement is synchronised, this approach enhances the asynchronous operation of the algorithm. The position of the ant within an edge is determined by the system time, where every time increment represents one move onto the edge until its full length has been traversed.

Global control is only restricted to the movement of the ant from one point to the next, synchronised by system time, and a process which also keeps the pheromone matrix constantly evaporating. The global controller is therefore a representative or an implementation of a time mechanism. In the MPACA, ants move one step within each time interval, as per algorithm (8).

Algorithm 8 The core of the MPACA algorithm

Require: Graph space with connecting edges and ants assigned to each feature.
while (*Termination not reached*) **do**
  $System\_Time \leftarrow System\_Time + 1$
  for ($Ant \in Ants\_In\_System$) **do**
    for (encountered_Feature $\in$ Ant.encountered feature queue) **do**
      if ($encountered\_Feature.TimeStamp < System\_Time -$ Time-Window) **then**
        Dequeue encountered_Feature
      end if
      if (Ant at a node) **then**
        Ant deposit mode is set to non-deposit
        if (Feature set on the node matches ant's carried features list) **then**
          Set ant to deposit mode
        end if
        Ant performs algorithm (6)
        Ant executes algorithm (7)
        Ant chooses an edge in accordance to the edge-selection equation (3.5)
      end if
      if (Ant is on an edge) **then**
        Ant moves one step forward onto the edge it is traversing
        if (Ant is in deposit mode) **then**
          Ant deposits pheromone signalling the features it is carrying as per equation (3.3)
        end if
      end if
    end for
    if (*Stopping criterion reached*) **then**
      Output cluster definitions
    else
      Perform system wide evaporation as per equation (3.4)
    end if
  end for
end while

3.7 The MPACA and Cluster Derivation

3.7.1 Mapping Colonies to Clusters

The MPACA algorithm forms colonies of ants, with each ant belonging exclusively to only one colony, and each ant having its own distinct carried feature list. Since each feature represents a dimension and a value, the collective features belonging to a colony via ants imply that each cluster is a weighted collection of these feature values.

Due to the nature of the clustering problems, it is unlikely that a perfect colony-to-cluster representation forms. In many cases smaller colonies form which would be negligible in size when compared to the larger colonies. Therefore, a mechanical tolerance is required which would filter out these small colonies that may form. In order to achieve this, the colonies are sorted by their descending population size, and only the top N sized ones are kept. The value N is determined by the known number of clusters existing in the given dataset. Hence, truncation of smaller, less densely populated colonies is used as a post-processing mechanism.

3.7.2 Termination Criteria

The MPACA uses two termination criteria, either a maximum number of iterations is reached, or ants reach a stable dynamic equilibrium in the colonies they form. The latter is defined when the populations of ants in the top N colonies (above section) stabilises.

Given a cluster, the next step is to determine cluster membership for data elements. The original data elements are matched to the cluster which best represents them in a cluster membership and evaluation mechanism.

3.7.3 Cluster Membership and Evaluation

The collection of features within each ant in each colony is used to generate centroid values for each dimension, in a similar mechanism to Centroid-based clustering algorithms presented in section (2.2.4.2). Two other secondary approaches namely, the Bayesian and a K-Nearest Neighbourhood are documented in chapter (5).

This thesis focuses on this centroid based method as there already multiple co-occurring events and parameters applicable to the MPACA, and this method albeit crude, is the simplest measure to determine overall success without being itself weighing overly on the results generated.

3.7.3.1 Centroid Cluster Membership Calculation

Algorithm 9 Proximity to Centroid calculation

```
for all data elements, de ∈ Dataset, ds do
    Let minimum distance min_d ← MaxValue
    for all cluster c ∈ Clusters C do
        Let d ← distance(centroid(c), de)
        if (d < min_d) then
            min_d ← d
        end if
    end for
end for
```

In the centroid calculation, feature combinations within each ant, and in each colony, are summed up to their centroid values. When dealing with a ordinal data, the value of each feature value is accumulated, and a centroid value is calculated for each dimension. For each feature dimension the values are weighted using the mechanism introduced by Greenacre [Greenacre, 2013], as per in section (2.2.2.3). Now that the locus of points is known, the next step is to allocate each data element as belonging to one cluster over another, depending on the proximity to the cluster centroid. In case only ordinal values are present in the data, a standard numeric Euclidean distance is applied. The data element is assigned to a cluster depending on its proximity to the centroid value. This mechanism operates as in algorithm (9).

When nominal values are also included, the centroid value is still used, however the matching process differs slightly, since an accumulated difference tally, dt, is used. In this approach, for each dimension a count for matches and mismatches takes place. When dimension matches dt is not incremented, whilst when there is a mismatch dt is incremented.

The evaluation metrics with which these mechanisms are explored is given in section (2.2.3).

3.8 The MPACA Parameters

Much of the MPACA detail resides with how it is parametrised. The determination of the effect of various intertwined parameters is certainly not a trivial task. This is hindered by the multitude of possible existing parameter combinations. A synthetic dataset is used to explore the internal effects of parameter adjustments.

3.8.1 A Synthetic Dataset for Demonstrating the Parameters' Impact

The chosen synthetic dataset for the internal evaluation is the Square1 dataset. Other synthetic datasets are evaluated in chapter (4), namely the 2D-4C and 10D-10C datasets [section (4.2.1)].

The Square1 dataset is a two-dimensional dataset consisting of four clusters arranged as a square. Data elements for each individual cluster are generated using the normal distribution, $N(\vec{\mu},\vec{\sigma})$. The number of clusters, the sizes of the individual clusters, the mean value, $\vec{\mu}$, and vector of the standard deviation, $\vec{\sigma}$, for each normal distribution are used to generate this set. That is, the normal distributions of data elements pertaining to this formulation are $(N(-5,2),N(-5,2))$,$(N(5,2),N(5,2))$, $(N(-5,2),N(5,2))$ and $(N(5,2),N(-5,2))$. The dataset is initialised to 100 data elements.

3.8.2 Baseline Analysis

This set of experiments builds on the research methodology later presented in section (4.3). In brief, a number of parameter configurations are executed by the MPACA and depending on the quality of the result attained, the parameters are narrowed further to improve results. Parameters are not adjusted during the MPACA execution, yet they are adjusted in between various execution instances. In order to establish a set of baseline values, the MPACA has been applied for 1,000 instances on varied parameter settings. The applicable parameter ranges used are given in table (3.1). Each of the experiments which now follow are the combination of results attained from these baseline experiments augmented with results from 50 execution runs for each tested parameter.

These experiments only demonstrate the internal interaction of parameters and their influence on each other. No further comparisons are undertaken as per section (2.2.3), with the latter produced in chapter (4). The units applicable to this experimental configuration also apply to other experiments presented in chapter (4) and are as follows:

1. Maximum Edge Length - the maximum number of steps on each edge;
2. Step Size - equates to a fraction of a SD;
3. Ant Complement - an integral unit representing the number of ants present per feature per node;
4. Detection Range for Ordinal Dimensions - steps above or below a mean;
5. Quantity of Pheromone (Ph.) Deposited - an integral value representing the pheromone, Q;
6. Maximum Coefficient of Ph. Deposited - an integral value representing a coefficient (for example $Q \times 2$);
7. Minimum Tolerance Value - an integral unit;
8. Evaporation Rate - a percentage applied to pheromone quantity, Q;
9. Residual Parameter - a percentage of the sum of all pheromones on selected edges;

Parameter	start	mean	SD
Max. Edge Length	5	8	1.42
Step Size	0.1	0.1	0
Ant Complement	1	5.09	4.51
Detection Range	1	1.5	0.5
Ph. Qty. Deposited	100	175	56
Max. Ph. Coefficient	1	1.49	0.5
Min. Tolerance	1	1	0
Evaporation Rate (%)	0.01	0.06	0.04
Residual Value	0	1.01	0.82
Feature Merging	3	4.51	1.5
Colony Merging	3	4.55	1.5
Visibility	3	3.5	0.5
Time-window	50	74	25

TABLE 3.1: The MPACA parameter settings as applied to the Square1 dataset. Values as varied over 1,000 instances of the algorithm, consisting of a minimum (start), mean, and standard deviation (SD). **Note:** Furthermore, isolated runs are executed on a per parameter basis (approximately 50 runs each) and are not included in the above baseline evaluation.

10. Feature Merging Threshold - an integral count;
11. Colony Merging Threshold - a separate integral count;
12. Visibility on Edge - an integral representing steps within an edge that an ant can see through; and
13. Time-window - the number of time stamped intervals which are analysed.

Due to the difficulty in measuring the interaction of all parameters concurrently, parameters are grouped together by their perceived action, as follows:

1. Domain initialisation, in which the maximum edge length and step size are analysed;
2. Ant initialisation, analysing the ant complement and detection range parameters;
3. Pheromone deposition and movement, in which the quantity, minimum and maximum amounts of pheromone present on each edge, evaporation rate, and also the residual parameter are explored; and
4. Merging thresholds, in which feature and colony merging are explored together with the edge visibility and the time-window parameters.

The evaluations to follow represent a combination of parameter specific experiments combined with the baseline evaluation, as per table (3.1). Each analysis has a varying optimality range, as optimality differs from the initial phases of execution towards the latter stages of execution. The optimal value should in theory build towards the "best" clustering solution. This analysis aims to determine the optimal value applicable to each parameter, its degree of influence, coupled with the influence this parameter has on other parameters, and why such behaviour takes place.

Graph are used to represent four key benchmarks, as described next.

Figure (a) shows the average feature combinations carried by ants. For the given 2D problem domain the optimal average feature combination value at the end of processing should be within the range of $1.5 - 1.7$. Higher values, especially at the early stage mean premature stagnation, whilst lower values towards the end mean that feature merging is not occurring. The latter lack of feature merging is an indication that not enough ant encounters are taking place.

Figure (b) shows the distribution of ants in the top N colonies versus other smaller colonies. The absorption of ants in the top N colonies is an indication of a good clustering solution. Under optimal conditions this value should range between 0.6 and 0.8, which indicates that enough ants are joining the larger colonies, whilst allowing some free ants to exist. It is unlikely that all ants combine correctly into the major colonies, and when no minor colonies are present this can indicate over aggregation. An important consideration is that sudden merging of features and colonies does not imply correct cluster formation.

Figure (c) shows the repetition in node traversals, where a high value indicates less randomness, whilst a lower value is indicative of a more varied search. In the opening phases, repetition should be low, indicating that ants are varied around the graph nodes. As time progresses more ants should be repeating the sequence of nodes that they traverse, which is indicative of correct pheromone path following. Under optimal conditions this value should range between 0.5 and 0.6, as some freedom in movement should always be present.

Figure (d) shows the progressive decline in the number of colonies present and the termination criteria. This is also time dependent, where the initial colony count is reduced as more colonies are engulfed by larger colonies. Coupled with figure (b), whilst the majority of ants are expected to be in the top N colonies, a number of smaller colonies are likely to exist, indicating correct functionality. Termination is not necessarily shown in these experiments given the short time cycles used. The decline in colony counts and the colony membership stabilisation is indicative of correct termination.

3.8.3 Domain Initialisation Analysis

Domain initialisation influences ant traversals and encounters, which are key building blocks of the MPACA.

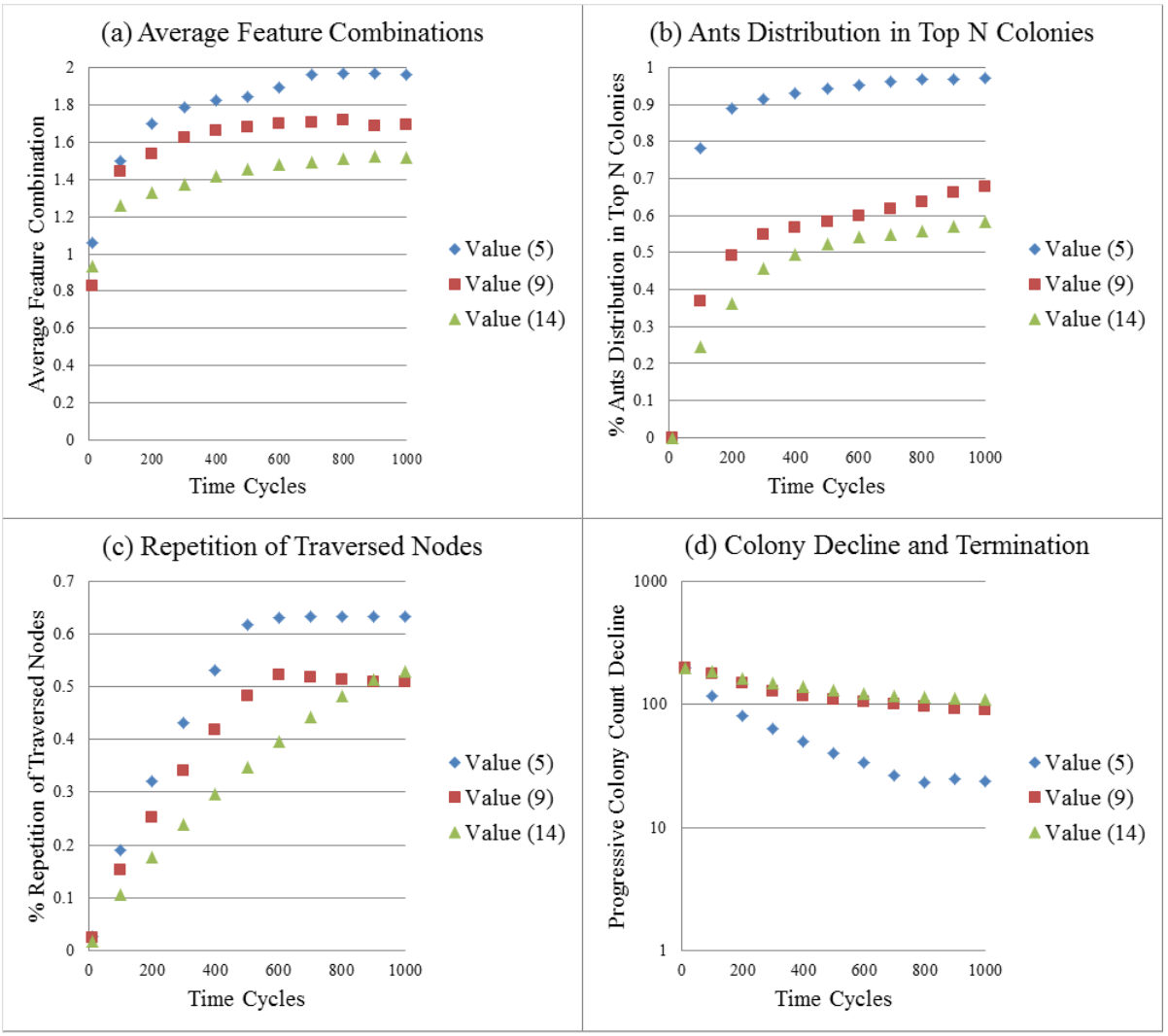

FIGURE 3.2: Results of varying the maximum edge length parameter are given as follows; figure (a) shows the average feature combinations carried by ants, figure (b) shows the distribution of ants in top N colonies versus other smaller colonies, figure (c) shows the repetition in traversed nodes, and figure (d) shows the progressive decline in the number of colonies present and the termination criteria.

3.8.3.1 Maximum Edge Length Parameter

Increasing the maximum edge length, increases the graph connectivity. This causes two events to occur; (i) distant nodes become connected to each other, and (ii) the number of outgoing edges from each node increases. This parameter determines the likelihood of ant encounters, influencing activities such as feature and colony merging. As shorter edges are traversed more quickly, evaporation has less time to take place. Thus, the impact of pheromone quantity is higher.

The optimal value of this parameter (within this domain being 9), is one which is long enough to allow proper connectivity and induce an adequate frequency of ant encounters, whilst not producing a fully connected graph. Figure (3.2, a) demonstrates that a lower edge length causes

the average feature combination to rise too quickly. This since lower edge lengths cause ants to stay located within the same area of space. Likewise, figure (3.2, a) also shows how this influence differs when larger lengths are selected. On increasing edge lengths the influence is more subdued than on decreasing edge lengths, this due to the impact of pheromone that drives ants.

Shorter edge lengths adversely increases the speed at which ant colonies merge. Having ants merge into the top N colonies too quickly, as is shown in figure (3.2, b) is an indicator that proper exploration has failed to take place. Longer edge lengths delay this process, giving the ants the ability to perform a broader search.

Ant movement is driven by pheromone quantity, thus shorter edge lengths cause more repetition in node traversals. What is interesting to note is that the variation in results between different edge lengths is not so protracted. This is due to the effect of pheromone deposited on connecting edges, which stabilises the overall stochastic behaviour [figure (3.2, c)].

Termination is quicker with a shorter edge length, since ants are more likely to be assimilated into colonies. This process is delayed with longer edges [figure (3.2, d)]. Therefore, too small values applied to this parameter inhibit correct clustering. Larger values still allow clustering to take place, however this parameter determines the speed of such occurrence.

3.8.3.2 Step Size Parameter

The step size parameter influence two key activities. Firstly, it determines the number of steps that exist within an edge, and secondly it determines the granularity of the search. This next experiment considers two cases, discussed earlier, having a step size of 0.1 of a SD or 0.4 of a SD, and demonstrates the optimality for the former parameter value.

Figure (3.3, a), shows that having a lower step size produces results which are in line with other experiments. However, on increasing the step size, the shortening of edges and the ability of ants react to less granular features, makes ant encounters more frequent. Thus, feature merging immediately rises to a high value, excluding the possibility of proper exploration.

Similar patterns of activities are shown in figure (3.3, b), where colony formation occurs way too rapidly for the larger step size. This is accelerated to an extent that all ants are absorbed into the top N colonies. Even more so, as shown in figure (3.3, d), termination occurs too suddenly, and most ants combine into just one colony.

Larger step sizes also induce repetition of node traversals, as per figure (3.3, c), resulting in a value which is too high for correct performance. The closer proximity of nodes and the elevated

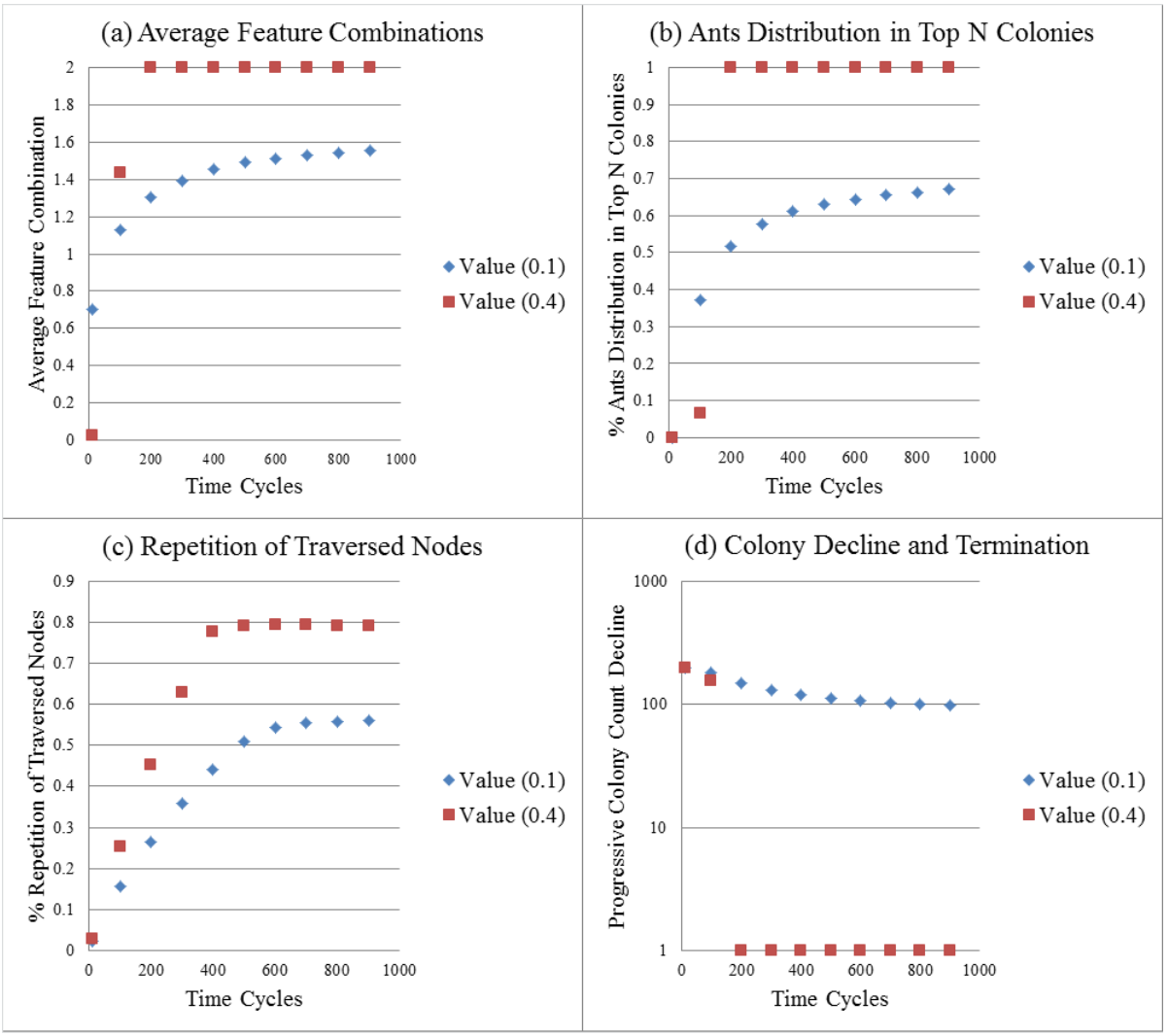

FIGURE 3.3: Results of varying the step size parameter are given as follows; figure (a) shows the average feature combinations carried by ants, figure (b) shows the distribution of ants in top N colonies versus other smaller colonies, figure (c) shows the repetition in traversed nodes, and figure (d) shows the progressive decline in the number of colonies present and the termination criteria.

pheromone connections cause ants to loop within the contained space. The smaller step size returns a more workable percentage repetition.

The above happens since the increased step size reduced edge lengths and heavily increased spatial compression, whilst at the same time also increased the ants ability to react to a wider range of features. Hence, both these factors compound to distort the final clusters which form. This experiment serves to demonstrate that a smaller step size returns a much finer and more workable granularity.

3.8.4 Ant Initialisation Analysis

The ant complement and feature detection ranges are important influences on all parameters, as evaluated next.

3.8.4.1 Ant Complement Parameter

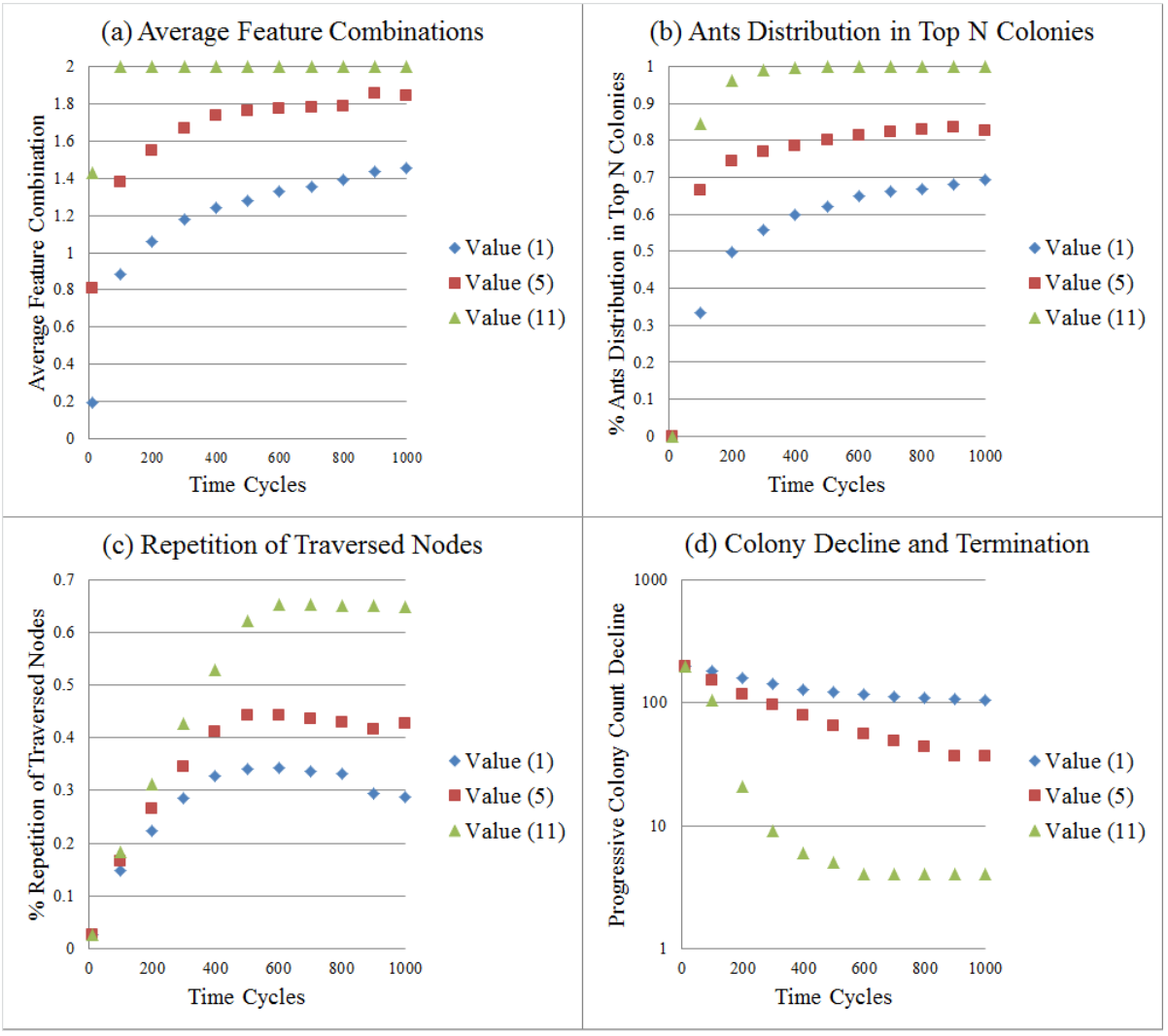

FIGURE 3.4: Results of varying the ant complement parameter are given as follows; figure (a) shows the average feature combinations carried by ants, figure (b) shows the distribution of ants in top N colonies versus other smaller colonies, figure (c) shows the repetition in traversed nodes, and figure (d) shows the progressive decline in the number of colonies present and the termination criteria.

This experiment determines the applicable parameter ranges which provide an optimal balance in terms of the number of ants, which is heavily dependent on the number of data samples present. For a smaller 2D domain, such as the one discussed, a higher ant complement is applicable. An increase in ants impacts all other activities which build on ant encounters. Hence, irrespective of how high the feature and colony merge thresholds are, having a very high ant

complement can nullify their influence. This in turn also dampens the influence of evaporation as more pheromone is deposited. An increase in ant counts also necessitates a proportionally smaller time-window.

Figure (3.4, a) demonstrates that the optimal parameter setting ranges at around 3-5. Exceeding a certain critical mass of ants causes the average feature merge to immediately escalate to the maximum. This is also evident in figure (3.4, b), where lower ant complements return higher variation in the colonies that form, whilst higher ant complements are immediately absorbed into the top N colonies.

The increased ant count increases the pheromone quantity deposited. This in turn increases the chances of ants traversing the same sequence of nodes, reducing randomness. Thus, as is shown in figure (3.4, c), higher complements reduce randomness. Interesting to note, is that randomness is still present. This since the mechanism employed in the MPACA will allow a degree of randomness. This is also coupled by the maximum coefficient which limits the amount of pheromone that can be deposited on any edge. Finally, the ant complement increase accelerates the termination process, as per figure (3.4, d).

3.8.4.2 Detection Range for Ordinal Dimensions Parameter

This parameter determines the granularity of the search; when a lower value is chosen, more specific connections can be made. When increased, broader less specific connections are created. This experiment shows how this parameter influences feature and colony merging, and the pheromone levels within the system.

The optimal setting for this parameter is a low value, in this case a value of one step in each direction. Figure (3.5, a) demonstrates that lower detection ranges produce the most suitable average feature merges. On the contrary, wider ranges cause links between all nodes to be reinforced. This increases the amount of pheromone deposited, with the result that linkages between nodes become less specific or meaningful.

A larger detection range is also detrimental to colony formation, this leading once more to a situation where membership into the top N colonies is too sudden, as shown in figure (3.5, b). A similar pattern is further observable in the repetition of traversed nodes [figure (3.5, c)]. This same repetition causes the ants to remain localised within a small area, potentially causing the ants to triangulate within a set of nodes, and accelerates termination [figure (3.5, d)].

In essence, a small feature detection range value slows down feature and colony merging and allows a more granular search to occur, a process which encourages better cluster recognition.

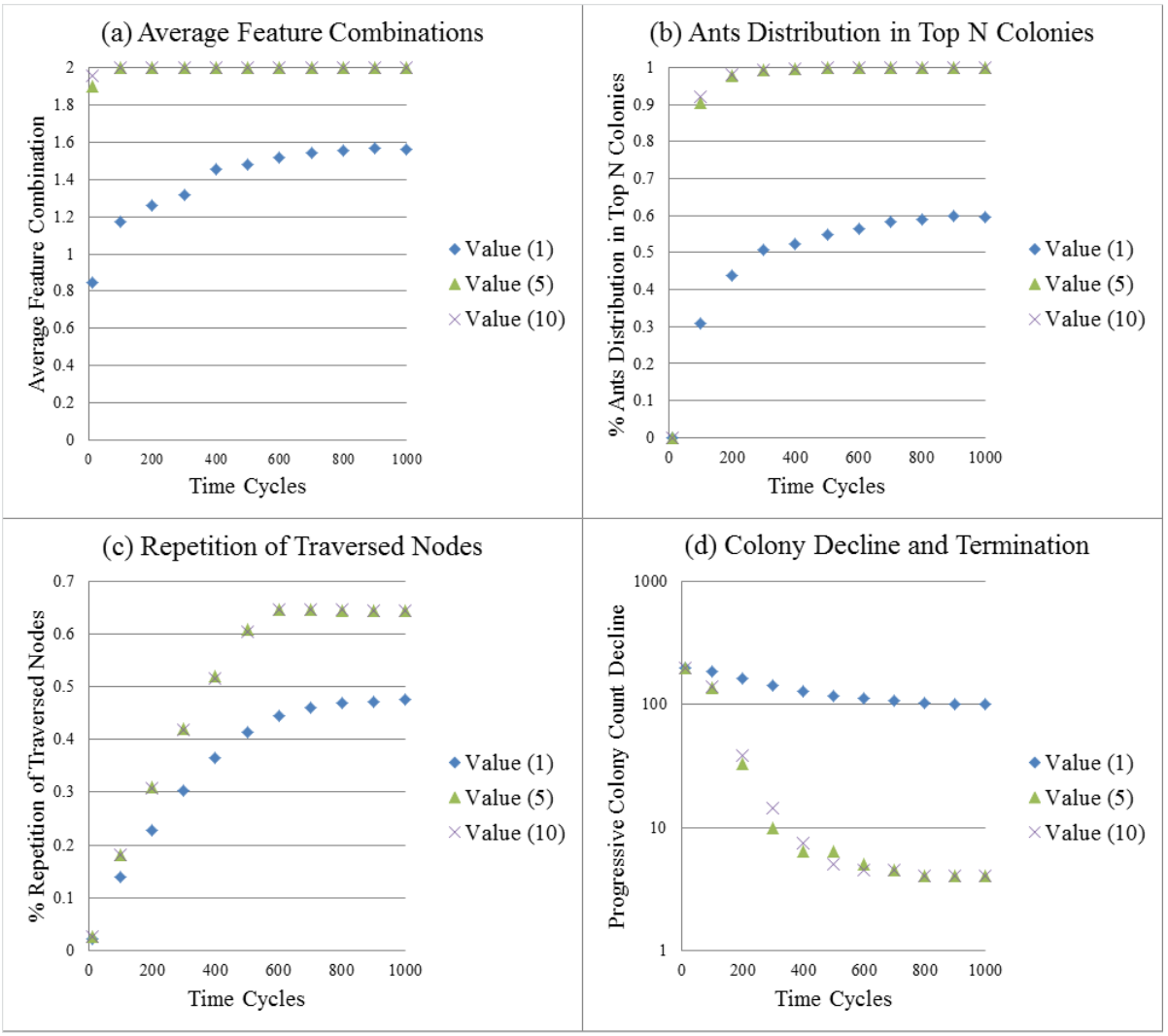

FIGURE 3.5: Results of varying the detection range for continuous domains parameter are given as follows; figure (a) shows the average feature combinations carried by ants, figure (b) shows the distribution of ants in top N colonies versus other smaller colonies, figure (c) shows the repetition in traversed nodes, and figure (d) shows the progressive decline in the number of colonies present and the termination criteria.

3.8.5 Pheromone Deposition and Movement Analysis

3.8.5.1 Pheromone quantity, maximum coefficient and the evaporation parameters

This experiment serves as a measure of the degree of pheromone influence. Ant movement is controlled by pheromone scents present on each edge. The following parameters are effectively an interpretation of one parameter action, which is pheromone deposition, and are hence evaluated collectively. An arbitrary value of one unit has been chosen representing the minimum pheromone tolerance parameter, whilst evaporation and the maximum coefficient are derived from average values presented in table (3.1). The focus of this experiment is on the quantity of pheromone deposited.

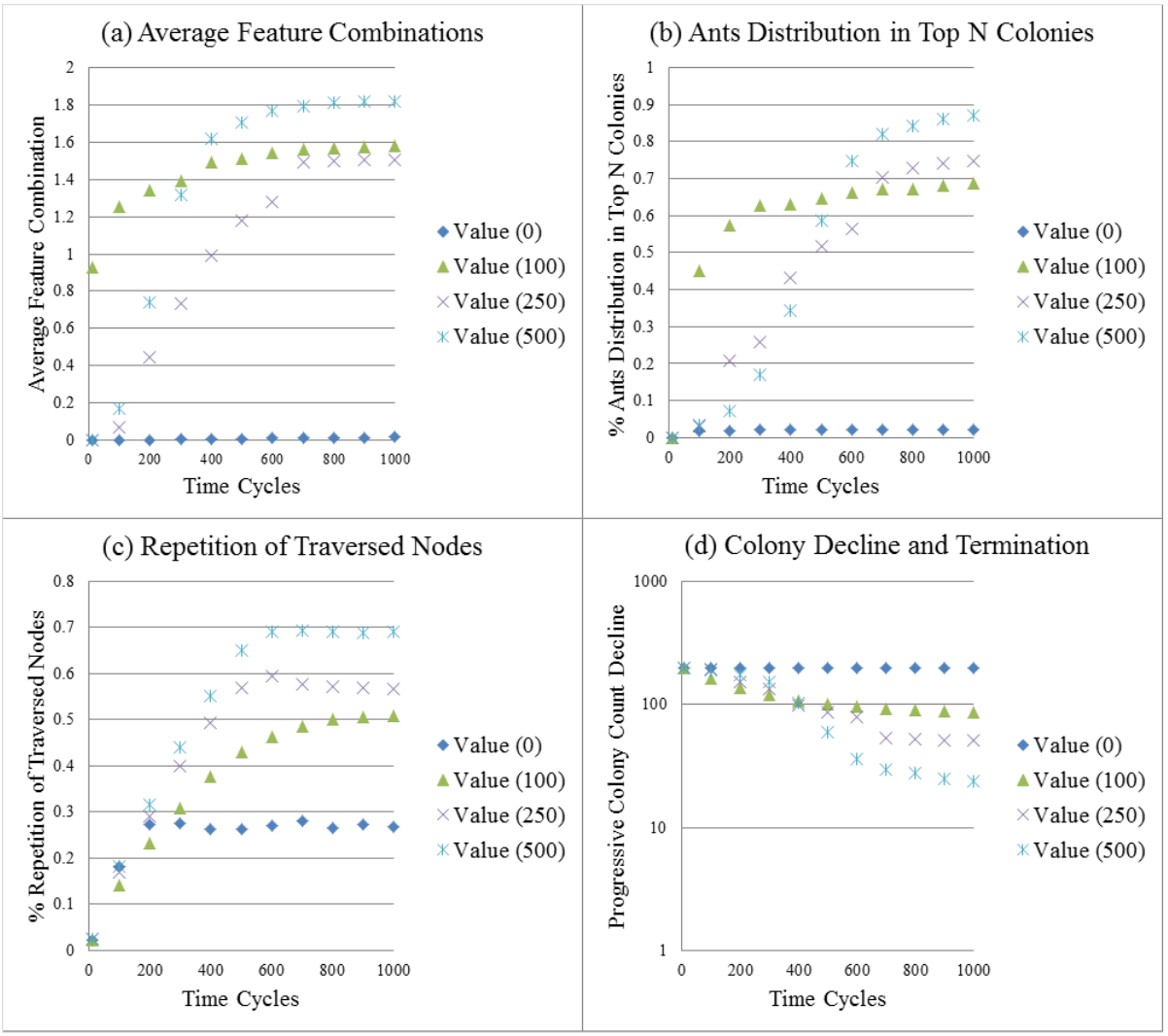

FIGURE 3.6: Results of varying the pheromone related parameters are given as follows; figure (a) shows the average feature combinations carried by ants, figure (b) shows the distribution of ants in top N colonies versus other smaller colonies, figure (c) shows the repetition in traversed nodes, and figure (d) shows the progressive decline in the number of colonies present and the termination criteria.

Pheromone controls the path following feedback mechanism, and as this experiment demonstrates that the optimal value of this parameter is in the range of 100-250 units of pheromone. Figure (3.6, a) shows that without pheromone limited feature merging takes place. The same is deducible in figure (3.6, b), where this inhibits ants from merging into the top N colonies. This lack of cohesion happens as ant traversals within the graph take place without pheromone interaction. As pheromone levels are increased, this increases the likelihood of both feature and colony merging, the former depicted in figure (3.6, a). This also increases colony merging, causing more ants to appertain to the top N colonies in quicker succession. Eventually, the higher level of pheromone present in the system reduces randomness, and hence more ants are likely to traverse a repetition of the same nodes. Interestingly, without the pheromone the ant movement randomness remains near constant all throughout the execution [figure (3.6, c)]. It is also

noted that higher pheromone values quicken termination, and this might lead to the formation of erroneous clusters, whilst also anticipating early termination [figure (3.6, d)].

Therefore, pheromone is required for both feature and colony merging, however overly strong pheromone quantities might obfuscate the search and precipitate termination, limiting the exploration required.

3.8.5.2 Residual Parameter

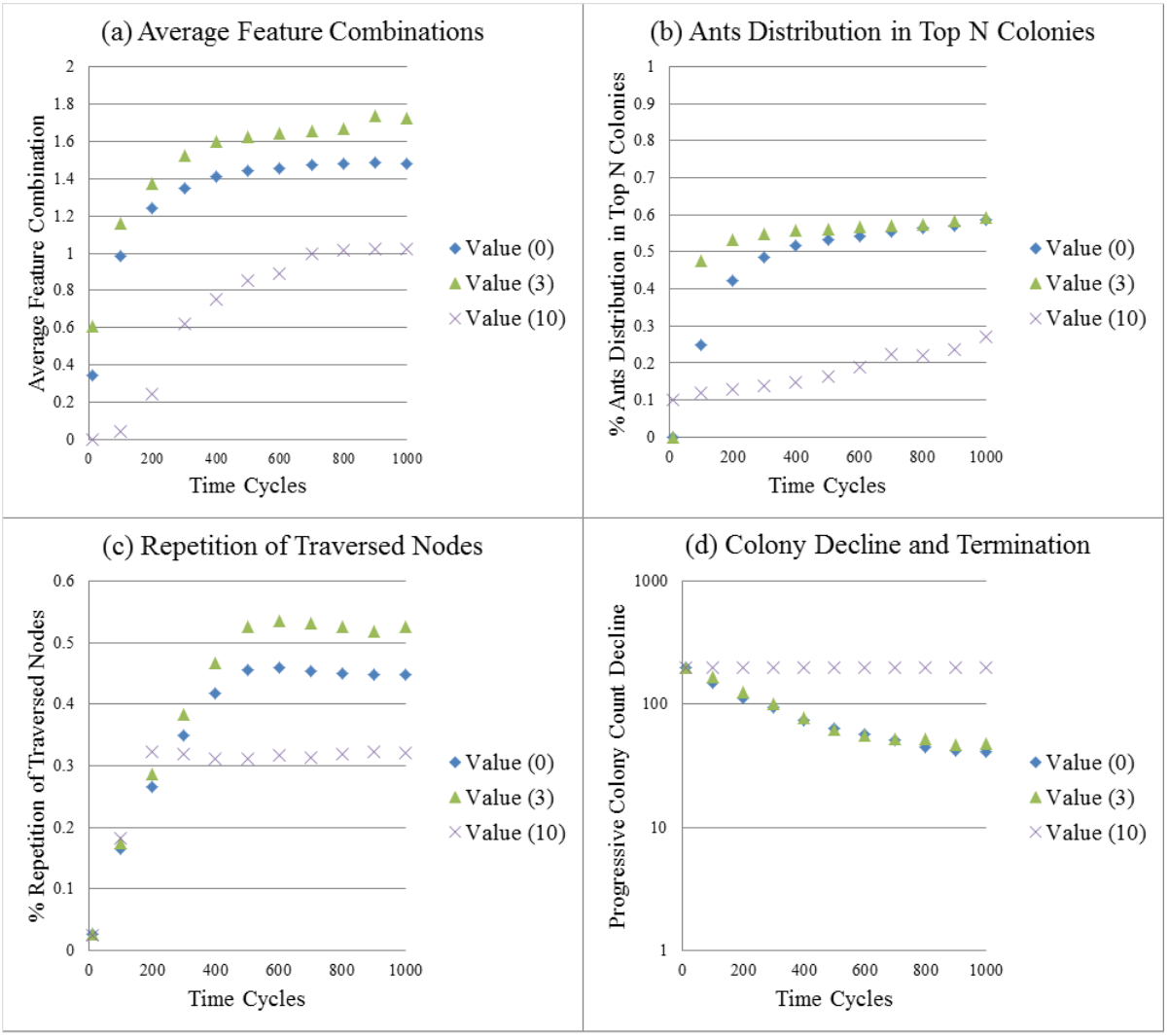

FIGURE 3.7: Results of varying the residual parameter are given as follows; figure (a) shows the average feature combinations carried by ants, figure (b) shows the distribution of ants in top N colonies versus other smaller colonies, figure (c) shows the repetition in traversed nodes, and figure (d) shows the progressive decline in the number of colonies present and the termination criteria.

This parameter is analysed by starting off with a zero value. That is, no additional residual value exists, and ants only follow paths with the highest pheromone quantity. Following which more randomness is included.

This experiment builds on section (3.8.5.1), as it impacts the same parameters, and similar results are achieved. The optimal value for this parameter is set to be a minimal value, ranging between approximately 2% - 3%. This experiment demonstrates that with a minimum amount of added randomness, interactions improve. However, feature and colony merging decrease in frequency as randomness increases, respectively depicted in figures (3.7, a) and (3.7, b). This since less ants head towards nodes of interest to them, reducing the chances of further pheromone deposition. Therefore, the residual randomness has to be contained or else the entire search degenerates. Higher randomness values, even if just 10% have a multiplied adverse effect. This since it inhibits the positive feedback effect of pheromone from occurring. As the pheromone levels decrease they are less likely to be reinforced and dissipate, and ant movement becomes even more random [figure (3.7, c)]. Termination is also severely impacted, with this taking longer to occur, and in many cases failing to do so [figure (3.7, d)].

3.8.6 Merging Thresholds

Ant encounters determine the features and colonies that are to be merged. The number of encounters seen by each ant at each node is determined by the visibility range. Further controlling all this is the fact that all encounters are only temporarily available to the ant, according to a time-window.

3.8.6.1 Feature Merging Threshold Parameter

This parameter influences colony merging and pheromone deposition in the following ways. Lower feature merging thresholds cause ants to immediately combine features as per figure (3.8, a). This premature combination has another side effect, as feature combinations cause ants to rapidly become specific to a particular area of graph space. This increases the likelihood of ants merging into colonies. However, this restricted area causes a number of smaller colonies to form rather than the complete colonies required, as is shown in figure (3.8, b). Increasing the feature merging threshold to a mid-value (in this case 6), increases the quality of colony merging into the top N colonies. Consequently the feature merge delay encourages further exploration of the domain [figure (3.8, b)]. This colony belonging directly influences the termination of the algorithm, and this terminates faster when more ants belong to the top N colonies [figure (3.8, d)].

The influence of this parameter is driven by the various pheromone-feature combinations which are deposited upon merging. An interesting eventuality occurs when a higher feature merge threshold is chosen. High values inhibit ants from merging their features so readily, this causing

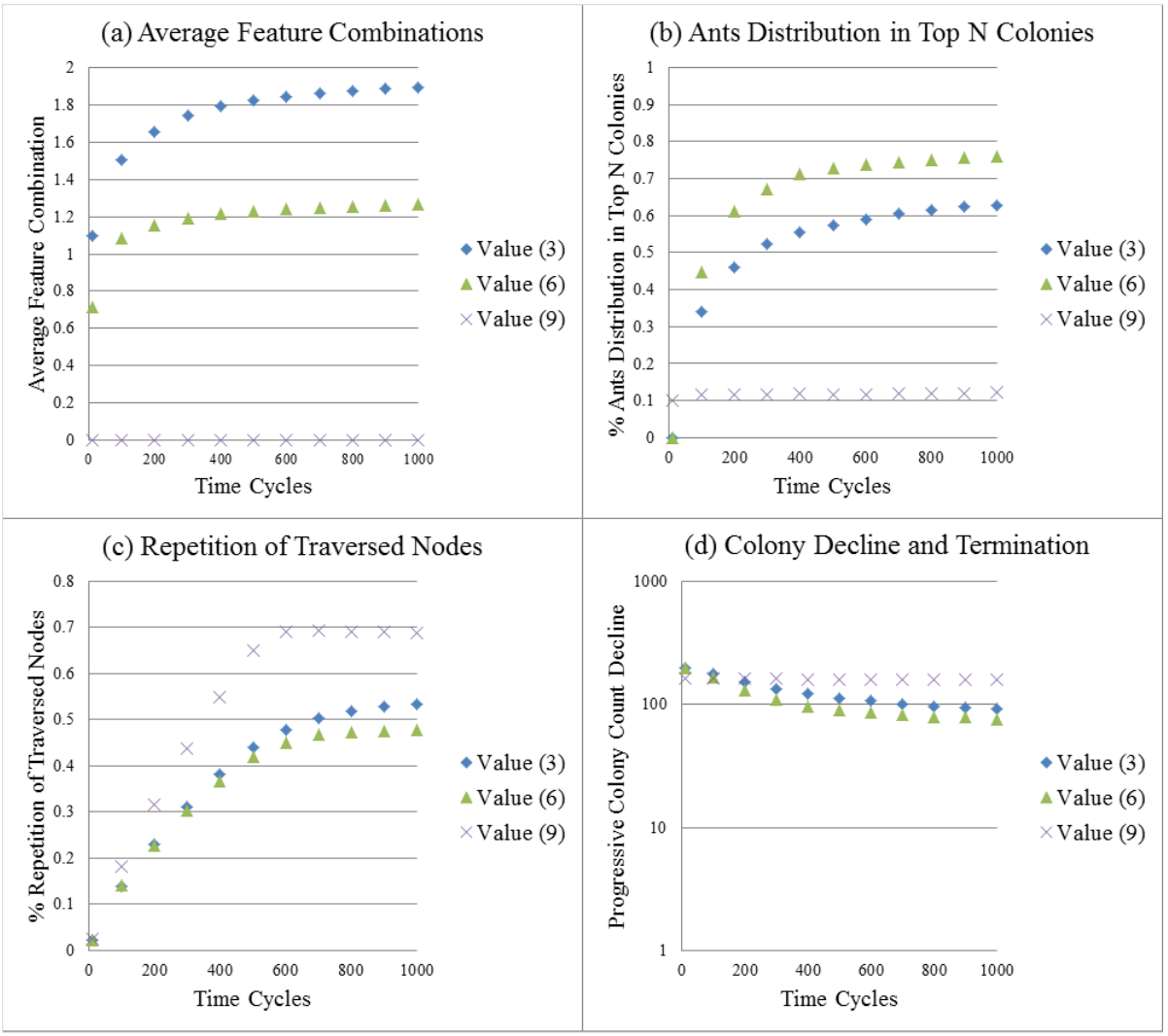

FIGURE 3.8: Results of varying the feature merging parameter are given as follows; figure (a) shows the average feature combinations carried by ants, figure (b) shows the distribution of ants in top N colonies versus other smaller colonies, figure (c) shows the repetition in traversed nodes, and figure (d) shows the progressive decline in the number of colonies present and the termination criteria.

ants to look solely for one feature. This turns the MPACA into a singular feature search, similar to the common ACO, which is known to optimise paths quite rapidly. Thus, as per figure (3.8, c), higher merge thresholds correspond to the highest repetition of nodes traversed. The contrary of this does not occur. Lower merge values as in this analysis, where value 6 is chosen, still allow feature merging to take place, implying that the search is not driven by a singular feature in such a case, with repetition being much lower than the case for value 9. When the feature merging threshold is lowered to value 3, in this case repetition increases slightly. This limited increase occurs as the lower feature merge causes the ants to stabilise quicker the features they are after, which in turn is reinforced more often than would occur if the ants keep exchanging the features they are carrying.

3.8.6.2 Colony Merging Threshold Parameter

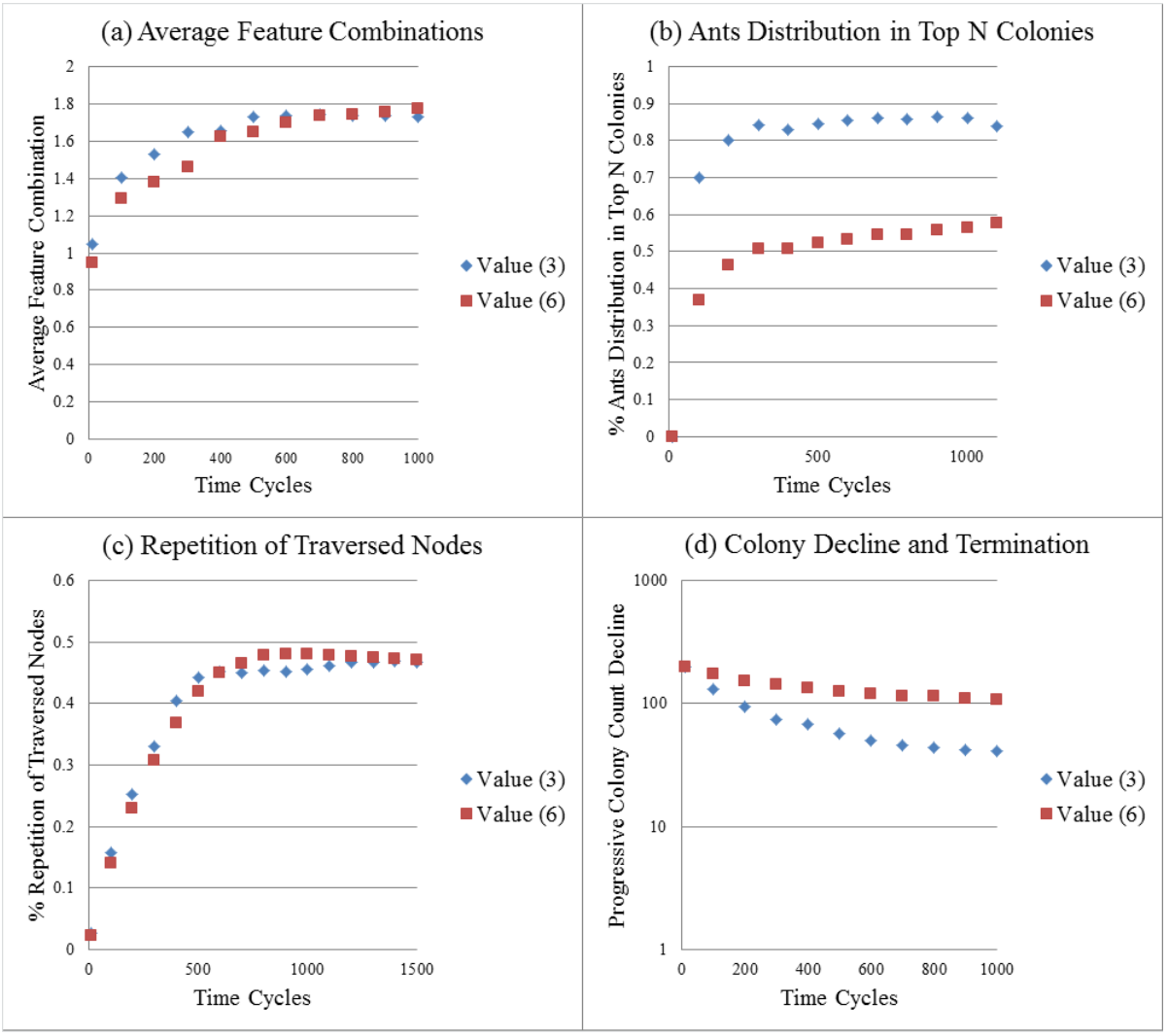

FIGURE 3.9: Results of varying the colony merging parameter are given as follows; figure (a) shows the average feature combinations carried by ants, figure (b) shows the distribution of ants in top N colonies versus other smaller colonies, figure (c) shows the repetition in traversed nodes, and figure (d) shows the progressive decline in the number of colonies present and the termination criteria.

The colony merging threshold is the founding mechanism for cluster formation. Colony membership is an isolated parameter as it does not influence any other parameter. It is only meaningful in the context of the colony formation. Thus, as figures (3.9, a) and (3.9, c) demonstrate, there is limited statistical significance difference (excluding that attributable to sampling and inherent randomness within the model) when varying colony membership levels vis-a-vis feature merging or ant movement. Hence, whilst feature merging directly influences colony formation, the opposite is not so. This is due to the fact that a colony can have ants with multiple feature combinations, and search specialisation occurs at the ant level, not at the colony level. On the other hand the two benchmarks affected by this parameter change are colony membership into the top N colonies, and the termination criteria, respectively depicted in figures (3.9, b) and

(3.9, d). Low colony membership means more ants are engulfed into the top N colonies, and algorithm termination on the other hand is heavily dependent on the colony merging threshold. A low threshold implies that excessive merges occur, which can inhibit the algorithm from reaching correct termination.

3.8.6.3 Visibility on Edge Parameter

This parameter gives the ants the ability to determine the internal content of other ants. The more extended this visibility is, the greater is the number of ant encounters counted.

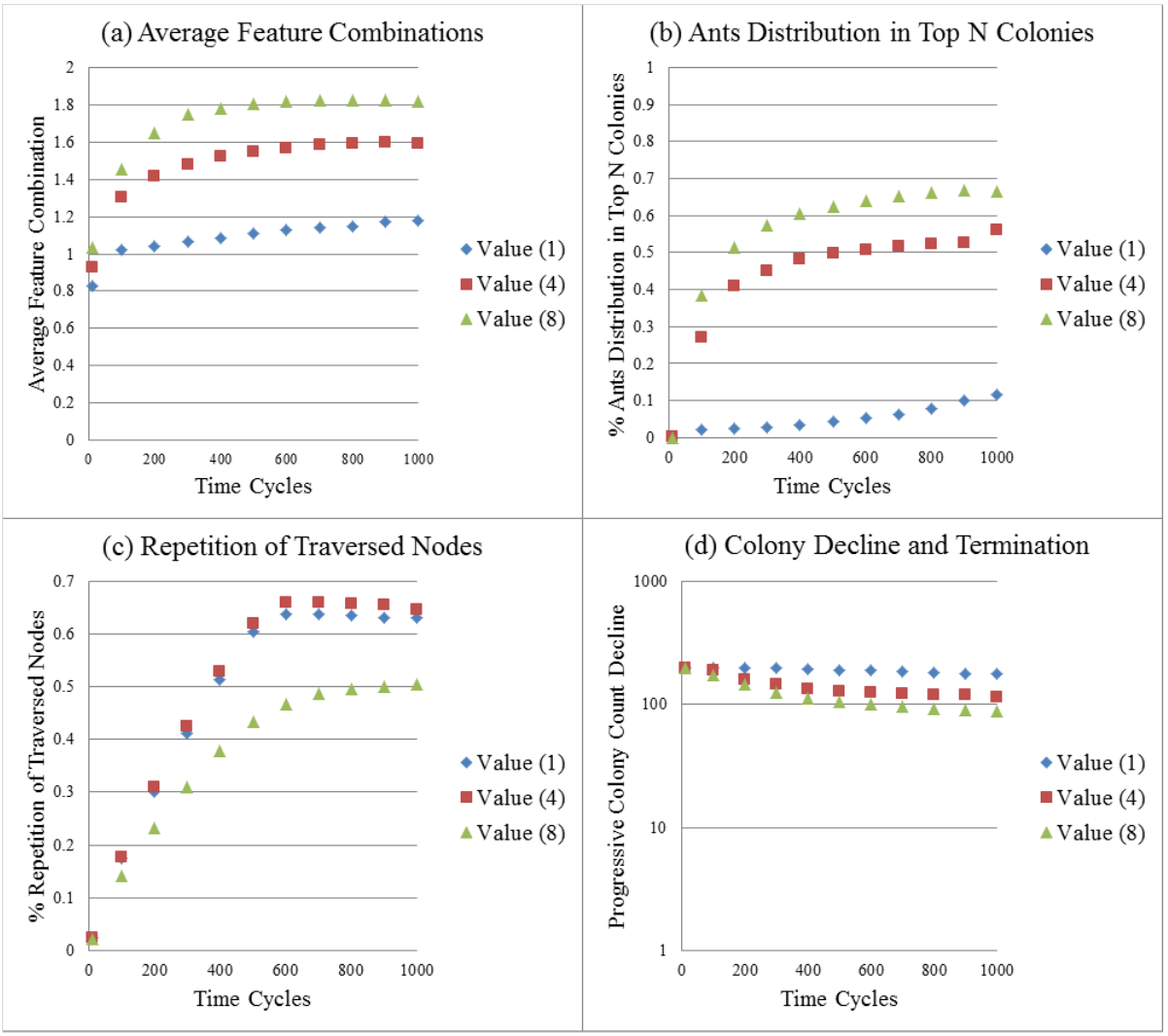

FIGURE 3.10: Results of varying the visibility range parameter are given as follows; figure (a) shows the average feature combinations carried by ants, figure (b) shows the distribution of ants in top N colonies versus other smaller colonies, figure (c) shows the repetition in traversed nodes, and figure (d) shows the progressive decline in the number of colonies present and the termination criteria.

In this experiment three values are considered; (i) single visibility where ants can only see one step away, (ii) partial visibility where ants can see steps belonging to approximately half an edge length away, and finally (iii) complete visibility where ants can see all ants present on the edge.

Visibility elongation accelerates feature and colony merging, as the more ants are detected, the more frequently these operators take place [figure (3.10, a)].

Figure (3.10, b) shows that partial visibility takes longer to place ants within the top N colonies than full visibility, whilst single step visibility is considerably slower than both. Furthermore, figure (3.10, c) shows that both single visibility and partial visibility have similar repetitive traversals. This is likely to occur for different reasons. Single visibility operates on a single feature, much in the same way as when a high feature merging threshold is chosen, whilst partial visibility performs node traversals seeking feature combinations. On the other hand complete visibility tends to have lower repetitive traversals, which can be symptomatic of incorrect feature merges which do not create correct pheromone traversals to be followed. Thus, even if feature and colony merging is faster at the maximum value, the partial setting is operationally more coherent. A large visibility increases the ant encounters, which precipitates termination.

3.8.6.4 Time-window Parameter

This parameter has a primary impact on feature and colony merging, as the longer the time-window, the higher the rate of feature and colony merging should be, and vice-versa. This experiment demonstrates the variations of the time-window as this is increased at intervals, starting from 50, 100, 250 and 500 units. Experimentation shows that for this domain, the optimal value is around 100 time units.

Short time-windows imply a low feature merging, and respectively a low feature merge average, as per figure (3.11, a). A similar activity takes place as the lack of having a specific set of features, and the lack of historical ant encounters imply that colony formation takes longer to occur at this value [figure (3.11, b)]. As is shown in figures (3.11, a) and (3.11, b), as the time-window is gradually increased, feature merging increases, and so do the ants belonging in the top N colonies.

The disparity between ants depositing pheromone and those that are not, and even the randomness in search is not particularly influenced by the time-window. This is because the likelihood of ants locating interesting nodes is unchanged [figure (3.9, c)]. Algorithm termination is directly influenced by colony formation, hence a delay in colony formation consequently delays termination [figure (3.11, d)].

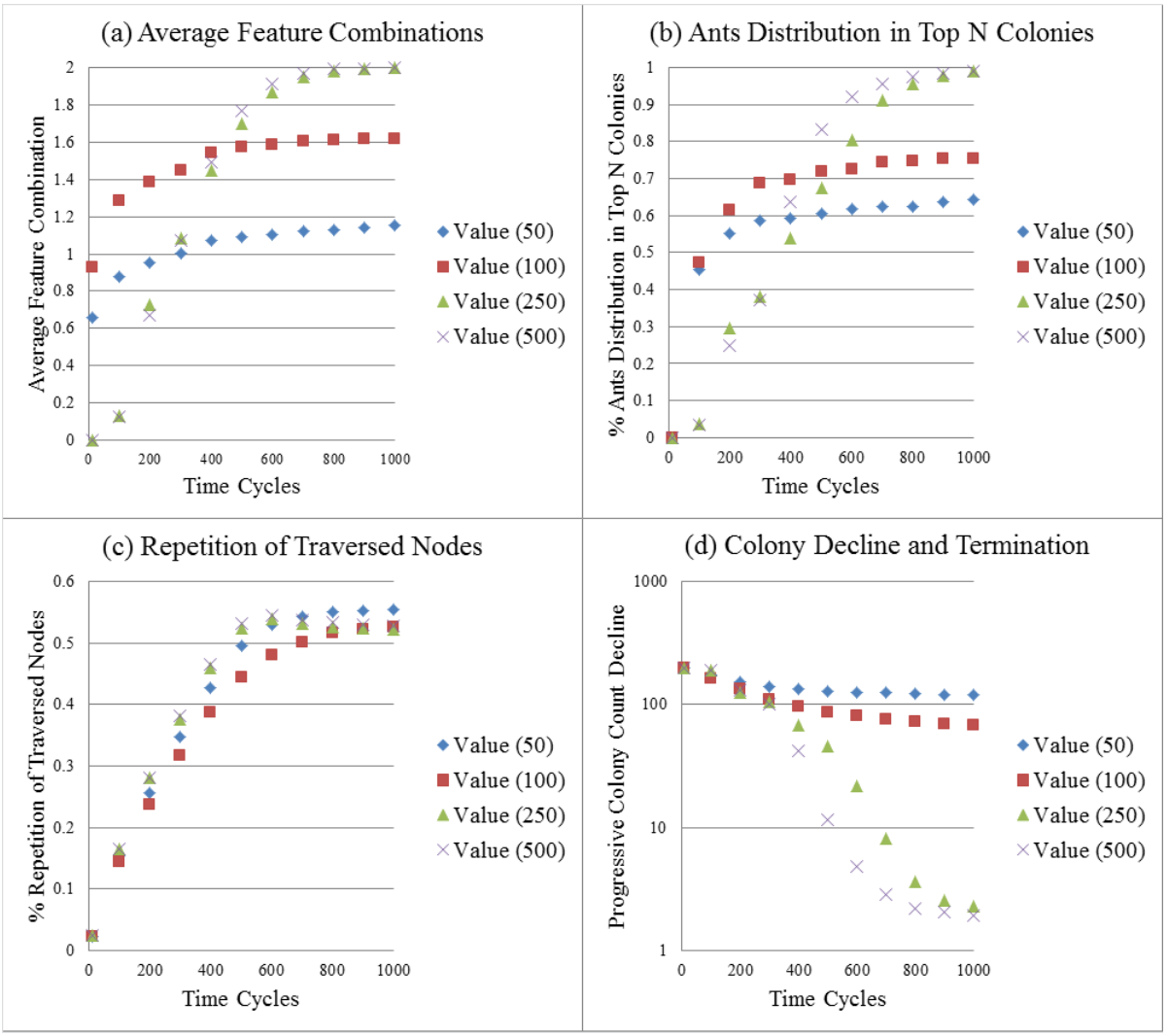

FIGURE 3.11: Results of varying the time-window parameter are given as follows; figure (a) shows the average feature combinations carried by ants, figure (b) shows the distribution of ants in top N colonies versus other smaller colonies, figure (c) shows the repetition in traversed nodes, and figure (d) shows the progressive decline in the number of colonies present and the termination criteria.

3.9 The MPACA as a Classifier

The MPACA has an additional second operating mechanism, that of a classifier. This is additional material which is only briefly introduced, serving as a prelude to ongoing and future work. Within this classification method two modes of operation exist, a training mode and a testing mode.

The operator using the MPACA is required to split the dataset into two sets accordingly, one for training and one for testing. The dataset is first normalised using the both subsets combined.

During the training mode, only data elements which pertain to the training set are used for graph formulation. The same MPACA learning process takes place with the notable exception that

ants are initialised to belong to a specific colony and are prohibited to migrate or merge into or with any other colony. That is, colonies of ants are assigned to fixed classes. What changes is the composition of features carried by the ants themselves, this since ants can still learn and drop features. At the end of training, ants will have acquired feature combinations, and allocated pheromone weights on edges connecting nodes.

When in testing mode, the entire dataset (training and testing) is once more used to reconstruct the graph structure. The training data elements are evaluated against this new structure as is learnt by the ant colonies. This evaluation process returns the most likely class membership that an unseen data element belongs into.

A number of more specific considerations include the termination criteria and the evaluation mechanism.

3.9.1 Training Termination Criteria

Classification learning is achieved by allowing the algorithm to run until the carried features within the ants stabilise or a maximum number of training iterations is set. To date this process is manually controlled. The speed of learning can be gauged, this since the MPACA allows the monitoring of the internal visibility of the average number of carried features occurring during processing, once the average number of carried features by the ants exceeds a reasonable threshold, this value can be arbitrarily used to terminate the training cycles. A reasonable threshold is considered to be one where the average ants carry an average of $N-1$ features, where N represents the total number of features present in the system.

3.9.2 Evaluation of the MPACA as a Classifier

At the end of learning, an ant colony has various ants each carrying a number of features. This collection of features determines class membership. The evaluation process works by comparing the testing data elements and determine which colony (class) they fit in best. This is once more the same mechanism as is used in the standard MPACA, with the difference being that rather than assigning a data element to a colony representative of a cluster, this is assigned to colony which represents the class. This process is achieved by analysing the composing features and feature values of the data element using the same mechanisms presented in section (3.7.3.1) [further extendible by methods presented in section (5.5.1, 5.5.2)].

3.10 Novelty and Contribution of the MPACA

The MPACA is a bottom-up, distributed swarm based clustering technique. It is a fully fledged decentralised ant colony algorithm, driven by ant interactions, where in turn movement is controlled by pheromone scents. Pheromones create the corresponding positive feedback that causes conglomeration of ants in space, which leads to clustering.

3.10.1 Distinctive Elements of the MPACA

Unlike most other ant algorithms found in the literature, the MPACA is used to cluster graph space in a spatial partitioning process. The main differences between the MPACA and other ant models are:

1. The ant chooses the edge to traverse next stochastically, where the edge with the highest amount of similar pheromone is preferred, but not always chosen. Pheromone stigmergy implies that nodes with similar feature values have higher levels of pheromone connecting them. As evaporation of pheromone occurs, the least reinforced edges are selected less frequently and therefore tend to have lower pheromone levels.
2. The MPACA includes a mechanism which allows ants to learn feature combinations, or interactions, that are central for clustering and classification. The underlying concept implies that features which frequently co-occur should be combined at the ant level. Ants record features carried by other ants during ant encounters. These ant recordings take place when the ant has reached a node. The focus ant locates ants within its visibility radius and records the feature values carried by each ant. If the feature encounters exceed a parametrised threshold, the ants combine each other's features. This results in ants now responding only to a combination of these features. This ability to learn feature combinations enables ants to pick up non-linear interactions. In response to such feature combination merging, each ant deposits a pheromone representative of a particular feature or feature combination it is after. This makes the MPACA a multi-pheromone mechanism like no other, as no other mechanism in literature has the ability to handle this many pheromone combinations.
3. The MPACA has the ability to perform clustering via the use of multiple colonies. The merging of colonies is similarly driven by the frequency of ant encounters. During the same ant encounter, the focus ant also stores colony Ids that pertain to the other detected ants. If the number of colony encounters for the focus ant exceeds the threshold for colony merging, then the focus ant updates its colony Id property accordingly. Both the feature

and colony merging processes are limited by a time-window. More importantly both merge operators are executed asynchronously and at the ant level, thus decentralisation is ensured.

It is not obvious whether these distinctive properties mean the MPACA will have equally distinctive behaviour.

3.10.2 Variations of the MPACA from the Traditional Clustering Algorithms

As highlighted in section (2.2.5), the MPACA is locatable under the graph-theoretic clustering algorithms, section (2.2.4.4). The graph represents the problem domain, where node proximity in space also signifies a higher degree of relevance. Ants in the MPACA traverse the graph structure and lay pheromone to further link nodes of higher relevance. The effect of this pheromone causes ants to form higher densities around certain regions at the expense of others.

The MPACA does not utilise a direct probabilistic mechanism to create clusters. It instead uses the ant population density to define the clusters, not the original objects. It then assigns nodes to the nearest ant colony, which is a representative of a cluster. This is a density function which differs from that presented in section (2.2.4.3), since the MPACA uses the frequency of ants, which are themselves proxies of node content, rather than making use of node density itself. This can be interpreted as a density-by-proxy or a mocked density function. Ants in the MPACA are not interested in nodes per se, but rather in the features that make up the node. These feature and ant encounters are used for the two-fold merging mechanism, applied to both features and colonies. The MPACA merges features which are carried by the ants depending on the frequency of their encounters, and also ants from different smaller colonies merge into bigger ones, using a similar mechanism. The MPACA uses the critical number of ants to merge into one colony, based on a colony level threshold. This can be considered as equivalent to the $MinPts$ in DBSCAN [Ester et al., 1996b], since once a colony level threshold is exceeded, two colonies merge.

The MPACA can be viewed as a hybridisation of the algorithm methodologies presented, since it uses the concepts of density as in the DBSCAN [Ester et al., 1996b] to form clusters, and the K-Means [MacQueen, 1967] to allocate data elements to one cluster or the other. It also uses a normalisation function, sharing similarities with Grid-based clustering (section 2.2.4.3). These classical clustering approaches serve as a benchmark for results.

3.10.3 Variations from Ant Based Clustering Literature

This subsection in part explores a global overview of the variations the MPACA variations from the algorithms presented in the literature overview chapter (2), based on typology groupings presented earlier. In this text ACO stands for typologies of type IV and their sub-derivative algorithms.

Type I: Akin to the phenomenon of self-assembly in the AntTree algorithm, ants in the MPACA also use self-assembly. This differs, as rather than building fixed structures, where the structure is the representation of the cluster, in the MPACA ants merge into colonies of ants, forming a dynamic structure. In the MPACA it is this colony of ants and the carried features that ultimately represent a cluster. This structure is dynamic, as ants can easily move from one colony to another. In the MPACA ants follow other ants which are searching for similar features. When a locale is found which has many ants searching for similar features, there is an increased tendency for the ant to stay fixed within it. The modus operandi of the AntTree, as discussed in section (2.6.1), remains that of building a hierarchical structure and cluster definitions are extracted from it. This is totally distinct from the MPACA which is based on ACO principles (type IV).

Type II: The MPACA uses a graph based architecture where data elements are represented as nodes, which are fixed during the clustering process. This differs substantially from the SACA approach in which data elements are represented as blocks within a 2D space and are moved around by ants to form clusters of higher density. Hybrid-SACA approaches introduce the notion of pheromone driven ants, such as the ACLUSTER and APC. In these algorithms the final spatial positioning of data elements is a result of the clustering process itself, whilst in the MPACA graph nodes are fixed. The notion of graph clustering clearly separates the MPACA from any type II method discussed earlier in section (2.6.2).

Type III: The ANTCLUST algorithm has introduced clustering based on the agreement of a colonial odour between ants. That is, the ants determine what pheromone values best represent the colony. This approach introduces a "learning" function within ants, as ants adjust their pheromone according to the interaction with other ants. This is a mechanism akin to the MPACA, where ants rather than learning pheromone carried by other ants, learn which features they should follow, and into which colonies they should join into. But the similarities with ANTCLUST are also limited as has been discussed in the relevance to the MPACA, as per section (2.6.3).

Type IV: The MPACA is graph based, and shares analogies with the core ACO principles. That is, ants traverse graph space, and during these traversals deposit pheromone traces. This

chapter has reviewed two different ACO (graph) based approaches. In the first instance, the problem is used for purely optimisation purposes, where at the end of each iteration a fitness function is applied to calculate the best tours performed by the ants and in order to determine the quantity of pheromone that is to be deposited. Other multi-objective, multi-colony and multi-pheromone implementations use an extension of this approach. All make use of the core ant colony optimisation mechanism, thus even if they do provide a multi-colony aspect, they are still distinct from the MPACA. Although some previous ant models have multiple pheromones, none of them have a different pheromone associated with each distinguishing feature of the objects being analysed. This is a key innovation of the proposed model.

In the second case, one finds the closest resemblance to the MPACA, where pheromone traces are used to create connections between nodes of higher relevance, and where node proximity also signifies a higher degree of relevance. This is an approach which follows on from graph theoretic clustering, section (2.2.4.4). Much like ACODF, the MPACA uses this ant population in specific areas of higher density to create clusters. The MPACA is a mechanism which shares similarities to ODUEC, where colonies compete to colonise nodes, whereas in the MPACA colonies seem to compete to take in more ants within their fold. Ants in the MPACA form into bigger colonies depending on the frequency of their encounters. If an ant detects many other ants belonging to one colony rather than another, there is a higher likelihood of the ant joining that colony. Once more, akin to ODUEC, the MPACA uses multiple pheromones. The effect of this pheromone causes ants to form in higher densities around certain regions.

Despite the MPACA being a graph-based clustering algorithm, a substantial effort has been invested in pursuing a graph architecture that is scalable and efficient.

3.10.4 Advantages of the MPACA Architecture

The MPACA is a graph-based spatial clustering algorithm, [section (2.2.4.4)], where ants populate specific areas of the graph in various densities. To achieve this, ordinal dimensions of the problem space are normalised in such a way that the effects of outliers are minimised, whilst the distances between nodes are contained, a process which allows the optimisation of ant movement and an increase in the likelihood of ants encountering each other. These values are then used to formulate a graph [section (3.3)]. The graph structure in the MPACA is not fully connected, as outlined in section (3.3.1). This architecture is useful since overly distant nodes do not necessitate to be connected. This also solves the problem of overly sparse ants having too many edges to choose from. This set-up allows the MPACA to perform well, irrespective of the size of the problem domain. If large dimensions are not normalised, this may lead to issues

pertaining to the curse of dimensionality, section (2.2.2.4). This could happen as the dispersion of data elements makes sampling inaccessible. Normalisation, although mechanically heavy, aides the MPACA in improving its operational capabilities.

The MPACA builds the same graph architecture irrespective of the sequence of data entry. This results in a consistent output, unlike that achieved by the algorithms presented in section (2.6.2), as the latter algorithms build variational clusters depending on the initial data entry.

The MPACA is not just any other modified ACO algorithm, it is radically different. The MPACA as a clustering algorithm has novelties which distinguish it from any other clustering algorithm presented in chapter (2). It consists of a collection of novelties, which although can be identified in other algorithms, are collectively unique. This section continues to build on sections (1.4) and (3.10.1), which respectively outline the objectives and the distinctive elements of this algorithm.

3.10.5 Novelty in Ant Movement

Ant movement is driven by the edge selection mechanism, which is itself solely driven by the pheromone deposited on edges. The mechanism is unique to the MPACA, as other ACO algorithms use transition functions [equations (2.22, 2.25)], whilst the MPACA uses an increased stochastic mechanism, section (3.5.2). This mechanism increases further exploration. Any edge can be selected irrespective of the amount of pheromone present. Edge selection is biased towards those edges which have higher pheromone quantities, however this does not eliminate edges with little or no pheromone irrespective if in the opening, middle or closing phases of the algorithm execution.

Another variant is the mechanism by which ants deposit pheromone, section (3.5.1.2). Unlike other ACO algorithms, pheromone deposition takes place at each edge traversal, and the quantity of pheromone to be deposited is not proportional to any fitness function. The ants deposit pheromone equally on all edges they are traversing, for each feature value they are carrying, as long as they are in deposit mode, making the MPACA more decentralised than other approaches.

Another paradigm shift within this movement mechanism is the existence of multiple steps within edges, section (3.3.3). This compensates for the lack of a distance or visibility function as in equation (2.22). This visibility in other ACO algorithms artificially increases the chances of ants choosing shorter edges over longer edges. The MPACA uses solely the pheromone values to portray both notions of distance and quality, making it truly a pheromone driven implementation. Longer edges, which take longer to traverse, effectively cause a punishing factor, since they are more prone to evaporation.

Another advantage of such a movement mechanism is that it encourages asynchronous ant movements. This since the MPACA does not require a fitness function, since the algorithm operates differently from standard ACO, where clustering problems are converted into an optimisation problem, section (2.6.4). By avoiding a centralised fitness function, this enhances the decentralisation of ants within the solution. Together with this, ants do not use a meta-heuristic function to determine the next edge to be selected, as this is entirely driven by the pheromone quantity which exists on the edge. This avoids the need to use a heuristic function to determine the quality of the edge being selected. All the information needed for movement selection is extracted from the pheromone concentrations which exist on the edges.

In the MPACA the ant does not need to keep a TABU list of all edges which it has visited. This is limited to just the previous visited node, or TABU list of size one. This means that the ant within the MPACA is a simple operand, which does not perform complex operations and only attempts to follow edges with the strongest pheromone signal which is of interest to it.

The only movement mechanism in literature which offers some resemblance to the MPACA pertains to ACODF [Tsai et al., 2004]. However, even so, as reviewed in section (2.6.4.4), in ACODF ants reduce the number of nodes they visit at each step, with the influence of pheromone trail intensity, which encourages ants getting localised within a particular set of nodes. The MPACA does not reduce the number of edges that can be visited at each iteration. The number of edges that can be visited from a node remains as is set-up during the graph initialisation phase, and remains unchanged throughout. Both the ACODF and MPACA vary in their transition mechanism from traditional ACO, however the mechanism used in the MPACA is still unlike the one presented in ACODF.

3.10.6 Ability to Learn and Acquire Features

Ants learn features depending on feature encounters. This key distinctive feature is truly unique to the MPACA. There is no other ant algorithm in literature which implements this mechanism. This mechanism allows colonies of ants to have ants with heterogeneous feature values, which adjust over time. This means that a distinct ant within a colony could for example match feature values colour “blue” and size value 15, and another ant within the same colony could match colour “green” and size value 15. Hence, at the ant level there is a distinction between nodes which have different colours, however at the colony level both coloured combinations are marked as positively belonging to the colony. This constitutes the implementation of an OR operator. This feature merging mechanism is also entirely decentralised and operates at the ant level.

3.10.7 Multiple Pheromones and Multiple Colonies

Ants within the MPACA use pheromones as their guiding principle, as in ACO, a concept which is adjusted and extended further. The intricacy of how the pheromones operate over each and every feature and feature combinations is a founding pillar behind the MPACA. At the time of writing, and to the knowledge of the author, there is no other ant algorithm which uses the multiple pheromone and feature learning abilities as presented in the MPACA.

The process of ant movement and ant encounters, sections (3.5, 3.5.3) respectively, lead towards the coupling of features by ants. These ants are eventually grouped into colonies which represent clusters. Therefore, a cluster consists of a number of co-occurring feature combinations within the ants present in the colony, allowing clusters to occur even with missing values, since not all ants need to merge on all feature dimensions. This process of learning features implies that after a feature merge, ants deposit a combination of pheromone feature values, and not just one single mono-thematic pheromone value. Although such a mechanism is not correlated to any biological counterpart, it does rekindle links to real ant colonies, where subsets of ants within the same colony that have different objectives, and may lay down distinctive pheromones accordingly [Dussutour et al., 2009]. Computational models of ACO usually exploit multiple pheromones to distinguish between colonies, not ants within colonies. The lack of multi-pheromone approaches in ACO literature is apparent, with limited exceptions [Ngenkaew et al., 2008b], [Ngenkaew et al., 2008a] even so only two types of pheromones are used. The MPACA uses a higher number of pheromones.

As reviewed in section (2.6.4.2) there are few multi-colony algorithms which in fact use different pheromone trails concurrently. Many algorithms mention the term multi-pheromone as part of their operation, but this tends to be misleading. In fact most multi-colony algorithms are only separate instances of the same ant colony running in sequence or in parallel, having their results synchronised or merged at some point. In most cases they do not apply any multiple pheromones, but only have a single pheromone applied to each different colony, and deploy multiple colonies in an attempt to solve the same problem. Therefore, having a collection of different pheromones interacting within the same colony is a novelty to the MPACA.

The MPACA uses a decentralised population based learning mechanism which allows ants to self-determine, that is in isolation at the ant level, into which colony they should belong to. This mechanism does share resemblance with the ANTCLUST algorithm, section (2.6.3, [Labroche, 2003]), where ants auto-organise into colonies and determine colony membership through the shared colonial odour. Despite this, the MPACA uses a more elaborate indirect population

based system. The underlying structure is still pheromone driven, however the ants determine colony membership depending on frequency of encounters with other colonies. Hence, ants are attracted towards the same areas based on pheromone scents, but they do not need to share this scent to be part of the same colony.

As reviewed in sections (2.6.4.4, 2.6.3), other multi-colony algorithms exist which use multiple features, such as ODUEC [Bertelle et al., 2006], [Cardon et al., 2006] and the Colour Ant System algorithm (CAS) [Bertelle et al., 2003]. These algorithms make use of competing colonies of ants to determine node colonisation (ODUEC). This mechanism is in effect also a truly multi-colony and multi-pheromone approach, however the underlying structure is still divergent from the MPACA, since the former still uses a single scent for each colony. In the MPACA each colony has multiple ants, each reacting to different pheromones and depositing different pheromone combinations, a process which is restricted only by the feature combinations that occur at the ant level, thus being a multi-colony and multi-pheromone algorithm. The MPACA uses colonies to form a dynamic structure, which learns the distribution of the cluster over graph space. The colony is not fixed, and adjusts itself during the execution of the algorithm. Thus, unlike hierarchical clustering algorithms, and even ant algorithms which learn fixed structures such as the AntTree algorithm [section (2.6.1.1)], the colony values keep adjusting.

3.11 Chapter Conclusion

This chapter has introduced the MPACA, highlighting various aspects behind the way it operations. It has provided a step-by-step breakdown of the algorithm and an internal evaluation of the interacting parameters. The next step is to provide evidence on how this algorithm operates on synthetic and real-world datasets and a discussion around the results attained.

Chapter 4

The MPACA Applied

4.1 Chapter Overview

This chapter presents the results obtained by the MPACA over a number of datasets subdivided into three main categories; (i) synthetic datasets, (ii) real-world UCI datasets, (iii) real-world GRiST and ADVANCE datasets. The chapter uses the first two dataset groupings for benchmarking the MPACA against other results presented in the literature. This is followed by a comprehensive sensitivity analysis of each parameter on a subset of the presented datasets, to understand better its role in the MPACA optimisation.

The GRiST and ADVANCE domains are chosen to demonstrate the applicability of the MPACA and its extended classification counterpart to more complex and messy real-world domains. These include a mental-health risk assessment dataset and a hub-and-spoke logistics dataset for predicting daily demand on resources. These domains present extremely high dimensions (over 200 for the mental-health domain) and extremely high numbers of cases (many millions for the logistics domain), presenting serious challenges for tractability. In each of these two domains, data from human expertise is used [Buckingham et al., 2012], [Buckingham and Adams, 2011], [Buckingham et al., 2008].

4.2 Evaluation Criteria and Experiment Set-up

External evaluation metrics discussed in section (2.2.3) are used to benchmark results attained by the MPACA against other algorithms presented in the literature review. These are chosen over internal evaluation metrics, since the latter are too heavily influenced by normalisation processes and cannot be adequately used for benchmarking purposes.

4.2.1 Synthetic Datasets: the 2D-4C and 10D-10C Datasets

Synthetic datasets are useful because they have a completely specified distribution and thus known properties. These help analyse the contributions of each model element and parameter as well as enabling comparisons between models. The data are produced by applying the xD-yC pattern. In this pattern, x describes the dimensionality (D) which is $\{2 \text{ or } 10\}$ and y denotes the number of Gaussian clusters (C), which is either $\{4 \text{ or } 10\}$. A set of y, x-dimensional normal distributions $N(\vec{\mu}, \vec{\sigma})$ are generated for each data element present in cluster y. The sample size, s, of each normal distribution, the mean vector, $\vec{\mu}$, and the vector of the standard deviation, $\vec{\sigma}$, are randomly determined by using uniform distributions over fixed ranges, $\mu_i \in [0, 100]$ and $\sigma_i \in [0, 5]$ [Handl et al., 2003b].

4.2.2 Real-world UCI datasets

The MPACA is applied to a number of standard datasets found in literature, originating from the UCI data repository [Bache and Lichman, 2013], namely; the Iris, Wine, Soya-bean, Wisconsin breast cancer (WBC), Pima Indians diabetes (Pima) and the Yeast datasets. The number of data elements, properties and class distributions for these datasets are given in table (4.1).

4.2.2.1 Iris dataset

The Iris dataset is probably the best known in pattern recognition literature. It is based on Fisher's classification of the Iris plant, according to four dimensional attributes; sepal length, sepal width, petal length and petal width (in cm). Three possible classes exist; "Iris-setosa", "Iris-versicolour", and "Iris-virginica". The Iris dataset contains 50 instances per class. The "Iris-setosa" class is linearly separable from the other two, whereas the "Iris-versicolour" and "Iris-virginica" are non-linearly separable.

4.2.2.2 Wine dataset

The Wine dataset describes chemical analysis of wines grown in the same region in Italy, but derived from three different cultivars. This dataset contains 13 ordinal attributes, and has 178 instances, which can pertain to three unevenly distributed categories. The categorisation of each class is as follows; class 1 with 59 elements, class 2 with 71 elements, and class 3 with 48 elements.

4.2.2.3 Soya-bean dataset

The Soya-bean dataset is used to diagnose experimental comparisons between various beans from around the world to determine disease diagnosis. This dataset contains 35 ordinal attributes, and has 47 instances, which can pertain to four unevenly distributed categories. The categorisation of each class is as follows; class 1 with 10 elements, class 2 with 10 elements, class 3 with 10 elements, and class 4 with 17 elements.

4.2.2.4 Wisconsin Breast Cancer dataset

The reduced Wisconsin Breast Cancer (WBC) dataset describes characteristics of the cell nuclei present in images, which are used for cancer prognosis. This dataset contains 10 ordinal attributes, and has 569 instances, which can pertain to two unevenly distributed categories, with instances distributed as 357 benign and 212 malignant.

4.2.2.5 Pima Indians Diabetes dataset

The Pima Indians diabetes dataset describes females that are at least 21 years old of Pima Indian heritage, a particular ethnic group prone to diabetes. This dataset is often used to highlight the link between obesity and the development of diabetes. It contains eight attributes, and 768 instances, which can pertain to two categories, either those affected by diabetes (268 instances) and those who are not (500 instances).

4.2.2.6 Yeast dataset

This database contains information about a set of yeast cells. The task is to determine the localization site of each cell. This dataset contains eight ordinal attributes, and has 1484 instances, which can pertain to ten unevenly distributed categories. The categorisation of each class is as follows; CYT (cytosolic or cytoskeletal) with 463 elements, NUC (nuclear) with 429 elements, MIT (mitochondrial) with 244 elements, ME3 (membrane protein, no N-terminal signal) with 163 elements, ME2 (membrane protein, uncleaved signal) with 51 elements, ME1 (membrane protein, cleaved signal) with 44 elements, EXC (extracellular) with 37 elements, VAC (vacuolar) with 30 elements, POX (peroxisomal) with 20 elements, and ERL (endoplasmic reticulum lumen) with 5 elements.

Name	C	S	N_i	D	Type
Square 1	2	100	50, 50	2	Continuous
2D-4C	2	400	100, 100, 100, 100	4	Continuous
10D-10C	2	500	50, 50, 50, 50, 50, 50, 50, 50, 50, 50	10	Continuous
Iris	3	150	50, 50, 50	4	Continuous
Wine	3	178	59, 71, 48	13	Continuous
Soya-Beans	4	47	10, 10, 10, 17	35	Discrete
WBC	2	699	458, 241	10	Discrete
Pima	2	768	500, 268	8	Continuous
Yeast	10	1484	463, 429, 244, 163, 51, 44, 37, 30, 20, 5	8	Continuous

TABLE 4.1: Summary of the selected datasets, where D is the dimensionality, C gives the number of clusters and, S gives the sample size or the number of data elements present, N_i gives the number of data elements for each cluster i, whilst the type can either be discrete or continuous.

4.3 Experimentation Framework

4.3.1 Basic Set-up

A specific test-controlling mechanism has been implemented allowing bulk parameter testing. This splits the MPACA into a core operator and a separate controlling mechanism. The controller instructs the core module which datasets and which parameter combinations are to be executed. It operates on a file driven mechanism, accepting as input an XML file with a set of parameters, and a comma delimited file from which the dataset is loaded. The output of the core module is a number of XML files, which store the results attained for each execution broken down by time.

4.3.2 Analysis based on a Simulated Annealing Technique

Parameter configurations are analysed using a mechanism inspired by simulated annealing (SA), typically used in parameter evaluation settings [Kirkpatrick et al., 1983]. SA originates from thermodynamics and the analogy of annealing in solids. SA uses a temperature parameter controlling the search. Temperature starts off high and is slowly lowered in every iteration. If the new configuration returns a better overall evaluation value it replaces the current parameter and the iteration counter is incremented. It is possible to move towards a worse point, and the probability of doing so is directly dependent on the temperature parameter. This unintuitive step sometime helps identify a new search region in hope of finding a better minimum.

The mechanism used in this evaluation is loosely inspired by such an approach facilitating and directing the search through the huge number of possible parameter combinations. The process

operates by executing the MPACA for a chosen set of parameters and storing the corresponding results. At each step, only one parameter is adjusted, in an attempt to seek a better result. If the parameter change results in an improved result, the algorithm investigates a further change increase in that direction (increase/decrease). The change on each parameter values have varying intervals between parameters, but are fixed within the same parameter, for example the time-window moves from 25 onto 30 and onto 35 (i.e. intervals of 5), whilst the edge length is extended by intervals of one unit. When a change in this parameter value no longer provides an improved result, the operation is repeated on the next parameter. This continues until all parameters have been processed. Once this is so, the system resumes a parameter by parameter analysis until results attained do not improve further. Multiple random starting points are chosen to vary the search as much as possible.

The mechanism used is said to be loosely inspired from SA as there are two limitations in the approach. Firstly, the variation between parameter adjustments is linear, and this variation is set to a fixed value applicable per parameter (as per above 5 for time, 1 for edge length). There is no gradual non-linear increase or decrease on the variation used. Secondly, in the chosen method only one parameter value is modified between execution instances. Hence, the measure of combined parameter changes is limited.

4.4 Baseline Experiments

The baseline experiments are executed over varied parameter ranges [section (3.8.2)] with the intention to determine similarities between chosen settings and adaptability over different domains. Tables (4.2, 4.3) represent the results of applying the MPACA for a number of instances over each and every dataset described in sections (4.2.1) and (4.2.2). In these tables “Edge” stands for maximum edge length, “Complement” stands for ant complement, “Detection” stands for detection range, “Ph. Qty.” stands for pheromone quantity deposited, “Coefficient” stands for maximum pheromone coefficient, “Tolerance” stands for minimum pheromone tolerance, “Evaporation” stands for evaporation rate (%), “Residual” stands for residual value, “Feature” stands for feature merging, “Colony” stands for colony merging and “SD” is the standard deviation.

Not all parameters are adjusted in these baseline experiments, with two parameters being excluded. These are the step size and the minimum pheromone tolerance parameters. These can effectively be considered as hidden parameters within the model. This builds on the discussion presented in section (3.3.2) where the derivation of possible values for the step size parameter

	Square1		2D-4C		10D-10C		Iris		Wine	
Parameters	**mean**	**SD**	**mean**	**SD**	**mean**	**SD**	**mean**	**SD**	**mean**	**SD**
Edge	8	1.4	7.36	2.24	9.31	1.65	8.56	0.68	8.52	1.60
Step-size	0.1	0	0.1	0	0.1	0	0.1	0	0.1	0
Complement	5.09	4.5	1.51	1.11	1.74	1.56	1.32	1.73	1.36	0.48
Detection	1.5	0.5	1.74	0.44	1.35	0.48	1.69	1.22	1.6	1.13
Ph. Qty.	175	56	205	56	149	54	253	130	173	79
Coefficient	1.49	0.5	1.36	0.48	1	0	1.67	0.54	1.49	0.87
Tolerance	1	0	1	0	1	0	1	0	1	0
Evaporation	0.06	0.04	0.04	0.05	0.05	0.04	0.07	0.04	0.07	0.04
Residual	1.01	0.82	1.66	0.53	1.39	0.78	1.6	2.05	1.68	0.51
Feature	4.51	1.5	3.77	0.59	4.57	1.5	4.39	1.54	4.46	1.36
Colony	4.55	1.5	3.73	0.64	4.81	1.47	3.95	1.11	4.36	1.44
Visibility	3.5	0.5	3.82	0.60	3.34	0.47	3.15	1.53	3.04	0.75
Time-window	74	25	80	57	87	78	73	49	58	18
	Square1		2D-4C		10D-10C		Iris		Wine	
Results	**mean**	**SD**	**mean**	**SD**	**mean**	**SD**	**mean**	**SD**	**mean**	**SD**
F-Measure	0.96	0.01	0.95	0.03	0.89	0.02	0.83	0.05	0.86	0.05
Rand-Index	0.98	0.01	0.98	0.02	0.98	0.01	0.89	0.03	0.91	0.03
Precision	0.98	0.01	0.98	0.02	0.98	0.01	0.89	0.03	0.91	0.03
Recall	0.94	0.02	0.93	0.05	0.82	0.03	0.79	0.05	0.83	0.06
Jaccard	0.92	0.02	0.91	0.06	0.80	0.03	0.71	0.07	0.77	0.08

TABLE 4.2: Baseline experimentation applied to the following datasets; Square1 executed for 1,000 instances, 2D-4C executed for 175 instances, 10D-10C executed for 50 instances, Iris executed for 2,000 instances and the Wine executed for 1,500 instances. Results are representative of the average best-fits.

are discussed, with these being either using a step size which is 0.1 or 0.4 of a SD. The preliminary investigation in section (3.8.3.2) has determined that a step size value of 0.4 distorts the granularity required for proper parameter interactions. This parameter also influences the search granularity, which determines what feature values ants react to. For reasons related to this experimentation this value has been fixed.

The other fixed parameter is the minimum amount of pheromone that can be present on each edge. The evaporation rate is applied as a percentage reduction. Hence, irrespective of how much time has passed, there will always be a minimal trace value present in the system. At a point, such a value is negligible for operating purposes and needs to be removed. It has been decided that pheromone traces which are lower than one integral unit are removed.

The baseline experiments attempt to determine the ideal optimal parameter value ranges for each parameter, and how these change for various domains. Given the substantial number of parameters existing in the MPACA, having various "random" starting positions for each parameter lengthens analysis time. Thus, theoretical starting values are chosen depending on the knowledge of the domains being tested, augmented with information from previous experimentation,

	Soya-bean		WBC		Pima		Yeast	
Parameters	**mean**	**SD**	**mean**	**SD**	**mean**	**SD**	**mean**	**SD**
Edge	7.78	1.34	6.98	0.15	6.29	0.72	6.44	0.5
Step-size	0.1	0	0.1	0.03	0.1	0	0.1	0
Complement	3.27	0.86	1.03	0.32	1.09	0.61	1.10	0.5
Range	1.71	0.62	2	0.08	1.22	0.80	1.5	0.5
Ph. Qty.	285	102	145	37	379	125	173	55
Coefficient	1.11	0.32	1.49	0.5	1.44	0.50	1	0
Tolerance	1	0	1	0	1	0	1	0
Evaporation	0.1	0.05	0.01	0.02	0.05	0.04	0.06	0.04
Residual	1.37	0.66	1.48	0.51	1.07	0.82	0.98	0.84
Feature	4.2	1.32	3.97	0.84	4.81	1.66	3.51	0.5
Colony	4.68	1.42	4	0.85	4.75	1.58	3.41	0.49
Visibility	3.12	1.84	3	0.82	2.97	1.51	3.52	0.5
Time-window	23	4	64	24	52	11	56	25
	Soya-bean		WBC		Pima		Yeast	
Results	**mean**	**SD**	**mean**	**SD**	**mean**	**SD**	**mean**	**SD**
F-Measure	0.88	0.14	0.94	0.03	0.61	0.22	0.11	0.07
Rand-Index	0.94	0.07	0.94	0.03	0.62	0.22	0.49	0.05
Precision	0.96	0.05	0.94	0.03	0.63	0.22	0.28	0.10
Recall	0.85	0.17	0.94	0.04	0.63	0.22	0.09	0.07
Jaccard	0.82	0.19	0.88	0.06	0.47	0.18	0.07	0.04

TABLE 4.3: Baseline experimentation applied to the following datasets; Soya-bean executed for 100 instances, WBC executed for 1,200 instances, Pima executed for 175 and the Yeast executed for 120 instances. Results are representative of the average best-fits.

as presented in chapter (3).

The testing rationale behind each parameter is as follows. The maximum edge length parameter determines node connectivity. As discussed in section (3.8.3.1), the normalisation process [section (3.3.2)] can still result in the needless connection of distant nodes and potentially induce a combinatorial explosion of outgoing edges from each node. Sufficient node connectivity needs to be allowed in order to allow ants to move around within the system. The smaller the average connectivity, the more likely it is that these nodes are completely isolated. Initial empirical experimentation applied over the Iris dataset demonstrates when selecting maximum edge length of $\{1, 2, 3, 4, 5\}$ respectively returns an average node connectivity of $\{2, 5, 11, 18, 27\}$ Thus, the smaller edge lengths were excluded, and the value five has been selected as a minimum value. This since values lower than five cause too many nodes to be detached from the graph inhibiting complete cluster formation.

Building on results presented in section (3.8.4.1), the ant complement determines the computational intensity of the algorithm. In order to minimise computation complexity the starting value is always selected as the minimum value possible, which is one. Eventually, this is increased to determine how this accelerates cluster formation. Again using results from section (3.8.4.2), the

detection range parameter determines the granularity of the search. Similarly, in order to maximise the discriminability of the detectors the minimum value for the detection range is selected as a starting position, which has value one.

As discussed in section (3.8.5.1), the pheromone quantity deposited and the evaporation rate are effectively one conjunct parameter. An initial deposition value is required that connects two nodes in a way that the pheromone scent is strong enough to survive the impact of evaporation for a predetermined amount of time. Functionally, this is determined by the time needed for the ant to traverse the longest edge and still have some pheromone on the first step. Given that the average edge lengths used are of approximately 8 steps, this means an ant takes 8 time cycles to traverse an edge. Thus, the pheromone on the edge must survive for 8 time cycles. If the amount deposited is 25 units, subject to an evaporation rate of 10%, after 10 steps it has nearly completely evaporated. Given this is longer than the average length, it seems a reasonable starting position to have a deposit of 25 units and evaporation rate of 10%. Pheromone evaporation only requires a small enough value to commence with which can reduce the importance of the pheromone trace, but not too quickly.

The maximum pheromone coefficient [section (3.5.1.2)] needs to be a value which limits stagnation from occurring. The base value for the maximum pheromone coefficient is set to the floor value of one. This value is increased accordingly to determine the required capping value.

Setting the residual amount parameter to zero stops ants from exploring uncharted territory. As demonstrated in section (3.8.5.2), small residual amounts produce the best results. The more it is increased, the less is the influence of the positive feedback loop. The starting point is chosen to be zero to determine the effect of no parameter, and then small increases are allowed because we know that it reduces performance when it becomes too large.

Section (3.8.6.2) showed that a low setting of the colony-merging threshold causes over-rapid colony formation and stops accurate clusters. A value of one means ants immediately join up on meeting, which is clearly too low. So the minimum operating value was set to two, for both colony and feature-merging thresholds.

The visibility parameter determines when ants are near enough to be considered an encounter. Initial experimentation during development stages showed that counting all ants on an edge as being encountered precipitated feature and colony merging too quickly. In contrast, counting only ants present on the current node slowed colony formation, and thus returned lower quality results. Hence the starting position for the parameter is all ants within half the average edge length for the domain, which means all ants within about four steps of the node.

The time-window parameter is dependent on the domain size. Having a large dataset with high dimensionality means more ants encounter each other, and all activities which revolve around ant encounters accelerate. The aim is to obtain a representation of ants within the same vicinity so that frequent colonies or features can be merged. A sensible minimum definition for a vicinity is three nodes and an ant would need to travel 24 steps on average for domains with an average edge length of 8. Hence the minimum time window was set to 25 steps or time cycles.

The chosen start values and intervals for parameters have now been established. The next step is to apply these settings to the selected datasets.

4.4.1 Observations from Baseline Experiments

A number of interesting observations can be derived from the results displayed in tables (4.2, 4.3). Larger datasets favour smaller edge values, as is the case with the Pima and Yeast datasets. All other smaller datasets return better results at higher edge lengths.

The ant complement in datasets with the largest data sample as in the WBC, Pima, and Yeast datasets, indicates that a complement equal to the base value of one returns the best results. The ant complement is only slightly higher in other datasets, and in most cases it is lower than an ant complement of two. Thus, in many cases the MPACA can safely operate within the range of between one and two ants per feature per node. The Square1 and the Soya-bean datasets defy this trend because the former has a small dimensionality and the latter has a low number of data elements.

Smaller datasets also tend to require more pheromone. This is likely to be due to the reduced ant counts within the system. Hence, as evaporation takes its toll on the entire system, ants can fail to keep up with standard pheromone deposition values and require an increase to the quantity deposited in order to improve result quality.

Pheromone values average vary between 5% and 10% in all databases analysed, which indicates that higher evaporation values are not recommended. The average maximum pheromone coefficient for all experiments never exceeds value two and underlines the importance of a ceiling amount. It also shows that this maximum amount can be equivalent to the pheromone quantity being deposited into the system. The residual value is necessary but only seems to require a small amount to have a positive impact. This indicates that some randomness is needed but not to the extent that it impedes stigmergy.

Feature merging and colony counts average out at around a value of four. Colony merge thresholds are slightly lower than feature merging thresholds, which suggests that joining a colony is

Algorithm	Square1		2D-4C		10D-10C	
	F-measure	**Rand-index**	**F-measure**	**Rand-index**	**F measure**	**Rand-index**
Average-link	0.98[1]	0.98[1]	1.00[1]	1.00[1]	1.00[1]	1.00[1]
K-Means	0.99[1]	0.99[1]	0.97[1]	0.98[1]	0.97[1]	0.99[1]
EM-Clustering	0.99[2]		0.99[2]		0.91[2]	
PSO/PSO-K-means			0.96[14]/0.98[8]		0.97[8]	
SACA	0.98[1]	0.98[1]	0.99[1]	0.99[1]	1.00[1]	1.00[1]
ATTA	0.98[15]		0.99[15]		1.00[15]	
	F-measure	**Rand**	**F-measure**	**Rand**	**F-measure**	**Rand**
MPACA	**0.96**	**0.98**	**0.95**	**0.98**	**0.89**	**0.98**

TABLE 4.4: The MPACA performance applied over synthetic datasets and how this compares to the algorithms reviewed in chapter (2). Columns respectively represent the F-Measure and Rand Index (accuracy).

slightly more important than merging features.

The visibility across domains fluctuates between three and five. In reality, this tends to be roughly half the average length (in steps) for each domain, which is close to the starting value.

A pattern emerges on observing time-windows. Higher dimensionality domains have a shorter average time-window, as is the case with the Wine, Soya-bean, WBC, Pima and Yeast datasets. This pattern excludes the 10D-10C dataset, because of the low number of instances (50) for which this dataset has been executed. Smaller time-windows can be used for larger dimensionality problems even if there are not many instances because the ant complement increases with dimensionality as well.

4.4.2 Evaluating the MPACA Clustering Performance

Results presented in tables (4.4, 4.5) demonstrate that the MPACA performance is equitable with alternative approaches, including both nature-inspired algorithms and more classical clustering approaches. An important point is that these datasets are not necessarily designed for unsupervised clustering methods, and are often used as benchmarks for supervised techniques. This puts the MPACA at a disadvantage and its performance is therefore better than it appears in the tables, in comparison to the supervised methods. In the MPACA, clusters are mapped to classes at the evaluation level and not able to use the known class membership as part of its training information. This does not influence the operation of the algorithm in any way.

Unfortunately, not all algorithms presented have corresponding results published applied to the datasets on which the MPACA has been applied to. In many cases the algorithms do not have enough detail that allows them to be reverse engineered, and contact with their authors did not

Algorithm	Iris		Wine		Soya-bean	
	F-measure	**Rand-index**	**F-measure**	**Rand-index**	**F measure**	**Rand-index**
APC	0.94[4]	0.94[4]				
Ant-Miner		0.95[5]		0.95[5]		0.96[19]
KNN		0.91[5]/0.95[22]		0.96[16]/0.96[22]		1.00[16]
ABC		0.96[25]				0.98[25]
SACA	0.82[1]	0.83[1]	0.86[3]	0.83[3]		
ATTA	0.82[15]		0.88[15]			
AntClass		0.85[9]/0.79[20]		0.94[9]		0.97[9]
AntTree		0.82[20]		0.82[20]		0.88[20]
ANTCLUST		0.78[21]				0.93[21]
ACO	0.78[10]		0.52[10]			
PSO	0.78[10]		0.52[10]/0.69[14]			0.93[17]
BCO	0.82[11]	0.83[11]				
Average-link	0.81[1]	0.82[1]	0.84[12]	0.81[12]		
K-Means	0.83[1]	0.82[1]	0.93[3]/0.82[12]	0.90[3]/0.82[12]		0.92[16]
DBSCAN		0.76[6]		0.73[6]		
EM-Clustering	0.70[2]		0.85[2]			1.00[16]
	F-measure	**Rand**	**F-measure**	**Rand**	**F-measure**	**Rand**
MPACA	**0.83**	**0.89**	**0.86**	**0.91**	**0.88**	**0.94**

Algorithm	WBC		Pima		Yeast	
	F measure	**Rand index**	**F measure**	**Rand index**	**F measure**	**Rand index**
APC	0.96[4]	0.97[4]	0.65[4]	0.70[4]		
Ant-Miner+		0.96[5]		0.72[13]		0.43[13]
KNN		0.96[5]		0.68[22]		
ABC		0.96[25]				
SACA	0.97[1]	0.94[1]	0.47[3]	0.50[3]	0.44[1]	0.68[1]
ATTA	0.97[15]				0.44[15]	
AntClass		0.97[9]		0.53[20]		
AntTree		0.79[24]		0.50[20]		
ANTCLUST				0.55[21]		
ACO	0.82[10]					
PSO	0.82[10]					
BCO					0.50[11]	0.82[11]
Average-link	0.97[1]	0.93[1]			0.45[1]	0.74[1]
K-Means	0.97[1]	0.93[1]	0.68[3]	0.69[3]	0.43[1]	0.75[1]
DBSCAN		0.63[23]		0.54[23]		0.64[7]
EM-Clustering	0.68[2]		0.69	0.70[18]	0.43[2]	0.51[18]
	F-measure	**Rand**	**F-measure**	**Rand**	**F-measure**	**Rand**
MPACA	**0.94**	**0.94**	**0.61**	**0.62**	**0.11**	**0.49**

TABLE 4.5: The MPACA performance applied over real-world datasets and how this compares to a subsection of the algorithms reviewed in chapter (2). Columns respectively represent the F-Measure and Rand Index (accuracy).

resolve the issue. References used in tables (4.4, 4.5) are as follows, (1) = [Handl et al., 2003a], (2) = [Tan et al., 2011], (3) = [Boryczka, 2010], (4) = [Halder et al., 2008], (5) = [Martens et al., 2007], (6) = [Xiong et al., 2012], (7) = [Chaimontree et al., 2010], (8) = [Breaban and Luchian, 2011], (9) = [Monmarché et al., 1999a], (10) = [Niknam and Amiri, 2010], (11) = [Santos and Bazzan, 2009], (12) = [Chandrasekar and Srinivasan, 2007], (13) = [Cano et al., 2013], (14) = [Wan et al., 2012], (15) = [Tan et al., 2006], (16) = [Bougenière et al., 2009], (17) = [Wang et al., 2007], (18) = [Jebara, 2002], (19) = [Rami and Panchal, 2012], (20) = [Azzag et al., 2007], (21) = [Labroche et al., 2002a], (22) = [Guo et al., 2003], (23) = [Yang and Zhang, 2007], (24) = [Ingaramo et al., 2005], (25) = [Shukran et al., 2011].

It is necessary to distinguish classification algorithms from clustering algorithms. For reasons previously outlined, classifiers have an obvious advantage over clustering approaches. Classifiers such as the APC, the Ant-Miner, or the KNN (as expressed in result publications) produce results which are superior to the MPACA, on all the presented datasets. This may be due to them being able to use known class membership in their training data, but also because the MPACA has a relatively crude method of mapping colonies to classes. A better way of using the MPACA results to generate class memberships would improve its evaluation without actually changing its performance.

Despite these caveats over the interpretation of the MPACA results, it still performs at a level close to or better than the other algorithms. An interesting comparison is with the SACA. To better understand why SACA returns better results on, for example, the yeast data set, one must return to the critique mentioned in chapter (2). This explained that the SACA uses a two-stepped approach. Objects are first re-positioned in space and then subsequently parsed by some other clustering tool. Thus, the SACA process of re-arranging objects is difficult to gauge because it is not a complete system in itself. The ATTA, another SACA-type algorithm, performs in much the same way. The MPACA returns comparative results on the Square1, Iris, Wine and WBC datasets, whilst being consistently inferior on the Yeast dataset.

A further improvement to the SACA is the AntClass algorithm, which includes a hybridisation of the K-Means within it. In fact, when applied to the Wine, Soya-bean and WBC, this algorithm returns results which are slightly superior to those attained by the MPACA. Therefore, the hybridisation of the K-Means algorithm provides the SACA approach core, a substantial boost. It is possible that using a better interpretation of the MPACA colonies with K-Means may likewise improve its results.

Investigating further the results attained by other clustering types, discussed earlier in chapter (2), a consistent pattern emerges. Once more the MPACA returns superior results over the

ANTCLUST for the Iris, Soya-bean and Pima datasets. Thus, even for the typology that the ANTCLUST represents, which differs from that of the MPACA, results still favour the MPACA approach. Furthermore, the MPACA returns superior results over the AntTree on all datasets mentioned.

Results for ACO as applied to clustering, despite being limited, also demonstrate the continued result trend, with the MPACA being superior on all datasets presented. The MPACA is superior to both clustering implementations of PSO and BCO, again on all datasets presented. As a rule, ant based clustering, be it the MPACA or otherwise, are demonstrated to be better suited to tackling the clustering problem than PSO or BCO. Although the MPACA is inferior to the ABC algorithm, the ABC has an advantage by being used as a classifier.

Mixed results are attained when comparing the MPACA against both hierarchical (average-link) and centroid based (K-Means) clustering approaches, with both algorithms outperforming it on most synthetic datasets. The MPACA outperforms them both on the Iris dataset, and again outperforms the average-link on the Wine dataset. Conversely, the average-link outperforms the MPACA over WBC and Yeast, whilst for its part K-Means outperforms it on Pima and Yeast. The simplicity of these mechanisms, and the relative compactness of the domains being investigated might give both of these approaches an advantage.

Excluding the clear vulnerability that the MPACA has over the Yeast dataset, so far the MPACA has shown to be on a par with most clustering algorithms, ant based or otherwise. This becomes interesting when considering more elaborate clustering mechanism, such as the Density based (DBSCAN) and probabilistic methods (EM-Clustering). The MPACA returns significantly superior results over the DBSCAN on the Iris, Wine, WBC and Pima datasets, and is only inferior on the Yeast dataset. When compared to the EM-Clustering, the MPACA is on a par for the synthetic datasets, but by far superior on the Iris, Wine and WBC. EM-Clustering is better at handling Pima, Soya-bean, and the Yeast dataset.

In general, then, it is possible to affirm that the MPACA returns favourable results when compared to other clustering algorithms. However, it signally fails to return statistically adequate results on the yeast dataset. One key reason is that the yeast dataset has uneven clusters, with the top two clusters having more elements than the other eight clusters combined. This imbalance and the lack of data elements inhibit the critical mass required for the MPACA to form correct clusters. Increasing the ant complement can improve the results, but can also cause further unwanted sub-clusters to form. More work is needed to determine how the MPACA can learn clusters that are grossly unbalanced. It may be that a supervised version is the best way to achieve this, because then the clusters are known and the MPACA can learn the distinctions

between them using known class memberships rather than guessing them. Potential ways in which this limitation can be overcome are later produced in chapter (5).

4.5 Sensitivity Analysis of chosen Parameters

4.5.1 Sensitivity Metric

In order to better evaluate the results attained, a Sensitivity Index (SI) is used as an operator to determine the sensitivity of each and every parameter The Sensitivity Index introduced by Hoffman and Gardner calculates the output difference when varying an input parameter from its minimum value to its maximum value [Hoffman and Gardner, 1983]. The SI is thus calculated by averaging the result difference between the minimum and maximum values attained from the optimal parameter value, as per table (4.6). SI is calculated as per equation (4.1):

$$SI = \frac{|Best - Low| + |Best - High|}{2} \tag{4.1}$$

The higher the SI value is the bigger the impact the parameter has on the overall model. The experiments which now follow are the combination of results attained from these baseline experiments augmented with results from 50 execution runs for each tested parameter.

Important issues for clustering algorithms are the number of parameters and how to find the best fitting values for different data sets. Too many parameters create an unnecessarily large search space for optimal settings as well as leading to over-fitting of training data. It is therefore useful to know how important the parameters are for the MPACA and how to limit the search space when applying it to different datasets. Tables (4.2) and (4.3) show that the best-fitting values for each parameter across a wide variety of datasets have low standard deviations, suggesting that they impact on results consistently across several runs. Furthermore, the ranges are comparable across the datasets, which means the starting points in the parameter search space can be confidently assumed to be in the region of the eventual optimal settings, which helps ensure they can be obtained accurately and within an acceptable time-frame.

Table (4.6) explores the problem in a little more detail, using two of the datasets: the Iris and Wine datasets with dimensionality of four and thirteen respectively. As the earlier analysis showed, many of the parameters are very tightly constrained, such as the edge length and step size. In fact, table (4.6) shows that most of them have optimal settings across all datasets that lie within a narrow range, making it easy to initialise them and learn the best fits. This includes those that have a high impact on performance, as indicated by the sensitivity values. This is

Parameters Iris dataset	Best value	Low (F-Measure)	High (F-Measure)	Sensitivity Index (%)
Edge	8 (0.84)	5 (0.42)	14 (0.77)	0.25
Step-size	0.1 (0.83)	0.1 (0.83)	0.4 (0.53)	0.15
Complement	2 (0.83)	1 (0.83)	5 (0.76)	0.04
Range	2 (0.83)	1 (0.83)	5 (0.70)	0.07
Ph. Qty.	250 (0.84)	25 (0.74)	500 (0.55)	0.20
Evaporation	0.07 (0.83)	0.01 (0.69)	0.15 (0.71)	0.13
Coefficient	2 (0.84)	1 (0.83)	5 (0.62)	0.12
Residual	2 (0.84)	0 (0.76)	10 (0.66)	0.13
Feature	4 (0.84)	2 (0.71)	6 (0.71)	0.13
Colony	4 (0.84)	2 (0.66)	6 (0.68)	0.17
Visibility	4 (0.84)	1 (0.59)	8 (0.68)	0.21
Time-window	75 (0.83)	25 (0.43)	500 (0.59)	0.32

Parameters Wine dataset	Best value	Low (F-Measure)	High (F-Measure)	Sensitivity Index (%)
Edge	8 (0.86)	5 (0.66)	11 (0.67)	0.20
Step-size	0.1 (0.86)	0.1 (0.86)	0.4 (0.50)	0.18
Complement	2 (0.87)	1 (0.86)	4 (0.84)	0.02
Range	2 (0.87)	1 (0.86)	5 (0.64)	0.12
Ph. Qty.	175 (0.86)	25 (0.65)	350 (0.63)	0.22
Evaporation	0.07 (0.86)	0.01 (0.66)	0.15 (0.64)	0.21
Coefficient	2 (0.86)	1 (0.84)	5 (0.60)	0.14
Residual	2 (0.87)	0 (0.79)	10 (0.61)	0.17
Feature	4 (0.86)	2 (0.75)	6 (0.72)	0.13
Colony	4 (0.91)	2 (0.74)	6 (0.71)	0.19
Visibility	4 (0.86)	1 (0.48)	8 (0.56)	0.34
Time-window	60 (0.86)	25 (0.52)	500 (0.69)	0.26

TABLE 4.6: Parameter sensitivity analysis as applied to the Iris and Wine dataset. Columns represent parameter, the performance of its best value, the starting value, the highest value, and the sensitivity measure. The parameter values are accompanied by their F-measure in brackets for the overall model performance. The best performing or optimal values for this parameter are extracted from the baseline experiments presented in table (4.2). Low and high values are executed over a set of fixed parameters for 50 instances each.

helpful because it ensures the parameter search does not have to traverse a large range, which would increase the chances of suboptimal settings. The sensitivity analysis combined with an understanding of the consistency of optimal settings suggests that the number of parameters within the MPACA will not undermine its usefulness for clustering different types of datasets.

4.5.2 Pheromone Driven versus a Random Model

A key question to be asked is; to what extent is pheromone important and is a pheromone driven search more powerful when compared to a random search? Unfortunately, in literature many ant algorithms are usually not compared against a random driven approach. By analysing the

Dataset Wine dataset	Parameter	Value 1 (F-Measure)	Value 2 (F-Measure)	Average (F-Measure)
Iris	Ph. Qty.	0 (0.59)	25 (0.74)	250 (0.84)
Wine	Ph. Qty.	0 (0.41)	25 (0.65)	175 (0.86)

TABLE 4.7: Pheromone Driven versus a Random Model applied to the Iris and Wine dataset. Columns indicate the following representations, the dataset being discussed, the applicable parameter being discussed, value 1 is when pheromone is null, value 2 is when pheromone is set to a higher value, average value represents an average amount of pheromone, both of the above have their respective F-Measure included in brackets. The best performing or optimal values for this parameter are representative of the average best-fits extracted from the baseline experiments presented in table (4.3).

influence of the pheromone factor, it is possible to determine the direct effect, or lack of it, that this has on final clustering results.

When a value zero is considered this indicates purely random ant interaction. This is coupled with an actual minimum value, one which has been derived by theoretical analysis earlier, which established that a minimum quantity to be deposited is 25 units. Empirical analysis presented in table (4.2) shows that the average parameter value for the Iris and Wine datasets is respectively 250, and 175 units each. The optimal amount of pheromone deposited varies between dataset sizes.

Results in table (4.7) demonstrate that when no pheromone is present, cluster quality is very low. As the amount of pheromone used is increased, the cluster quality increases. Even a small amount, in this case 25 units, immediately impacts the clustering process. This demonstrates that clustering results are superior when pheromone is introduced, and proves the importance of the stigmergic effect of pheromone.

4.6 Real-world GRiST and ADVANCE datasets

This section applies the MPACA to two real-world problem domains. They differ from the synthetic and UCI datasets by increased uncertainty in the output decisions and by variability of data quality. One domain is in the field of mental health, which will be discussed first, and the other is a logistics dataset.

4.6.1 GRiST - Mental health risk assessment

The Galatean Risk and Safety Tool, GRiST [Buckingham and Adams, 2013] analyses mental-health risk-assessment data. GRiST assists mental-health practitioners assess patient risks of suicide, self-harm, harm to others, self-neglect, and vulnerability. It is based on the knowledge of multidisciplinary practitioners working in all areas of mental health and was designed to

	Grist Dataset		
Parameters	**start**	**mean**	**SD**
Max. Edge Length	7	8.09	0.39
Step Size	0.1	0.1	0
Ant Complement	1	1.19	0.5
Detection Range	1	1.94	0.24
Ph. Qty. Deposited	100	193	32
Max. Ph. Coefficient	1	1.13	0.34
Min. Tolerance	1	1	0
Evaporation Rate (%)	0.05	0.10	0.03
Residual Value	0	1.86	0.46
Feature Merging	3	4.00	0.39
Colony Merging	3	3.98	0.39
Visibility	3	3.16	0.54
Time-window	50	85	16
Corresponding Results	**best**	**mean**	**SD**
F-Measure	0.90	0.83	0.04
Rand-Index	0.92	0.74	0.09
Precision	0.95	0.74	0.09
Recall	1.00	0.96	0.07
Jaccard	0.83	0.71	0.06

TABLE 4.8: Parameter settings for the MPACA as applied to the GRiST dataset, the start value at the beginning of clustering and the mean and standard deviation (SD). This dataset has been executed for 500 instances.

disseminate their expertise to services where people do not have specialist mental-health training [Chircop and Buckingham, 2013].

In this global GRiST analysis, input patient information potentially consists of 138 individual attributes. These patient vectors are assigned a clinical risk evaluation by the assessor and the database contains more than 50,000 patient records. Unfortunately this data varies considerably in its completeness because of various assessment mechanisms. Hence, clinical judgements are not based on full vectors, and may have less than 50 per cent of the values present. The output risk judgements are along a sliding scale from 0 (minimum risk) to 10 (maximum risk). In effect, this is a sliding scale from 0 to 10 rather than 11 output classes and it is unclear how it maps onto clinical risk classes. In other words, there is no definition of high or low risk patients per se.

As introduced so far, the MPACA does not accommodate missing values so the analysis of GRiST data was conducted on patient vectors for twenty-five most commonly supplied variables. A random sample of 250 patients was obtained from the database and the MPACA was applied to it to find any clusters. The task was to detect patients in the low-risk category, defined

as having a clinical judgement of three or less, and the high-risk category defined by patients with a judgement of seven or more.

Table (4.8) displays the initial parameter values and results attained averaging over 500 execution instances. The results show that the MPACA can correctly learn two clusters, one for representing low clinical risk and the other for high clinical risk categories, with an average precision of 74% and an F-Measure of 83%, and with best values reaching a precision of 95% and an F-Measure of 90%. Although this seems like a sufficiently good result, it was made easier by only trying to detect gross errors where high and low risks are confused. Attempting to predict the exact judgement between 0 and 10 would obviously be harder, but enough encouragement has been given with these initial results to make it worth pursuing. Certainly there is evidence that the MPACA has utility for datasets that have high levels of inherent uncertainty such as GRiST, where the best algorithms such as random-forest classification [Breiman, 2001] had a precision of 87%, albeit for classifying patients within plus or minus one of the actual judgement, which is harder than the MPACA task.

4.6.2 ADVANCE - Hub-and-Spoke Logistics Networks

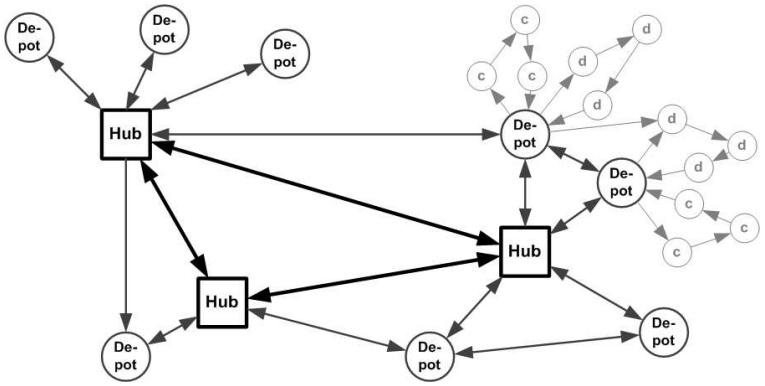

FIGURE 4.1: Transportation in a multiple hub-and-spoke logistics system.

The second real-world dataset is for hub-and-spoke logistics networks. These have a standard modus operand [Zapfel and Wasner, 2002], that consists of a number of haulage depots collecting and delivering shipments to and from one or more central hubs. Figure (4.1) shows a simplified diagram of these activities for a network with 3 hubs and 8 depots. The idea is that a depot takes its own customers' shipments to the hub and brings back shipments from any of the other depots that require delivery to the depot's assigned delivery area.

The problem depots have is with predicting how many shipments will be at the hub by the end of the day for depot to deliver. Knowing the likely number of shipments improves the decision

making process and was a key focus of the EU FP7 ADVANCE project [adv, 2013]. A machine-learning algorithm was used to generate the predictions and the task for the MPACA is to see how well it can use these predictions to detect when demand is likely to be greater or less than expected.

Field work derived from ADVANCE showed that fluctuations in the numbers of shipments (pallets, in this domain) are considerable and have a significant impact on operational performance figure(4.2). If the shipments are more than expected, depots may not take enough lorries to the hub and will have to leave shipments behind, with costly penalties if the network has to deploy alternative resources to deliver them. If shipments are less than expected, depots may take too many lorries and will have wasted space on the return trip. To explore the potential of the MPACA in supporting hub-and-spoke decision making, the first step was to find out how well it could predict whether the demand was above or below the mean and compare this with the machine learning program chosen for ADVANCE [Welch et al., 2012].

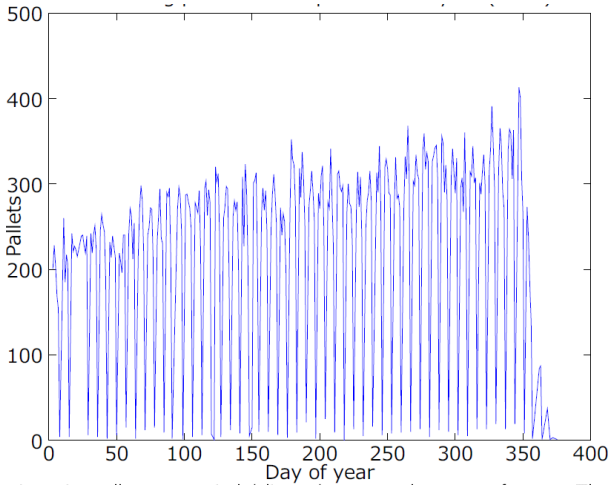

FIGURE 4.2: Fluctuations in the number of pallets each day for a specific depot in the ADVANCE project (the regular very low troughs represent the weekends).

4.6.2.1 Predicting Shipments

The ADVANCE machine learning program for comparision with the MPACA consists of two main processes: select the most appropriate attributes for a depot and then learn the accompanying linear regression model for predicting the number of shipments or total demand at the end of the day [Welch et al., 2012]. The attributes used to predict demand include the known current demand (what has already been committed to the hub) and a number of other variables to do with stages of shipment orders, when they were made, and so on. These numbers obviously change as the day progresses so models were learned for separate time points. In fact, a separate

regression model was learned for each depot at selected times of each day for each day of the week.

The attribute-selection process picked out 15 of the most influential variables from sixty potential ones and these were used to learn the regression model. The same ones, including the known end-of-day demand, were used by the MPACA to set up the hyper-dimensional graph space. Each node (or day in this domain) was assigned to one of two classes: "above" if the known demand was above the mean and "below" otherwise.

At the start of learning, the ants were assigned to the colony matching the class of their starting node. The ants then moved around the graph according to the algorithm described earlier until they had formed population clusters. The method differs from the MPACA's origins in clustering because it exploits known outcomes through supervised learning: the actual number of shipments required for delivery is made part of the hyper-dimensional space for learning and then removed when classifying unknown cases.

Testing was conducted by putting the unknown objects into the hyper-graph but with the known-demand dimension removed. In other words, the outcome information about these unknown objects was not included in the domain. They were assigned to the colony that had the nearest centroid (multidimensional mean), measured as the Euclidean distance from the object to that point. This provided the MPACA with the ability to predict whether the demand was going to be greater than or less than normal for the day, depending on whether it was in the colony for demand above the mean or below the mean.

Four depots were tested at two different times of the day and week: 12.00 and 15.00, on a Wednesday. The mean number of shipments for the depots was around 100 (which equates to between two and three lorry loads). Thirteen separate training and testing cycles were conducted for the MPACA and the results are compared with the machine-learning regression model using precision, which is the percentage of outcomes and predictions agreeing with each other with respect to the total sample size of predictions. The sample for each depot consisted of 206 days and these were randomly divided into two equal sets for training and testing.

Table (4.9) shows the same consistent pattern of optimal parameter values as that for the data used to evaluate the parameters earlier. This also means the settings are very similar across models for different depots and different times of the day. Hence it is feasible to apply the MPACA to all depots, week days, and times of the day, despite this requiring learning about five hundred separate models. Table (4.10) compares the prediction precision of the MPACA with the machine-learning regression program produced by ADVANCE [Welch et al., 2012]. The

	12:00 Depot 2			12:00 Depot 3			12:00 Depot 5		
Parameters	**start**	**mean**	**SD**	**start**	**mean**	**SD**	**start**	**mean**	**SD**
Max. Edge Length	8	8.58	0.49	8	8.55	0.50	8	8.51	0.50
Step Size	0.1	0.1	0	0.1	0.1	0	0.1	0.1	0
Ant Complement	1	1	0	1	1	0	1	1	0
Detection Range	2	2	0	2	2	0	2	2	0
Ph. Qty. Deposited	100	159	50	100	158	50	100	150	50
Max. Ph. Coefficient	1	1.58	0.49	1	1.58	0.50	1	1.47	0.50
Evaporation Rate (%)	0.01	0.03	0.01	0.01	0.03	0.02	0.01	0.03	0.02
Residual Value	2	2	0	2	2	0	2	2	0
Feature Merging	3	3.99	0.82	3	4.01	0.81	3	4	0.82
Visibility	2	2.94	0.84	2	2.96	0.84	2	3.03	0.78
Time-window	50	63	13	50	63	13	50	63	13
	12:00 Depot 7			15:00 Depot 2			15:00 Depot 3		
Parameters	**start**	**mean**	**SD**	**start**	**mean**	**SD**	**start**	**mean**	**SD**
Max. Edge Length	8	8.46	0.50	8	8.59	0.494	8	8.57	0.50
Step Size	0.1	0.1	0	0.1	0.1	0	0.1	0.1	0
Ant Complement	1	1	0	1	1	0	1	1	0
Detection Range	2	2	0	2	2	0	2	2	0
Ph. Qty. Deposited	100	157	50	100	161	50	100	159	50
Max. Ph. Coefficient	1	1.47	0.50	1	1.55	0.5	1	1.59	0.50
Evaporation Rate (%)	0.01	0.03	0.02	0.01	0.03	0.02	0.01	0.03	0.02
Residual Value	2	2	0	2	2	0	2	2	0
Feature Merging	3	4.03	0.82	3	4.04	0.82	3	4	0.82
Visibility	2	2.99	0.79	2	2.97	0.84	2	2.96	0.84
Time-window	50	63	13	50	63	13	50	63	13

	15:00 Depot 5			15:00 Depot 7		
Parameters	**start**	**mean**	**SD**	**start**	**mean**	**SD**
Max. Edge Length	8	8.55	0.5	8	8.51	0.50
Step Size	0.1	0.1	0	0.1	0.1	0
Ant Complement	1	1	0	1	1	0
Detection Range	2	2	0	2	2	0
Ph. Qty. Deposited	100	150	50	100	158	50
Max. Ph. Coefficient	1	1.54	0.50	1	1.45	0.5
Evaporation Rate (%)	0.01	0.02	0.02	0.01	0.02	0.02
Residual Value	2	2	0	2	2	0
Feature Merging	3	4.11	0.85	3	3.97	0.79
Visibility	2	2.97	0.82	2	2.96	0.84
Time-window	50	62	13	50	63	13

TABLE 4.9: Parameter settings for the MPACA. The start value is the one set at the beginning of learning followed by the mean and standard deviation (SD) are the average values as these parameters were varied over instances ranging between 75 and 100 training instances for each individual time-point.

		Precision		
Depot	**Time**	**ML**	**MPACA**	**SD**
2	12:00	0.75	0.76	0.01
3	12:00	0.61	0.84	0.011
5	12:00	0.86	0.92	0.039
7	12:00	0.76	0.85	0.011
2	15:00	0.76	0.76	0.008
3	15:00	0.69	0.91	0.009
5	15:00	0.84	0.98	0.013
7	15:00	0.74	0.93	0.012
MPACA mean		0.75	0.87	

TABLE 4.10: Results for predicting whether demand will be above or below the average for a Wednesday testing for four depots at two times. ML gives the machine-learning regression model prediction and the MPACA precision is its mean for learning and testing cycles utilising the parameters as given in table (4.9). The final standard deviation (SD) column gives the SD of the mean across the cycles.

MPACA compares favourably with it and demonstrates its efficacy for application to real-world data that have been analysed by more traditional methods. The variation for which of the two models is better for a particular depot and time is probably due to using categorical outcomes, where outcome demands only marginally above or below the mean are equally weighted with those having much larger deviations.

4.7 Discussion of Results Attained

The MPACA has been evaluated in detail against numerous datasets and has been shown to perform favourably compared with both ant clustering and traditional clustering methods. It returns results which are comparable and in some cases superior to existing clustering methods. The parameter analyses demonstrated a degree of consistency for optimal settings that make it feasible to search for the in many types of problem domain with datasets varying in sizem dimensionality, and certainty of results. The sensitivity analyses showed that those parameters where the particular settings had impact on performance had optimal values within a constrained range covering all the data sets. Again, this shows that finding the best settings for the MPACA is not searching for a needle in a haystack because it is clear where the search should begin and the answers lie nearby. The MPACA has thus shown considerable promise as a versatile, practical, and high-performance clustering algorithm.

Chapter 5

Conclusion and Future Work

This thesis explored the Multiple Pheromone Ant Colony Algorithm (MPACA) and its application to clustering problems. It builds on the argument introduced in chapter (1), namely the ongoing difficulty that computer scientists face when dealing with big, messy, unstructured data and the quest of how such data can be transformed into useful information. The clustering of unstructured distributed datasets is a problem ideally suited to be tackled by decentralised, bottom-up approaches. Amongst the variety of nature inspired algorithms are swarm techniques, which are based on the idea of emergent properties from decentralised behaviour. However, as is shown in this thesis, not all swarm algorithms conform to the required properties, as some have too much central control. This thesis proposed an alternative algorithm, the MPACA that was governed by the need to remove central control as far as possible.

As has been shown, the three key elements present in the MPACA which make it an ideal swarm based implementation are; its ability to self-organise (section 1.3.1), the use of stigmergy (section 1.3.2) and positive feedback (section 1.3.3). A number of ant based typologies were presented in chapter (2), and, to a degree, each of these offers some similarity to the MPACA, with the type IV typology [section (2.6.4)] bearing the greatest resemblance. Since the MPACA is mainly a clustering algorithm, it is also compared with both traditional clustering algorithms and other ant based clustering mechanisms. Chapter (3) introduces the model details and identifies where the MPACA is distinctive. Chapter (4) then demonstrates that these distinctive properties enable the algorithm to produce useful results for multiple clustering problem domains and that it compares well with alternative models, with some advantages. This final chapter sums up the qualities of the algorithm and considers avenues of exploration that are still required.

Section (5.1) revisits the overall goal and how this has been achieved. It is followed by section (5.2), in which alternative paths to the development of the MPACA are briefly reviewed and then

the final model advances are given in section (5.3). Limitations of the current mechanism and possible proposed improvements are given in section (5.4). Finally, sections (5.5) and (5.6) discuss the future work being implemented in parallel with this thesis write-up and the concluding arguments are presented.

5.1 Summary

The scope of the MPACA was to deliver a unique swarm inspired clustering algorithm, which matches the three criteria outlined in section (1.4). Even if each of these criteria in isolation can be found in one or more typologies outlined in chapter (2), it is the combination of these criteria and the use of individual pheromone for each node feature that makes the MPACA unique.

5.1.1 Unique properties of the MPACA

The MPACA borrows from and builds on similar properties of other ACO algorithms. However, it does so in ways that make them uniquely implemented within the MPACA and the overall system produces clustering behaviour that is demonstrably different.

5.1.1.1 Modified Ant Transition Mechanism

The MPACA uses an adjusted ant movement model, which although similar to typical ACO, is solely pheromone driven. In ACO, ant movement mechanisms involve ants which have to wait for all other ants to perform tours, and out of each and every tour implement a quality analysis to determine which tour has the best quality. The MPACA differs because ant movements do not depend on any other ants and there is no global mechanism for analysing them. In fact, in the MPACA there is no real tour concept; ants are isolated and asynchronous from each other. No one ant can adjust the internal state of any other ant. The only synchronisation mechanism is a global timer, which gives the ants a notion of time. At each time step, ants move one step onto an edge, or reach a node and evaluate its content.

5.1.1.2 Feature Learning and the Multi-Pheromone Mechanism

The MPACA gives ants the ability to learn various feature combinations. For each feature they acquire and its new feature combination, they deposit distinct pheromone values. No other ant algorithm in literature allows ants to learn features and deposit information about features in such a way. In fact, there is no other algorithm that attempts to leave pheromones for each and every descriptive attribute and value of all nodes in the domain. This vast information array of pheromones makes the MPACA a multi-pheromone approach like no other.

5.1.1.3 Multi-Colony Clustering via Colony Formation

The MPACA allows ants to determine which colony they should belong to, on their own initiative and without a controlling function. Starting with one colony per node and separate pheromones for every feature value of each node, the MPACA has the information power to create colony representations that, in theory, can match any cluster definition required.

5.1.2 Goals and Objectives met by the MPACA

The collective action of the criteria discussed above, is shown to perform accurate clustering via experimentation over a number of datasets. As with any other algorithm, the parameters offer a great deal of influence on how the algorithm reacts. Certain parameter settings exclude the formation of clusters, whilst other over-merge and prohibit linear separation to occur. It is important to know how these parameters work because they are an integral part of the model's specification. Synthetic datasets were used along with standard real-world datasets to explore the MPACA and to benchmark results against other algorithms. Two real-world datasets, one for logistics and one for mental health, were also used to show its utility for real, messy, and complex decision domains. The structure, dimensionality, and clustering objectives differ widely between the two sets, but the results show that the MPACA can induce and utilise patterns to produce helpful advice.

The project can be considered successful because it has created a demonstrably new algorithm that produces clustering information for multiple domain structures. Of course, there are elements that can be improved, and this will be the focus of future research. Some of the research choices and their rationale may need to be explored and the next section gives a brief resume of their exploration for this thesis.

5.2 Alternative Paths

The MPACA is the result of numerous incarnations. These include differences in the ant data structure, how they deposited pheromones, their set up for detecting particular pheromone values, their feature-merging mechanisms, colony formation, and the way the colonies are used to classify unknown objects. These implementations are of interest since they can still be reused in future variations.

5.2.1 Variation in the Ant Types

Initially, two different ant types were considered; home based ants and nomad ants. Home based ants always return back to their point of origin and deposit pheromones on their way back home. This causes star like formations to form as ants will always return back to their point of origin. On the other hand, nomad ants impose no pattern of ant behaviour specifically linked to their origin (home). It seemed sensible to let ants roam, constrained only by the paths and nodes they encounter. This allows clustering to be based on the areas of high-density ants, which are not constrained by their starting points.

5.2.2 Acquisition of Multiple Features on Each Dimension

In the current implementation of the MPACA, each ant is allowed to learn just one feature value per dimension. Feature matching consists of AND operators on feature values and there is no OR operator. In a previous iteration, ants were allowed to merge various feature values over the same dimension. This is possible by extending the feature range detected for dimensions or having more than one range. Alternatively it could achieved by allowing different nominal values for features such as colour or shape that are non-ordinal.

The complexity of allowing OR operators within features rather than only having AND operators across different attributes was not mitigated by any improvement in clustering results. Instead, the OR operator is effectively implemented at the colony level, as various ants can react to different feature values and still belong to the same colony; the colony shows the mixture of matching feature values rather than individual ants.

5.2.3 Feature Merging with No-Forgetting Mechanism

A key consideration in a previous iteration of the MPACA was the ability to acquire features and never forget such feature combinations. This meant that ants could eventually match most data nodes present given enough time and there was a problem with stabilisation of the colonies. A forgetting mechanism was therefore included in which feature merges that are not adequately reinforced within the time-window are removed.

5.2.4 Cluster Representation in First Order Logic

Initially, a first order logic description (FOL) was used to describe the clusters which are formed by the ants. The aim of the use of logic is to generate a description of the cluster in terms of the features of interests being carried by the ants, which form part of a colony. The FOL description

is expressive enough as a tool when describing linearly separable datasets but was less useful for non-linearly separable datasets. However, it is worth exploring further because FOL gives a rule-based approach to defining classes that has the potential for efficient implementation. The expressiveness can certainly accommodate non-linearly separable classes in theory and it really comes down to how easily the ant colonies provide enough guidance on the form of the FOL expression.

5.3 Advances of the Algorithm

The aim of the thesis is to demonstrate that a clustering algorithm can successfully operate by having a fully decentralised and distributed algorithm, that allow ants to learn and define clusters without any supervision. The presented results validate this objective. The MPACA is a mechanism which merges features and colonies asynchronously, with no guidance and no pre-set conditions on the outcome. It renders the approach very desirable in the current environment where parallel, multi-threaded and multi-CPU computation applications are highly sought after in industry. The MPACA is more than a collection of tweaks or minor adaptations of other ACO algorithms, but should be viewed as a new sub-domain for the ant algorithm paradigm. Its key innovation derives from the lack of global optimisation processes, which is why it is ideally suited at exploiting parallelisation.

The novelty of the MPACA also translates into general applicability to different data domains. The MPACA returns results which are often superior and usually equitable to other ant colony algorithms, as well as comparing favourably with traditional clustering algorithms. The algorithm performs well when the number of elements within a dataset are balanced. However, it has a certain weakness when applied to imbalanced datasets, such as the yeast dataset presented. This known limitation and proposed mechanisms to compensate for such limitations are given in section (5.4.4).

Crucially, although most of the empirical work used the MPACA as a clustering tool, it compared well with classification algorithms such as the Ant-Miner, which have the advantage of knowing the assignment of objects to classes during the learning process. Future work with the MPACA will explore its role in classification and determine whether the exploitation of prior knowledge about the classes can improve its results even more.

Type	Execution time
Parallel	10:43
Non-Parallel	12:26

TABLE 5.1: The Iris dataset as executed for100 instances separately with identical parameter settings for both parallel and non-parallel modes.

5.4 Current Limitations and Recommended Improvements

Developing and implementing a new computational intelligence algorithm has a number of phases. The most important is clarifying how the algorithm differs from alternatives and how these differences impact on performance. These have been the focus of this thesis but there are other issues that are also relevant. Optimisation is a particular concern because the learning process is comparatively slow for ACO, making it difficult to explore, for example, the best settings for parameters. This is where parallelisation may have its biggest influence because speeding up the learning process will help understand better where the MPACA needs adjustments and how it can fit different types of data sets such as those with uneven class sizes.

5.4.1 Parallel versus Non-Parallel

A parallel version of the MPACA was implemented using the .NET 4.5 framework, a Microsoft technology allowing a simplified parallel implementation. It was applied to the movement of the ants, which is where parallelisation would provide the maximum advantage. In this version, all ants are generated as separate parallel threads. Each ant moves independently from any other ant, as there is no selected sequence by which ants move.

Cluster quality derived between standard and parallel approaches does not differ. However, the variation between speed of execution is certainly significant. A preliminary investigation contrasting execution speeds is given in table (5.1). This is the result of executing the Iris dataset using the standard versus the parallel mode for 100 instances each, with both experiments using the exact same set of parameters. Results indicate that the parallel version is approximately 16% faster.

Even if parallel execution can significantly shorten execution time, improvements in the data structures used and superior hardware than currently available to the author, could furthermore accelerate the MPACA processing.

5.4.2 Parameters and Parameter Adjustment

Parameters are an important influence on the model's operation and all optimisation algorithms depend on finding the values that provide the best fit. The problem will always be that ACO methods are computationally expensive and time consuming, requiring careful optimisation of the MPACA code to generate the necessary execution speed. This would help the search for a best fit but more work is needed on better methods. There may even be room for improvement within the algorithm by having certain parameters merged or eliminated. This would benefit the MPACA and experiments already carried out on the parameters shows that some do not change their settings much when learning optimal values. This suggests they could be linked in to the architecture and governed by fewer parameters.

5.4.3 Termination Criteria

Termination is said to occur when ants reach a dynamic equilibrium between colony populations, or a maximum number of iterations has been reached. In many cases when the clusters to be recognised are relatively well dispersed, this stabilisation of colonies occurs rapidly. Thus, this mechanism is successful in cases when balanced datasets exist. However, a clear limitation occurs when tackling uneven datasets. This since, varying population sizes can cause premature termination, as too many ants join a particular reduced set of colonies, and from an opposite perspective, in some cases might cause the algorithm to fail to build the correct colonies and adequately terminate. A limitation discussed next.

5.4.4 Tackling Uneven Datasets

The MPACA is shown to return results comparable to literature for many datasets. However, it has failed to return good quality results on datasets which are unevenly distributed, as the case with the yeast dataset. The evaluation of such a failure is given in section (4.4.2), however it definitely opens further avenues where improvements can occur.

At present the algorithm has difficulties in defining smaller clusters which would be spatially positioned within a larger cluster. If the number of nodes representing the cluster to be learnt/recognised is too small, this will fail to generate the required critical mass for the creation of colonies. This occurs because ants join colonies depending on ant encounters, and a small denser concentration within a larger concentration is counter intuitive to this process. Postulating an alternative, if the colony level merge threshold is set to an overly high value, the smaller cluster region within the larger cluster will form a colony to define it. However, the larger cluster might

not have a colony which represents it in full, but have a number of smaller colonies. Thus, the larger cluster might fail to be recognised, which is also an undesired result.

A possible solution which could potentially solve both problems of learning uneven datasets and correct termination, is to investigate further the influence of the visibility parameter. This discussion builds on the material presented in chapters (3) and (4), where lower visibility is shown to slow termination, whilst higher visibility accelerates it. By allowing this parameter to be self-adjustable, depending on the node concentrations, nodes of higher concentration to each other would automatically lower visibility, whilst more sparse nodes would have higher visibility. Therefore, linking this parameter with node density should in theory improve the results attained.

5.5 Future work

A clear limitation existing within the current MPACA implementation is the final assignment of data elements to clusters. At present a centroid driven approach is used. Despite this returning comparable results to other algorithms in literature as given in chapter (4), this analysis is still deemed to be too crude. This mechanism is also prone to the pitfalls of K-Means and other centroid based algorithms described in chapter (2). In order to compensate for these limitations two alternative methods are currently being investigated; (i) a Bayesian driven membership calculation and (ii) a K-Nearest driven approach. Preliminary results attained for these methods are too premature to be included in this thesis.

5.5.1 Bayesian Cluster Membership Calculation

The Bayesian approach determines the most likely membership of a given data element depending on the features which constitute it, this being called the evidence set. This mechanism uses the notion that each colony consists of multiple ants, and each ant carries multiple features. Cluster membership for a data element is determined by analysing the population distribution of the ants in each class based on their feature values. The evidence set is analysed per feature, and a collective evaluation of all results is performed. For each feature, the count of this feature within the colony is taken as a ratio of that feature within the entire system. This results in a mechanism which is dependent on the frequency of encounters to probabilistically determine which colony data elements should belong to, as outlined in algorithm (10). Therefore, the probability of a data element de, being in a cluster c, is defined as in equation (5.1):

$$P(c|de) = \frac{P(de|c)P(c)}{P(de)} \tag{5.1}$$

Algorithm 10 Bayesian Approach to cluster membership calculation

for all data elements, $de \in$ Dataset, ds **do**
  Let HighestSigma $\rightarrow$ 0.0
  Let Evidence-Set $\rightarrow$ Features at Node
  Let $|F| \rightarrow$ all distinct features in the system
  for all colonyId $\in$ Colony **do**
    Let colonySigma $\rightarrow$ 1.0
    for all feature, $f \in$ evidenceSet **do**
      Let featuresInColony $\rightarrow$ all feature in this colony which match f
      Let numerator $\rightarrow$ featureInColonyCount + 1
      Comment: where the addition of value 1 ensures a level of smoothing
      Let featureInSystemCount $\rightarrow$ all features in system which match f
      Let denominator $\rightarrow$ featureInSystemCount + $|F|$
      Comment: where again $|F|$ ensure smoothing
      Let colonySigma $\rightarrow colonySigma \times \frac{numerator}{denominator}$
    end for
    if ($colonySigma \leq HighestSigma$) **then**
      Continue
    else
      Let HighestSigma $\rightarrow$ colonySigma
    end if
  end for
end for

Smoothing is applied to ensure that non-zero results are handled without skewing the calculations. Unlike the centroid calculation, which uses a similar proximity method on the locus of points, this mechanism uses the actual feature match counts to create the probability distribution.

5.5.2 K-Nearest Neighbourhood Cluster Membership Calculation

Algorithm 11 K-Nearest Neighbourhood approach to cluster membership calculation

for all data elements, $de \in$ Dataset, ds **do**
  Match de, to Node Id
  Let NodesInProximity $\rightarrow$ all nodes which are within K-neighbouring distance
  Let AntsWithinK-Radius $\rightarrow$ all ants on all nodes within NodesInProximity which have their deposit mode set to TRUE
  Let ColonyCount $\rightarrow$ all distinct colonies which are in AntsWithinK-Radius
  Let ColonyVotingArray[ColonyCount] $\rightarrow$ 0
  for all ant $\in$ Ants-Within-K-Radius **do**
    ColonyVotingArray[ant.ColonyId]++
  end for
  Set Node membership to highest colony count Id in ColonyVotingArray
end for

The K-Nearest Neighbourhood mechanism uses the knowledge of colony distribution on nodes. Each node is representative of an original data element, and colonies of ants are distributed unevenly on such nodes, where certain node groupings have a higher tendency to be populated with ants belonging to one colony rather than another. The mechanism first filters out ants

within colonies which are not in deposit mode are filtered since technically they should not belong there. Subsequently each data element (node) is allocated to a colony depending on the most frequent colony Id of ants present on it, and also of nodes within its K-Nearest distance, which is effectively a majority polling mechanism. All nodes which are less than K-steps away are collectively grouped under this capping, as outlined in algorithm (11).

5.5.3 Ongoing Research

During this research period, the MPACA has been introduced at a number of conferences and publications, namely; [Chircop and Buckingham, 2013], [Chircop and Buckingham, 2014], [Chircop and Buckingham, 2011b], [Chircop and Buckingham, 2011a]. Given the success of applying the MPACA to real-world domains, amongst these the GRiST [Buckingham and Adams, 2013] and the ADVANCE [adv, 2013] datasets, further work is continuing in these areas. Both domains have their own set of challenges for ACO and the promising initial results of the MPACA make them well worth addressing. Both are essentially prediction problems that are most obviously modelled by regression. However, the exact number produced by the prediction is not as important as the decision class to which it is assigned. For the mental-health domain, the GRiST clinical risk judgements have one of eleven categories, zero to ten, but the psychological representation of risk has less granular categories that map onto risk management decisions. These are more like low, medium, and high risk, where the most important category is the high-risk one. The MPACA has shown some success in detecting these categories. Future work will explore how its clustering abilities can be applied to GRiST data more associated with risk management than risk evaluation to help provide more robust linkage between risk categories and their management. It is clear from the current GRiST data that only a subset is required for evaluating the risks and that much of the supporting data is about how to manage them. There is little understanding about the role of these management data and finding patterns within them is an ideal application of the MPACA. The rewards are high because it is clear that the clinically-relevant representations of risk (e.g. none, low, medium, high, maximum) are linked to how patients in each category are managed. Management data can help define and refine the categories, which then feeds back into better thresholds for assigning patients via the evaluation process.

Regarding the ADVANCE logistics domain, the prediction was for the expected demand on vehicle space each day so that haulier companies know how many lorries to deploy. Again, the exact number is less important than the impact on decisions. If the demand rises or falls more than a threshold amount from the normal amount, this has severe consequences, either

by wasting space (and money) or by failing to deliver goods on time. The MPACA showed good results when clustering data to find those days when the thresholds are exceeded. Future work will attempt to match the thresholds more accurately to the cognitive model of decisions Buckingham et al. [2012]. In this case, we have a clear understanding of what situations drive different decisions but not how to detect those situations from the data; the MPACA could be a valuable resource for achieving the latter.

5.5.4 Application of the MPACA as a Classifier

Results presented in chapter (4) demonstrate that the MPACA can quite easily be used in a classification mode. This is still ongoing research work, and further experimentation is required. It would certainly be interesting to compare the MPACA as a classifier versus results of the other classifiers presented in the results chapter.

An important guideline to remember for future research on the MPACA is to avoid chasing performance optimisation without understanding how it is being achieved. Otherwise, the particular qualities of the MPACA could be lost or diluted, with improvements failing to come from the metaphor that has motivated the research in the first place. Future work will attempt to exploit the novel strengths of the algorithm rather than forcing it to fit unsuitable problem domains by bloating its functionality and diluting its distinctive properties.

5.6 Concluding Arguments

The basic learning mechanisms of the MPACA are demonstrated over relatively simple domains. These domains were deliberately chosen to display the ability of the MPACA to form colonies representative of the required clusters. Despite their simplicity they demonstrate that the MPACA successfully operates as a clustering tool and is tractable. The results illustrate the potential to learn clusters, with further testing applied to more real-world domains attaining similar but less ostentatious results.

Finally, the MPACA can be best summarised into the following; “a process of self-determination with ants choosing which features to merge and which colonies (clusters) they should combine into”.

Bibliography

(2013). Advance: Advanced predictive-analysis-based decision-support engine for logistics. http://www.advance-logistics.eu/. [Online; accessed Sept-2013].

(2013). University at buffalo, the state university of new york. http://www.cse.buffalo.edu/faculty/azhang/data-mining/density-based.ppt/. [Online; accessed Sept-2013].

1902encyclopedia (2013). Animal kingdom part 3. http://www.1902encyclopedia.com/A/ANI/animal-kingdom-03.html. [Online; accessed Nov-2013].

A, D. (2013). Sintef (2013, may 22). big data, for better or worse: 90% of world's data generated over last two years. *ScienceDaily*. http://www.sciencedaily.com/releases/2013/05/130522085217.htm. [Online; accessed Sept-2013].

Adleman, L. M. (1994). Molecular computation of solutions to combinatorial problems. *Science*, 266(11):1021–1024.

Admane, L., Benatchba, K., Koudil, M., Siad, L., and Maziz, S. (2006). Antpart: an algorithm for the unsupervised classification problem using ants. *Applied Mathematics and Computation*, 180(1):16–28.

Agarwal, A., Lim, M. H., Er, M. J., and Chew, C. Y. (2005). Aco for a new tsp in region coverage. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2005), 2005.*, pages 1717–1722.

Agrawal, R., Gehrke, J., Gunopulos, D., and Raghavan, P. (1998). Automatic subspace clustering of high dimensional data for data mining applications. *SIGMOD Rec.*, 27(2):94–105.

Akbar, Y. (2008). Fire ants form bridge. http://hungeree.com/nature/fire-ants-form-bridge/. [Online; accessed Sept-2013].

Al-Saleh, M. F. and Yousif, A. E. (2009). Properties of the standard deviation that are rarely mentioned in classrooms. *Austrian Journal of Statistics*, 38(3):193–202.

Analoui, M., Beheshti, M., Tayefeh Mahmoudi, M., and Jadidi, Z. (2010). Tecno-streams approach for content-based image retrieval. In *Second World Congress on Nature and Biologically Inspired Computing (NaBIC), 2010*, pages 109–114.

Andreev, K. and Räcke, H. (2004). Balanced graph partitioning. In *Proceedings of the Sixteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '04, pages 120–124, New York, NY, USA. ACM.

Ankerst, M., Breunig, M. M., Kriegel, H. P., and Sander, J. (1999). Optics: ordering points to identify the clustering structure. In *Proceedings of the 1999 ACM SIGMOD international conference on Management of data*, SIGMOD '99, pages 49–60, New York, NY, USA. ACM.

Antoine, V., Monmarché, N., and Slimane, M. (2008). Data clustering with artificial ants: the api algorithm case study. In *Proccedings of META'08*, Hammamet, Tunisia.

Azzag, H., Monmarché, N., Slimane, M., Guinot, C., and Venturini, G. (2003). AntTree: A new model for clustering with artificial ants. In Banzhaf, W., Christaller, T., Dittrich, P., Kim, J. T., and Ziegler, J., editors, *Advances in Artificial Life - Proceedings of the 7th European Conference on Artificial Life (ECAL)*, volume 2801 of *Lecture Notes in Artificial Intelligence*, pages 564–571. Springer Verlag Berlin, Heidelberg.

Azzag, H., Venturini, G., Oliver, A., and Guinot, C. (2007). A hierarchical ant based clustering algorithm and its use in three real-world applications. *European Journal of Operational Research*, 179(3):906–922.

Bache, K. and Lichman, M. (2013). UCI machine learning repository. http://archive.ics.uci.edu/ml. [Online; accessed Dec-2013].

Baraldi, A. and Blonda, P. (1999). A survey of fuzzy clustering algorithms for pattern recognition. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 29(6):778–785.

Bellman, R. (1957). *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1 edition.

Berkhin, P. (2006). A survey of clustering data mining techniques. In Kogan, J., Nicholas, C., and Teboulle, M., editors, *Grouping Multidimensional Data*, pages 25–71. Springer Berlin Heidelberg.

Bertelle, C., Dutot, A., Guinand, F., and Olivier, D. (2003). Color ant population algorithm for dynamic distribution in simulation. In *Proceedings of the ESS Conference*, Delft, The Netherlands.

Bertelle, C., Dutot, A., Guinand, F., and Olivier, D. (2007). Organization detection for dynamic load balancing in individual-based simulations. *Multiagent and Grid Systems, special issue on Nature-Inspired Systems for Parallel, Asynchronous and Decentralized Environments*, 3(1):141–163.

Bertelle, C., Dutot, A., Guinand, F., Olivier, D., et al. (2006). Organization detection using emergent computing. *International Transactions on Systems Science and Applications*, 2(1):61–69.

Bezdek, J. C., Ehrlich, R., and Full, W. (1984). : The fuzzy c-means clustering algorithm. *"Computers and Geosciences*, 10(2–3):191–203.

Bin, W. and Zhongzhi, S. (2001). A clustering algorithm based on swarm intelligence. In *ICII 2001 - International Conferences on Info-tech and Info-net Beijing.*, volume 3, pages 58–66.

Bishop, J. (1989). Stochastic searching networks. In *1st IEE Conf. on Artificial Neural Networks*, pages 329–331.

Bishop, J. and Torr, P. (1992). The stochastic search network. In Linggard, R., Myers, D., and Nightingale, C., editors, *Neural Networks for Vision, Speech and Natural Language*, volume 1 of *BT Telecommunications Series*, pages 370–387. Springer Netherlands.

Blum, C. and Merkle, D., editors (2008). *Swarm Intelligence: Introduction and Applications (Natural Computing Series)*. Springer, 1 edition.

Bonabeau, E., Dorigo, M., and Théraulaz, G. (1999). *Swarm intelligence: From natural to artificial systems*. Santa Fe Institute Studies in the Sciences of Complexity. Oxford University Press, Oxford.

Boryczka, U. (2010). Ant colony metaphor in a new clustering algorithm. *Control & Cybernetics*, 39(2).

Bougenière, G., Cariou, C., Chehdi, K., and Gay, A. (2009). Non parametric stochastic expectation maximization for data clustering. In Filipe, J. and Obaidat, M., editors, *E-business and Telecommunications*, volume 23 of *Communications in Computer and Information Science*, pages 293–303. Springer Berlin Heidelberg.

Breaban, M. E. and Luchian, H. (2011). Pso aided k-means clustering: Introducing connectivity in k-means. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, GECCO '11, pages 1227–1234, New York, NY, USA. ACM.

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

Bruni, L. (2002). *Vilfredo Pareto and the birth of modern microeconomics*. Elgar, Cheltenham [u.a].

Buckingham, C. D. and Adams, A. (2011). The grist web-based decision support system for mental-health risk assessment and management. In *Proceedings of the First BCS Health in Wales/ehi2 joint Workshop*, pages 37–40.

Buckingham, C. D. and Adams, A. E. (2013). Grist galatean risk and safety tool. http://www.egrist.org/. [Online; accessed Oct-2013].

Buckingham, C. D., Adams, A. E., and Mace, C. (2008). Cues and knowledge structures used by mental-health professionals when making risk assessments. *Journal of Mental Health*, 17(3):299–314.

Buckingham, C. D., Buijs, P., Welch, P. G., Kumar, A., and Ahmed, A. (2012). Developing a cognitive model of decision-making to support members of hub-and-spoke logistics networks. In Elisabeth Ilie-Zudor, Zsolt Kemény, L. M., editor, *Proceedings of the 14th International Conference on Modern Information Technology in the Innovation Processes of the Industrial Enterprises*, pages 14–30.

Bullnheimer, B., Hartl, R., and Strauss, C. (1999). A New Rank Based Version of the Ant System: A Computational Study. *Central European Journal for Operations Research and Economics*, 7(1):25–38.

Bullnheimer, B., K. G. and Strauss, C. (1997). Parallelization strategies for the ant system. *High Performance Algorithms and Software in Nonlinear Optimization*, pages 87–100.

Cano, A., Zafra, A., and Ventura, S. (2013). An interpretable classification rule mining algorithm. *Inf. Sci.*, 240:1–20.

Cardon, A., Dutot, A., Guinand, F., and Olivier, D. (2006). Competing ants for organization detection application to dynamic distribution. In Aziz-Alaoui, M. and Bertelle, C., editors, *Emergent Properties in Natural and Artificial Dynamical Systems*, Understanding Complex Systems, pages 25–52. Springer Berlin Heidelberg.

Chaimontree, S., Atkinson, K., and Coenen, F. (2010). Clustering in a multi-agent data mining environment. In Cao, L., Bazzan, A., Gorodetsky, V., Mitkas, P., Weiss, G., and Yu, P., editors, *Agents and Data Mining Interaction*, volume 5980 of *Lecture Notes in Computer Science*, pages 103–114. Springer Berlin Heidelberg.

Chandrasekar, R. and Srinivasan, T. (2007). An improved probabilistic ant based clustering for distributed databases. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 2701–2706, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Chazelle, B. (2000). A minimum spanning tree algorithm with inverse-ackermann type complexity. *J. ACM*, 47(6):1028–1047.

Chen, B., Song, S., and Chen, X. (2007a). A multi-ant colony system for vehicle routing problem with time-dependent travel times. In *2007 IEEE International Conference on Automation and Logistics*, pages 446–449.

Chen, E. and Liu, X. (2009). Multi-colony ant algorithm using both repulsive operator and pheromone crossover based on multi-optimum for tsp. In Wang, S., Yu, L., Wen, F., He, S., Fang, Y., and Lai, K. K., editors, *BIFE*, pages 69–73. IEEE Computer Society.

Chen, J., MacEachren, A., and Peuquet, D. (2009). Constructing overview + detail dendrogram-matrix views. *IEEE Transactions on Visualization and Computer Graphics,*, 15(6):889–896.

Chen, L., Xu, X. H., and Chen, Y. X. (2004). An adaptive ant colony clustering algorithm. In *In Proceedings of the Third International Conference on Machine Learning and Cybernetics*, pages 1387–1392. IEEE.

Chen, M., Fu, Y. C., Ge, J., Zhou, X. K., and Cui, Z. M. (2007b). Study of logistics vehicle routing problem based on gis. *2007 Workshop on Intelligent Information Technology Applications,*, 0:129–132.

Chen, Y., Chen, L., and Tu, L. (2006). Parallel ant colony algorithm for mining classification rules. In *2006 IEEE International Conference on Granular Computing*, pages 85–90.

Chialvo, D. R. and Millonas, M. M. (1995). How swarms build cognitive maps. In Steels, L., editor, *The Biology and Technology of Intelligent Autonomous Agents*, volume 144 of *NATO ASI Series*, pages 439–450. Springer Berlin Heidelberg.

Chintalapati, J., Arvind, M., Priyanka, S., Mangala, N., and Valadi, J. (2010). Parallel ant-miner (pam) on high performance clusters. In Panigrahi, B., Das, S., Suganthan, P., and Dash, S.,

editors, *Swarm, Evolutionary, and Memetic Computing*, volume 6466 of *Lecture Notes in Computer Science*, pages 270–277. Springer Berlin Heidelberg.

Chircop, J. and Buckingham, C. D. (2011a). Clustering via a multi-pheromone ant colony algorithm. In *EVOLVE 2011, A bridge between Probability, Set Oriented Numerics and Evolutionary Computation*, pages 1–6.

Chircop, J. and Buckingham, C. D. (2011b). A multiple pheromone algorithm for cluster analysis. In *ICSI 2011: International conference on swarm intelligence*, pages 1–10.

Chircop, J. and Buckingham, C. D. (2013). The multiple pheromone ant clustering algorithm and its application to real world domains. In *2013 Federated Conference on Computer Science and Information Systems (FedCSIS),*, pages 27–34.

Chircop, J. and Buckingham, C. D. (2014). A multiple pheromone ant clustering algorithm. In Terrazas, G., Otero, F. E. B., and Masegosa, A. D., editors, *Nature Inspired Cooperative Strategies for Optimization (NICSO 2013)*, volume 512 of *Studies in Computational Intelligence*, pages 13–27. Springer International Publishing.

Chrétien, L. (1996). *Organisation spatiale du matériel provenant de l'excavation du nid chez Messor barbarus et des cadavres d'ouvriéres chez Lasius niger (Hymenopterae: Formicidae).* PhD thesis, Département de Biologie Animale, Université Libre de Bruxelles.

Chu, S. C., Huang, H. C., Roddick, J. F., and Pan, J.-S. (2011). Overview of algorithms for swarm intelligence. In Jedrzejowicz, P., Nguyen, N., and Hoang, K., editors, *Computational Collective Intelligence. Technologies and Applications*, volume 6922 of *Lecture Notes in Computer Science*, pages 28–41. Springer Berlin Heidelberg.

Chu, S. C., Roddick, J. F., Pan, J. S., and Su, C. J. (2003). Parallel ant colony systems. In Zhong, N., Ras, Z., Tsumoto, S., and Suzuki, E., editors, *Foundations of Intelligent Systems*, volume 2871 of *Lecture Notes in Computer Science*, pages 279–284. Springer Berlin Heidelberg.

Chuang, L. Y., Lin, Y. D., and Yang, C. H. (2012). An improved particle swarm optimization for data clustering. In *Proceedings of the International MultiConference of Engineers and Computer Scientists 2012 Vol I, IMECS 2012,*.

Cumps, B., Martens, D., De Backer, M., Viaene, S., Dedene, G., Haesen, R., Snoeck, M., and Baesens, B. (2009). Inferring rules for business/ICT alignment using Ants.

Dawson, L. and Stewart, I. (2013). Improving ant colony optimization performance on the gpu using cuda. In *2013 IEEE Congress on Evolutionary Computation (CEC),*, pages 1901–1908.

de Castro, L. N. (2007). Fundamentals of natural computing: an overview. *Physics of Life reviews*, 4:1–36.

de Castro, L. N. and Von Zuben, F. J. (1999). Artificial immune systems - part i: Basic theory and applications. Technical report, Department of Computer Engineering and Industrial Automation, School of Electrical and Computer Engineering, State University of Campinas, Brazil.

Demontis, R. (2009). A simple np-hard problem. *SIGACT News*, 40(2):45–48.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B*, 39:1–38.

Deneubourg, J. L., Aron, S., Goss, S., and Pasteels, J. M. (1990a). The self-organizing exploratory pattern of the argentine ant. *Journal of Insect Behavior*, 3(2):159–168.

Deneubourg, J. L., Goss, S., Franks, N., Sendova-Franks, A., Detrain, C., and Chrétien, L. (1990b). The dynamics of collective sorting robot-like ants and ant-like robots. In *Proceedings of the first international conference on simulation of adaptive behavior on From animals to animats*, pages 356–363, Cambridge, MA, USA. MIT Press.

Dong, M., Wang, N., and Tao, J. (2009). Dna computing in control systems: a survey. In *Proceedings of the 21st annual international conference on Chinese control and decision conference*, CCDC'09, pages 4978–4982, Piscataway, NJ, USA. IEEE Press.

Dorigo, M. (1992). *Optimization, Learning and Natural Algorithms (in Italian)*. PhD thesis, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy.

Dorigo, M., Birattari, M., and Stützle, T. (2006). Ant colony optimization. *Computational Intelligence Magazine, IEEE*, 1(4):28–39.

Dorigo, M. and Gambardella, L. (1997a). Ant Colony Sytem: A Cooperative Learning Approach to the Travelling Salesman Problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66.

Dorigo, M. and Gambardella, L. M. (1997b). Ant colonies for the travelling salesman problem. *Biosystems*, 43(2):73 – 81.

Dorigo, M., Maniezzo, V., and Colorni, A. (1991a). Positive feedback as a search strategy. Technical Report 91-016, Politecnico di Milano, Italy.

Dorigo, M., Maniezzo, V., and Colorni, A. (1991b). The Ant System: An Autocatalytic Optimizing Process. Technical Report 91-016 Revised, Milano, Italy.

Dorigo, M., Maniezzo, V., and Colorni, A. (1996). The Ant System: Optimization by a Colony of Cooperating Agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, 26(1):29–41.

Duan, H., Liu, S., and Lei, X. (2008). Air robot path planning based on intelligent water drops optimization. In *Neural Networks, 2008. IJCNN 2008. (IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, pages 1397–1401.

Dunn, J. C. (1974). Well separated clusters and optimal fuzzy-partitions. *Journal of Cybernetics*, 4:95–104.

Dussutour, A., Nicolis, S. C., Shephard, G., Beekman, M., and Sumpter, D. J. T. (2009). The role of multiple pheromones in food recruitment by ants. *J Exp Biol*, 212(15):2337–2348.

Eker, J., Janneck, J. W., Lee, E. A., Liu, J., Liu, X., Ludvig, J., Neuendorffer, S., Sachs, S., and Xiong, Y. (2003). Taming heterogeneity - the ptolemy approach. *Proceedings of the IEEE*, 91(1):127–144.

Ester, M., Kriegel, H. P., Sander, J., and Xu, X. (1996a). A density-based algorithm for discovering clusters in large spatial databases with noise. In Simoudis, E., Han, J., and Fayyad, U. M., editors, *KDD*, pages 226–231. AAAI Press.

Ester, M., Kriegel, H. P., Sander, J., and Xu, X. (1996b). A density-based algorithm for discovering clusters in large spatial databases with noise. In Simoudis, E., Han, J., and Fayyad, U. M., editors, *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pages 226–231. AAAI Press.

Flikkema, P. and Leid, J. (2005). Bacterial communities: a microbiological model for swarm intelligence. In *Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005.*, pages 416–419.

Franti, P., Virmajoki, O., and Hautamaki, V. (2006). Fast agglomerative clustering using a k-nearest neighbor graph. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(11):1875–1881.

Gambardella, L. and Dorigo, M. (1996). Solving Symmetric and Asymmetric TSPs by Ant Colonies. In IEEE, editor, *Proceedings of the third IEEE International Conference on Evolutionary Computation (ICEC)*, pages 622–627, Nagoya, Japan. IEEE Press.

Ghosh, A., Halder, A., Kothari, M., and Ghosh, S. (2008). Aggregation pheromone density based data clustering. *Information Sciences*, 178(13):2816–2831.

Grassé, P. P. (1959). La reconstruction du nid et les coordinationsinter-individuelles chez bellicositermes natalensis et cubitermes sp. la theorie de la stigmergie: Essai dinterpretation du comportement des termites constructeurs. *Insect. Sociaux*, 6:41–81.

Greenacre, M. (2013). Correspondence analysis and related methods, department of statistics, stanford university. retrieved novemeber 2013. http://www.econ.upf.edu/~michael/stanford/maeb4.pdf. [Online; accessed Sept-2013].

Grira, N., Crucianu, M., and Boujemaa, N. (2004). Unsupervised and Semi-supervised Clustering: a Brief Survey. *A Review of Machine Learning Techniques for Processing Multimedia Content, Report of the MUSCLE European Network of Excellence (FP6)*.

Guntsch, M. (2004). *Ant Algorithms in Stochastic and Multi-Criteria Environments*. PhD thesis, Universität Fridericiana zu Karlsruhe.

Guntsch, M. and Middendorf, M. (2001). Pheromone Modification Strategies for Ant Algorithms Applied to Dynamic TSP. In Boers, E., Gottlieb, J., Lanzi, P., Smith, R., Cagnoni, S., Hart, E., Raidl, G., and Tijink, H., editors, *Applications of Evolutionary Computing : EvoWorkshops 2001: EvoCOP, EvoFlight, EvoIASP, EvoLearn, and EvoSTIM*, volume 2037 of *Lecture Notes in Computer Science*, pages 213–222, Como, Italy. Springer Berlin / Heidelberg.

Guntsch, M. and Middendorf, M. (2002). Applying Population Based ACO to Dynamic Optimization Problems. In Dorigo, M., Di Caro, G., and Sampels, M., editors, *Proceedings of the Third International Workshop on Ant Algorithms (ANTS'2002)*, volume 2463 of *Lecture Notes in Computer Science*, pages 111–122, Brussels, Belgium. Springer Verlag.

Guo, G., Wang, H., Bell, D., Bi, Y., and Greer, K. (2003). Knn model-based approach in classification. In Meersman, R., Tari, Z., and Schmidt, D., editors, *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*, volume 2888 of *Lecture Notes in Computer Science*, pages 986–996. Springer Berlin Heidelberg.

Gutowitz, H. (1995). Complexity-Seeking Ants. In Moran, F., Moreno, A., Merelo, J., and Chacon, P., editors, *Proceedings of the Third European Conference on Artificial Life (ECAL)*, pages 429–439, Granada, Spain. Springer Verlag.

Halder, A., Ghosh, S., and Ghosh, A. (2008). Aggregation pheromone density based classification. In *International Conference on Information Technology, 2008. ICIT '08.*, pages 100–105.

Hamming, R. W. (1950). Error Detecting and Error Correcting Codes. *Bell System Technical Journal*, 26(2):147–160.

Han, J. (2005). *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Handl, J., Knowles, J., and Dorigo, M. (2003a). Ant-based clustering: a comparative study of its relative performance with respect to k-means, average link and 1d-som. In *Proceedings of the Third International Conference on Hybrid Intelligent Systems, IOS Press*.

Handl, J., Knowles, J., and Dorigo, M. (2003b). On the performance of ant-based clustering. In *Design and Application of Hybrid Intelligent Systems*, volume 104 of *Frontiers in Artificial Intelligence and Applications*, pages 204–213. Amsterdam, The Netherlands: IOS Press.

Handl, J., Knowles, J., and Dorigo, M. (2006). Ant-based clustering and topographic mapping. *Artificial Life*, 12(1):35–61.

Hao, Y., Shen, Z., and Zhao, Y. (2009). Path planning for aircraft based on maklink graph theory and multi colony ant algorithm. In *Proceedings of the 2009 International Joint Conference on Computational Sciences and Optimization - Volume 02*, CSO '09, pages 232–235, Washington, DC, USA. IEEE Computer Society.

Hasan, A., Jafar, M., and Sivakumar, R. (2011). A survey: hybrid evolutionary algorithms for cluster analysis. *Artif. Intell. Rev.*, 36(3):179–204.

He, J., Long, D., and Chen, C. (2007). An improved ant-based classifier for intrusion detection. In *Proceedings of the Third International Conference on Natural Computation - Volume 04*, ICNC '07, pages 819–823, Washington, DC, USA. IEEE Computer Society.

Heppner, F. and Grenander, U. (1990). A stochastic nonlinear model for coordinated bird flocks. In Krasner, E., editor, *The ubiquity of chaos*, pages 233–238. AAAS Publications.

Hinneburg, A. and Keim, D. A. (1998). An efficient approach to clustering in large multimedia databases with noise. In Agrawal, R., Stolorz, P. E., and Piatetsky-Shapiro, G., editors, *KDD*, pages 58–65. AAAI Press.

Hochbaum, D. S., editor (1997). *Approximation algorithms for NP-hard problems*. PWS Publishing Co., Boston, MA, USA.

Hoffman, E. and Gardner, R. (1983). Evaluation of uncertainties in environmental radiological assessment models. Technical Report NUREG/CR-3332, Till, J.E.; Meyer, H.R. (eds) Radiological Assessments: a Textbook on Environmental Dose Assessment. Washington, DC: U.S. Nuclear Regulatory Commission, Milano, Italy.

Holland, J. H. (1973). Genetic algorithms and the optimal allocation of trials. *SIAM J. Comput.*, 2(2):88–105.

Hsiao, Y.-T., Chuang, C.-L., Jiang, J.-A., and Chien, C.-C. (2005). A novel optimization algorithm: space gravitational optimization. In *IEEE International Conference on Systems, Man and Cybernetics, 2005*, volume 3, pages 2323–2328 Vol. 3.

Inbarani, H. H. and Thangavel, K. (2006). Clickstream intelligent clustering using accelerated ant colony algorithm. In *ADCOM 2006 International Conference on Advanced Computing and Communications, 2006.*, pages 129–134.

Ingaramo, D. A., Leguizamón, M. G., and Errecalde, M. L. (2005). Adaptive clustering with artificial ants. *Journal of Computer Science & Technology*, 5.

Jafar, O. A. M. and Sivakumar, R. (2010). Ant-based clustering algorithms: A brief survey. *International journal of computer theory and engineering*, 2(5):1793–8201.

Jain, A. K. and Dubes, R. C. (1988). *Algorithms for clustering data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Jain, A. K., Murty, M. N., and Flynn, P. J. (1999). Data clustering: A review. *ACM Comput. Surv.*, 31(3):264–323.

James, E. A., Milenkiewicz, M. T., and Bucknam, A. (2008). *Participatory Action Research for Educational Leadership: Using Data-Driven Decision Making to Improve Schools*. SAGE Publications.

Jebara, T. (2002). *Discriminative, Generative and Imitative Learning*. PhD thesis. AAI0804044.

Jiang, W., Xu, Y., and Xu, Y. (2005). A novel data mining method based on ant colony algorithm. In Li, X., Wang, S., and Dong, Z., editors, *Advanced Data Mining and Applications*, volume 3584 of *Lecture Notes in Computer Science*, pages 284–291. Springer Berlin Heidelberg.

Jin, P., Zhu, Y., Hu, K., and Li, S. (2006). Classification rule mining based on ant colony optimization algorithm. In Huang, D.-S., Li, K., and Irwin, G., editors, *Intelligent Control and Automation*, volume 344 of *Lecture Notes in Control and Information Sciences*, pages 654–663. Springer Berlin Heidelberg.

Kablan, A. and Ng, W. (2010). High frequency trading strategy using the hilbert transform. In *2010 Sixth International Conference on Networked Computing and Advanced Information Management (NCM),*, pages 466–471.

Kamkar, I., Akbarzadeh T, M. R., and Yaghoobi, M. (2010). Intelligent water drops a new optimization algorithm for solving the vehicle routing problem. In *2010 IEEE International Conference on Systems Man and Cybernetics (SMC),*, pages 4142–4146.

Kennedy, J. and Eberhart, R. C. (1995). Particle swarm optimization. In *Proceedings., IEEE International Conference on Neural Networks, 1995.*, volume 4, pages 1942–1948 vol.4. IEEE.

Kheirkhahzadeh, M. and Barforoush, A. (2009). A hybrid algorithm for the vehicle routing problem. In *CEC '09. IEEE Congress on Evolutionary Computation, 2009.*, pages 1791–1798.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science, Number 4598, 13 May 1983*, 220, 4598:671–680.

Kordon, A. K. (2010). Swarm intelligence: The benefits of swarms. In *Applying Computational Intelligence*, pages 145–174. Springer Berlin Heidelberg.

Korürek, M. and Nizam, A. (2008). A new arrhythmia clustering technique based on ant colony optimization. *Journal of Biomedical Informatics*.

Kreyszig, E. (2000). *Advanced Engineering Mathematics: Maple Computer Guide*. John Wiley & Sons, Inc., New York, NY, USA, 8th edition.

Kriegel, H. P., Kröger, P., and Zimek, A. (2009). Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Trans. Knowl. Discov. Data*, 3(1):1:1–1:58.

Krüger, F., M. M. and Merkle, D. (1998). Studies on a parallel ant system for the bsp model.

Kruskal, J. (1964). Nonmetric multidimensional scaling: a numerical method. *Psychometrika*, 29:115–129.

Kryszkiewicz, M. (2013). Determining cosine similarity neighborhoods by means of the euclidean distance. In Skowron, A. and Suraj, Z., editors, *Rough Sets and Intelligent Systems - Professor Zdzisław Pawlak in Memoriam*, volume 43 of *Intelligent Systems Reference Library*, pages 323–345. Springer Berlin Heidelberg.

Kuntz, P., Snyers, D., and Layzell, P. (1998). A stochastic heuristic for vizualising graph clusters in a bi-dimensional space prior to partitioning. *Journal of Heuristics*, 5(3):327–351.

Labroche, N. (2003). *Modélisation du système de reconnaissance chimique des fourmis pour le problème de la classification non-supervisée : application à la mesure d'audience sur Internet*. Thèse de doctorat, Laboratoire d'Informatique, Université de Tours.

Labroche, N., Monmarché, N., and Venturini, G. (2002a). A new clustering algorithm based on the chemical recognition system of ants. In Harmelen, F., editor, *Proceedings of the 15th European Conference on Artificial Intelligence*, pages 345–349, Lyon, France. IOS Press.

Labroche, N., Monmarché, N., and Venturini, G. (2003a). AntClust: Ant Clustering and Web Usage Mining. In Cantu-Paz, E., editor, *Genetic and Evolutionary Computation Conference*, volume 2723 of *Lecture Notes in Computer Science*, pages 25–36, Chicago. Springer-Verlag Telos.

Labroche, N., Monmarché, N., and Venturini, G. (2003b). Web sessions clustering with artificial ants colonies. In *WWW2003 Conference*, page Poster, Budapest.

Labroche, N., Richard, F. J., Monmarché, N., Lenoir, A., and Venturini, G. (2002b). Modelling of the chemical recognition system of ants. In Hemelrijk, C. K., editor, *International Workshop on Self-Organization and Evolution of Social Behaviour*, pages 283–292, Monte Verità, Ascona, Switzerland.

Li, Z. F. and Bai, H. (2010). Multi-ant colony optimization algorithm for the route optimization of logistic distribution. In *Second International Conference on Computational Intelligence and Natural Computing Proceedings (CINC), 2010*, volume 1, pages 141–144.

Lim, C. P. and Dehuri, S. (2012). *Innovations in Swarm Intelligence*. Springer Publishing Company, Incorporated.

Ling, S. and Wei, W. (2009). Multi-ant-colony based multi-path routing algorithm for overlay network. In *Proceedings of the 2009 WRI Global Congress on Intelligent Systems - Volume 01*, GCIS '09, pages 188–192, Washington, DC, USA. IEEE Computer Society.

Lioni, A., Sauwens, C., Theraulaz, G., and Deneubourg, J. L. (2001). The dynamics of chain formation in oecophylla longinoda. *Journal of Insect Behavior*, 14:679–696.

Liu, S., Dou, Z. T., Li, F., and Huang, Y. L. (2004). A new ant colony clustering algorithm based on DBSCAN. In *3rd International Conference on Machine learning and Cybernetics*, pages 1491–1496, Shanghai.

Liu, Z. and Chai, Y. (2006). A hybrid ant colony algorithm for capacitated vehicle routing problem. In *SMC '06. IEEE International Conference on Systems, Man and Cybernetics, 2006.*, volume 5, pages 3907–3911.

Lumer, E. and Faieta, B. (1994). Diversity and Adaptation in Populations of Clustering Ants. In Cliff, D., Husbands, P., Meyer, J., and W., S., editors, *Proceedings of the Third International Conference on Simulation of Adaptive Behavior (SAB)*, pages 501–508. MIT Press, Cambridge, Massachusetts.

MacQueen, J. B. (1967). Some methods for classification and analysis of multivariate observations. In Cam, L. M. L. and Neyman, J., editors, *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press.

Maesschalck, R. D., Jouan-Rimbaud, D., and Massart, D. (2000). The mahalanobis distance. *Chemometrics and Intelligent Laboratory Systems*, 50(1):1–18.

Maimon, O. and Rokach, L. (2005). *Data Mining and Knowledge Discovery Handbook*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Maniezzo, V. (1999). Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem. *INFORMS J. on Computing*, 11(4):358–369.

Manjurul Islam, M. M., Waselul Hague Sadid, M., Mamun Ar Rashid, S. M., and Kabir, M. J. (2006). An implementation of aco system for solving np-complete problem; tsp. In *International Conference on Electrical and Computer Engineering, ICECE '06. 2006.*, pages 304–307.

Martens, D., Bruynseels, L., Baesens, B., Willekens, M., and Vanthienen, J. (2008). Predicting going concern opinion with data mining. *Decision Support Systems*, 45(4):76–777. Information Technology and Systems in the Internet-Era.

Martens, D., De Backer, M., Haesen, R., Vanthienen, J., Snoeck, M., and Baesens, B. (2007). Classification with ant colony optimization. *Evolutionary Computation, IEEE Transactions on*, 11(5):651–665.

Martens, D., van Gestel, T., De Backer, M., Haesen, R., Vanthienen, J., Mues, C., and Baesens, B. (2010). Credit rating prediction using ant colony optimization. Open access publications from katholieke universiteit leuven, Katholieke Universiteit Leuven.

Martin, M., Chopard, B., and Albuquerque, P. (2002). Formation of an ant cemetery: Swarm intelligence or statistical accident? *Future Gener. Comput. Syst.*, 18(7):951–959.

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

Mettu, R. R. (2002). *Approximation Algorithms for NP-hard Clustering Problems*. PhD thesis. AAI3099495.

Middendorf, M., Reischle, F., and Schmeck, H. (2000). Information exchange in multi colony ant algorithms. In Rolim, J., editor, *Parallel and Distributed Processing*, volume 1800 of *Lecture Notes in Computer Science*, pages 645–652. Springer Berlin Heidelberg.

Middendorf, M., Reischle, F., and Schmeck, H. (2002). Multi colony ant algorithms. *Journal of Heuristics*, 8(3):305–320.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition.

Monmarché, N. (1999). On data clustering with artificial ants. In Freitas, A., editor, *AAAI-99 & GECCO-99 Workshop on Data Mining with Evolutionary Algorithms: Research Directions*, pages 23–26, Orlando, Florida.

Monmarché, N., Guinand, F., and Siarry, P. (2010). *Artificial Ants From collective intelligence to real-life optimization and beyond*. Wiley.

Monmarché, N., Slimane, M., and Venturini, G. (1999a). Antclass: discovery of clusters in numeric data by an hybridization of an ant colony with the kmeans algorithm.

Monmarché, N., Slimane, M., and Venturini, G. (1999b). On Improving Clustering in Numerical Databases with Artificial Ants. In Floreano, D., Nicoud, J., and Mondala, F., editors, *5th European Conference on Artificial Life (ECAL'99)*, volume 1674 of *Lecture Notes in Artificial Intelligence*, pages 626–635, Swiss Federal Institute of Technology, Lausanne, Switzerland. Springer-Verlag.

Montgomery, E. J. (2005). Solution biases and pheromone representation selection in ant colony optimisation. Technical report, Bond University, Bond, Australia.

Morrison, T. and Aickelin, U. (2002). An artificial immune system as a recommender system for web sites. In *Proceedings of the 1st International Conference on Artificial Immune Systems (ICARIS 2002)*.

Mouhoub, M. and Wang, Z. (2008). Improving the ant colony optimization algorithm for the quadratic assignment problem. In *IEEE World Congress on Computational Intelligence, 2008. CEC 2008.*, pages 250–257.

Murtagh, F. (1984). Structure of hierarchic clusterings: implications for information retrieval and for multivariate data analysis. *Inf. Process. Manage.*, 20(5-6):611–617.

Nakahori, I. (2000). Management for emergent properties in the research and development process. In *Proceedings of the 2000 IEEE Engineering Management Society, 2000.*, pages 491–496.

Nesamalar, E. and Chandran, C. P. (2012). Genetic clustering with bee colony optimization for flexible protein-ligand docking. In *2012 International Conference on Pattern Recognition, Informatics and Medical Engineering (PRIME)*, pages 82–87.

Ngenkaew, W., Ono, S., and Nakayama, S. (2008a). The deposition of multiple pheromones in ant-based clustering. *International Journal of Innovative Computing, Information and Control*, 4(7):1349–4198.

Ngenkaew, W., Ono, S., and Nakayama, S. (2008b). Pheromone-based concept in ant clustering. In *ISKE 2008. 3rd International Conference on Intelligent System and Knowledge Engineering, 2008.*, volume 1, pages 308–312.

Niknam, T. and Amiri, B. (2010). An efficient hybrid approach based on pso, aco and *k*-means for cluster analysis. *Applied Soft Computing*, 10(1):183–197.

Niknam, T., Amiri, B., Olamaei, J., and Arefi, A. (2009). An efficient hybrid evolutionary optimization algorithm based on pso and sa for clustering. *Journal of Zhejiang University Science*, 10(4):512–519.

Nikolić, M. and Teodorović, D. (2013). Empirical study of the bee colony optimization (bco) algorithm. *Expert Systems with Applications*.

Norouziy, M., Fleety, D. J., and Salakhutdinov, R. Hamming Distance Metric Learning. In *NIPS 2012*.

Oleynik, A., Subbotin, S., and Oleynik, A. (2010). Bee colony optimization for clustering. In *2010 International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science (TCSET),*, pages 286–286.

Ono, O. (2009). Dna computing models and practice. In *Proceedings of the 2009 Third Asia International Conference on Modelling & Simulation*, AMS '09, pages 1–2, Washington, DC, USA. IEEE Computer Society.

Otero, F., Freitas, A., and Johnson, C. (2008). c Ant-Miner: An Ant Colony Classification Algorithm to Cope with Continuous Attributes. In Dorigo, M., Birattari, M., Blum, C., Clerc, M., Stützle, T., and Winfield, A., editors, *Ant Colony Optimization and Swarm Intelligence – 6th International Conference, ANTS 2008*, volume 5217 of *Lecture Notes in Computer Science*, pages 48–59. Springer Berlin / Heidelberg, Brussels, Belgium.

Otero, F., Freitas, A., and Johnson, C. (2009). Handling continuous attributes in ant colony classification algorithms. In *CIDM '09. IEEE Symposium on Computational Intelligence and Data Mining, 2009.*, pages 225–231.

Parpinelli, R., Lopes, H., and Freitas, A. (2001). An ant colony based system for data mining: applications to medical data. In Spector, L. and Goodman, E., editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 791–798, San Francisco, California. Morgan Kaufmann, San Francisco, CA.

Parpinelli, R., Lopes, H., and Freitas, A. (2002a). An Ant Colony Algorithm for Classification Rule Discovery. In Abbass, H., Sarker, R., and Newton, C., editors, *Data Mining: a Heuristic Approach*, pages 191–208. London: Idea Group Publishing.

Parpinelli, R., Lopes, H., and Freitas, A. (2002b). Data mining with an ant colony optimization algorithm. *IEEE Transactions on Evolutionary Computation*, 6(4):321–332.

Parpinelli, R., Lopes, H., and Freitas, A. (2002c). Mining Comprehensible Rules from Data with an Ant Colony Algorithm. In *Proceedings of SBIA*, pages 259–269.

Parpinelli, R., Lopes, H., and Freitas, A. (2005). Classification-Rule Discovery with an Ant Colony Algorithm. In Khosrow-Pour, M., editor, *Encyclopedia of Information Science and Technology*, volume 1, chapter Classification-Rule Discovery with an Ant Colony Algorithm, pages 420–424. Information Science Reference.

Pavan, K. K., Rao, A. A., Rao, A. V. D., and Sridhar, G. R. (2012). Robust seed selection algorithm for k-means type algorithms. *CoRR*, abs/1202.1585.

Pingdom (2013). Royal pingdom (2013, jan 16). 2012 internet in numbers. http://royal.pingdom.com/2013/01/16/internet-2012-in-numbers/. [Online; accessed Sept-2013].

Poli, R., Kennedy, J., and Blackwell, T. (2007). Particle swarm optimization. *Swarm Intelligence*, 1(1):33–57.

Powell, S. and Clark, E. (2004). Combat between large derived societies: A subterranean army ant established as a predator of mature leaf-cutting ant colonies. *Insectes Sociaux*, 51(4):342–351.

Prigogine, I., Stengers, I., and Prigogine, I. (1984). *Order out of chaos : man's new dialogue with nature / Ilya Prigogine and Isabelle Stengers ; foreword by Alvin Toffler*. Bantam Books Toronto ; New York, N.Y.

Rabanal, P., Rodríguez, I., and Rubio, F. (2008). Solving dynamic tsp by using river formation dynamics. In *Proceedings of the 2008 Fourth International Conference on Natural Computation - Volume 01*, ICNC '08, pages 246–250, Washington, DC, USA. IEEE Computer Society.

Rabanal, P., Rodríguez, I., and Rubio, F. (2010). Applying river formation dynamics to the steiner tree problem. In Sun, F., Wang, Y., Lu, J., Zhang, B., Kinsner, W., and Zadeh, L. A., editors, *IEEE ICCI*, pages 704–711. IEEE.

Rami, S. P. and Panchal, M. H. (2012). Comparative analysis of variations of ant-miner by varying input parameters. *International Journal of Computer Applications*, 60.

Ramos, V., M. F. and Pina, P. (2002). Self-organized data and image retrieval as a consequence of inter-dynamicsynergistic relationships in artificial ant colonies. hybrid intelligent systems. In *in Javier Ruiz-del-Solar, Ajith Abraham and Mario Köppen (Eds.), Frontiers in Artificial Intelligence and Applications, Soft Computing Systems Design, Management and Applications, 2nd Int. Conf. on Hybrid Intelligent Systems*, volume 87, pages 500–509. IOS Press.

Ramos, V. and Merelo, J. (2002). Self-organized stigmergic document maps: environments as a mechanism for context learning. In *Proceedings of the first Spanish Conference on Evolutionary and Bio-Inspired Algorithms*, pages 284–293.

Raykar, V. C., Yu, S., Zhao, L. H., Jerebko, A., Florin, C., Valadez, G. H., Bogoni, L., and Moy, L. (2009). Supervised learning from multiple experts: Whom to trust when everyone lies a bit. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 889–896, New York, NY, USA. ACM.

Roozmand, O. and Zamanifar, K. (2008). Parallel ant miner 2. In *Artificial Intelligence and Soft Computing – ICAISC 2008*, volume 5097 of *Lecture Notes in Computer Science*, pages 681–692. Springer Berlin / Heidelberg.

Santos, D. and Bazzan, A. (2009). A biologically-inspired distributed clustering algorithm. In *SIS '09. IEEE Swarm Intelligence Symposium, 2009.*, pages 160–167.

Schaeffer, S. E. (2007). Survey: Graph clustering. *Comput. Sci. Rev.*, 1(1):27–64.

Sendova-Franks, A. and Franks, N. (1995). Spatial relationships within nests of the antleptothorax unifasciatus(latr.) and their implications for the division of labour. *Animal Behaviour*, 50(1):121 – 136.

Shah-Hosseini, H. (2007). Problem solving by intelligent water drops. In *CEC 2007. IEEE Congress on Evolutionary Computation, 2007.*, pages 3226–3231.

Shanthi, D. and Amalraj, R. (2012). Collaborative artificial bee colony optimization clustering using {SPNN}. *Procedia Engineering*, 30(0):989–996. International Conference on Communication Technology and System Design 2011.

Sharma, A. and Omlin, C. (2009). Performance Comparison of Particle Swarm Optimization with Traditional Clustering Algorithms used in Self-Organizing Map. *International Journal of Computational Intelligence*, 5(1):1–12.

Shelokar, P., Jayaraman, V., and Kulkarni, B. (2004a). An ant colony approach for clustering. *Analytica Chimica Acta*, 509(2):187–195.

Shelokar, P. S., Jayaraman, V. K., and Kulkarni, B. D. (2004b). An ant colony classifier system: application to some process engineering problems. *Computers and Chemical Engineering*, 28:1577–1584. In Press.

Shi, L., Hao, J., Zhou, J., and Xu, G. (2004). Short-term generation scheduling with reliability constraint using ant colony optimization algorithm. In *Intelligent Control and Automation, 2004. WCICA 2004. Fifth World Congress on*, volume 6, pages 5102–5106 Vol.6.

Shukran, M. A. M., Chung, Y. Y., Yeh, W.-C., Wahid, N., and Zaidi, A. M. A. (2011). Artificial bee colony based data mining algorithms for classification tasks. *Modern Applied Science*, 5(4).

Silver, D. (2011). Machine lifelong learning: Challenges and benefits for artificial general intelligence. In Schmidhuber, J., Thórisson, K. R., and Looks, M., editors, *Artificial General Intelligence*, volume 6830 of *Lecture Notes in Computer Science*, pages 370–375. Springer Berlin Heidelberg.

Sneath, P. H. A. and Sokal, R. R. (1973). *Numerical Taxonomy. The Principles and Practice of Numerical Classification*. Freeman.

Solnon, C. (2001). Ants can solve constraint satisfaction problems. *IEEE Transactions on Evolutionary Computation*, 6:347–357.

Steinhaus, H. (1956). Sur la division des corp materiels en parties. *Bull. Acad. Polon. Sci*, 1:801–804.

Stützle, T. (1998). Parallelization strategies for ant colony optimization. In *Proceedings of PPSN-V, Fifth International Conference on Parallel Problem Solving from Nature*, pages 722–731. Springer-Verlag.

Stützle, T. and Hoos, H. (2000). $MAX - MIN$ Ant System. *Future Generation Computer Systems*, 16(8):889–914.

Sun, Z. G. and Teng, H. F. (2002). An ant colony optimization based layout optimization algorithm. In *TENCON '02. Proceedings. 2002 IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering*, volume 1, pages 675–678 vol.1.

Talbi, E.-G., Roux, O., Fonlupt, C., and Robillard, D. (1998). Parallel ant colonies for combinatorial optimization problems. In *in J. Rolim et al. (Eds.) Parallel and Distributed Processing, 11 IPPS/SPDP'99 Workshops, LNCS 1586*, pages 239–247. Springer.

Tan, S. C., Ting, K. M., and Teng, S. W. (2006). Reproducing the results of ant-based clustering without using ants. In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pages 1760–1767.

Tan, S. C., Ting, K. M., and Teng, S. W. (2011). A general stochastic clustering method for automatic cluster discovery. *Pattern Recogn.*, 44(10-11):2786–2799.

Teodorović, D. and Dell'Orco, M. (2005). Bee colony optimization–a cooperative learning approach to complex transportation problems. *Advanced OR and AI Methods in Transportation: Proceedings of 16th Mini–EURO Conference and 10th Meeting of EWGT (13-16 September 2005).–Poznan: Publishing House of the Polish Operational and System Research*, pages 51–60.

Teodorovic, D., Lucic, P., Markovic, G., and Orco, M. D. (2006). Bee colony optimization: principles and applications. In *Neural Network Applications in Electrical Engineering, 2006. NEUREL 2006. 8th Seminar on*, pages 151–156. IEEE.

Thangavel, K. and Jaganathan, P. (2007). Rule mining algorithm with a new ant colony optimization algorithm. In *2007. International Conference on Conference on Computational Intelligence and Multimedia Applications*, volume 2, pages 135–140.

Thangavel, K., Karnan, M., Sivakumar, R., and Kaja Mohideen, A. (2005). Ant Colony System for Segmentation and Classification of Microcalcification in Mammograms. *The International Journal of Artificial Intelligence and Machine Learning*, V - III.

Thengade, A. and Dondal, R. (2012). Article: Genetic algorithm - survey paper. *IJCA Proceedings on National Conference on Recent Trends in Computing*, NCRTC(5):25–29. Published by Foundation of Computer Science, New York, USA.

Theodoridis, S. and Koutroumbas, K. (2006). *Pattern Recognition, Third Edition*. Academic Press, Inc., Orlando, FL, USA.

Tsai, C. F., Tsai, C. W., Wu, H. C., and Yang, T. (2004). Acodf: a novel data clustering approach for data mining in large databases. *J. Syst. Softw.*, 73(1):133–145.

Tsutsui, S. and Liu, L. (2007). Solving quadratic assignment problems with the cunning ant system. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pages 173–179.

Tsutsui, S., Pelikan, M., and Ghosh, A. (2005). Performance of Aggregation Pheromone System on Unimodal and Multimodal Problems. In *The IEEE Congress on Evolutionary Computation, 2005 (CEC2005)*, volume 1, pages 880–887.

Van Der Merwe, D. W. and Engelbrecht, A. (2003). Data clustering using particle swarm optimization. In *CEC '03. The 2003 Congress on Evolutionary Computation, 2003.*, volume 1, pages 215–220 Vol.1.

van Rijsbergen, C. J. (1979). *Information Retrieval*. Butterworth.

Vandecruys, O., Martens, D., Baesens, B., Mues, C., Backer, M. D., and Haesen, R. (2008). Mining software repositories for comprehensible software fault prediction models. *Journal of Systems and Software*, 81(5):823 – 839. Software Process and Product Measurement.

Viet, N. H., Vien, N. A., Lee, S., and Chung, T. (2008). Obstacle avoidance path planning for mobile robot based on multi colony ant algorithm. In *Proceedings of the First International Conference on Advances in Computer-Human Interaction*, ACHI '08, pages 285–289, Washington, DC, USA. IEEE Computer Society.

von Frisch, K. (1993). *The dance language and orientation of bees*. Belknap Press. Harvard University Press.

Wan, M., Wang, C., Li, L., and Yang, Y. (2012). Chaotic ant swarm approach for data clustering. *Appl. Soft Comput.*, 12(8):2387–2393.

Wang, J., Tu, A., and Huang, H. (2012). An ant colony clustering algorithm improved from {ATTA}. *Physics Procedia*, 24, Part B:1414–1421. International Conference on Applied Physics and Industrial Engineering 2012.

Wang, J. B. and Wang, W. (2008). Research on aco with multiple nests cooperation and its application on narrow tsp. In *BICTA 2008. 3rd International Conference on Bio-Inspired Computing: Theories and Applications, 2008.*, pages 143–148.

Wang, X., Yang, J., Teng, X., Xia, W., and Jensen, R. (2007). Feature selection based on rough sets and particle swarm optimization. *Pattern Recognition Letters*, 28(4):459 – 471.

Ward Jr, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244.

Watada, J. and binti Abu Bakar, R. (2008). Dna computing and its applications. In *ISDA '08. Eighth International Conference onIntelligent Systems Design and Applications, 2008.*, volume 2, pages 288–294.

Wei, W. and Ji, M. (2011). Spatiotemporal modeling of 2000=2009 financial development in yangtze river delta using support vector machine and clustering analysis. In *2011 19th International Conference on Geoinformatics*, pages 1–5.

Weinstein, J. N. and Deyo, R. A. (2000). Clinical research issues in data collection. *SPINE*, 25(24):3104–3109.

Welch, P. G., Kemeny, Z., Ekart, A., and Ilie-Zudor, E. (2012.). "application of model-based prediction to support operational decisions in logistics networks". In *In Proc. AILog, ECAI*, pages 25–31.

Worall, M. (2011). Homeostasis in nature: Nest building termites and intelligent buildings. *Intelligent Buildings International*, 3(2):87–95.

Wu, H. and Sun, K. (2012). A simple heuristic for classification with ant-miner using a population. In *2012 4th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, volume 1, pages 239–244.

Wu, L. and Ren, A. (2008). Urban fire risk clustering method based on fire statistics. *Tsinghua Science and Technology*, 13(S1):418–422.

Xiong, Z., Chen, R., Zhang, Y., and Zhang, X. (2012). Multi-density dbscan algorithm based on density levels partitioning. *Journal of Information and Computational Science*, 9(10):2739–2749.

Xue-chun, L., Sen-fa, C., and Liu-Yan (2007). The study of small enterprises credit evaluation based on incremental antclust. In *IEEE International Conference on Grey Systems and Intelligent Services, 2007. GSIS 2007.*, pages 294–298.

Yadav, C., Wang, S., and Kumar, M. (2013). Algorithm and approaches to handle large data- a survey. *CoRR*, abs/1307.5437.

Yang, S. and Zhang, Y. (2007). Key point based data analysis technique. In Huang, D.-S., Heutte, L., and Loog, M., editors, *Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence*, volume 4682 of *Lecture Notes in Computer Science*, pages 444–455. Springer Berlin Heidelberg.

Yang, X. S., Cui, Z., Xiao, R., Gandomi, A. H., and Karamanoglu, M. (2013). *Swarm Intelligence and Bio-Inspired Computation: Theory and Applications*. Elsevier Science & Technology Books.

YouTube (2013). Youtube offical statistics (2013, oct 10). http://www.youtube.com/yt/press/statistics.html. [Online; accessed Oct-2013].

Zaharie, D. and Zamfirache, F. In *Congress on Evolutionary Computation*, pages 2395–2401. IEEE.

Zahn, C. T. (1971). Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Transactions on Computers,*, 20(1):68–86.

Zapfel, G. and Wasner, M. (2002). Planning and optimization of hub-and-spoke transportation networks of cooperative third-party logistics providers. *International Journal of Production Economics*, 78(2):207–220.

Zhang, P. and Lin, J. (2010). An adaptive heterogeneous multiple ant colonies system. In *(ISME), 2010 International Conference of Information Science and Management Engineering*, volume 1, pages 193–196.

Zhou, A., Qu, B.-Y., Li, H., Zhao, S.-Z., Suganthan, P. N., and Zhang, Q. (2011). Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation*, 1(1):32 – 49.

Zhu, Y.-F. and Tang, X.-M. (2010). Overview of swarm intelligence. In *2010 International Conference on Computer Application and System Modeling (ICCASM)*, volume 9, pages V9–400–V9–403.

Zong, X., Xiong, S., Fang, Z., and Li, Q. (2010). Multi-ant colony system for evacuation routing problem with mixed traffic flow. In *IEEE Congress on Evolutionary Computation*, pages 1–6. IEEE.