SWARM INTELLIGENCE: A PRACTICAL APPROACH

An Investigation into the Implementable Practicality of Statistical Classification Based on Biological Systems

MAY 5, 2019

AUTHOR: LUKE MCCANN
The University of Huddersfield

Contents

Abstract 3
1 Introduction 3
1.1 The ACO Metaheuristic 4
1.2 Project Plan ..... **Error! Bookmark not defined.**
1.2.1 Tools ..... **Error! Bookmark not defined.**
1.2.2 Motivation 5
2 Introducing Swarms 5
2.2 Metaheuristics 5
2.2.1 Problem Independence 6
3 Origins of Swarm Intelligence 6
3.1 Swarm Intelligent Systems 6
3.1.1 Positive Feedback 7
3.1.2 Negative Feedback 7
3.1.3 Probabilistic Fluctuations 7
3.1.4 Multiple Interaction 8
4 Self-Organising Systems 8
5 Ant Colony Optimisation 8
6 Artificial Ants 9
7 Solution Construction 10
7.1 Local Pheromone Update 10
8 ACO: Artificial Double Bridge 11
8.1 Initial Test 13
8.2 Complex TSP Test 14
8.3 Complex TSP Experiment1 14
8.4 Optimising Parameters 15
8.5 TSP Doubled Population 16
8.6 TSP Increased Evaporation and Desirability (IED) 16
Fig. 15 IED Experiment Results 17
9 Clustering 17
9.1 ACA Basic-Clustering Experiment 19
10 Artificial Bee Colony (ABC) 21
11 K-means 21
12 Centroid Based Ant Clustering 22

12.1 Algorithm 22
12.2 AC Iris Experiment 23
13 Classification 25
14 Fuzzy Logic 27
15 Conclusion 27
Ethical Review Form 30
Appendices ..... **Error! Bookmark not defined.**
References 31

Abstract

Solutions to some of the most complex problems faces today have already been solved. Swarm Intelligence (SI) is an emerging field which has gained traction in the last decade as parallelisation needs increase. As datasets become larger, companies and researchers need more processing power to analyse these sets. This study aims to investigate some of the more popular implementations of these algorithms, presenting the initial idea behind the algorithms and comparing and analysing the algorithms to other similar algorithms regarding the classification problem. Alternative solutions to these algorithms are then proposed to enhance the efficiency, reliability and overall effectiveness of the algorithm.

1 Introduction

In recent years artificial intelligence (AI) and machine learning (ML) have gained renewed interest among the scientific community, after an initial dark period within the industry of research and academia regarding intelligent computational systems. As such, many researchers have acquired an accelerating interest in the design of natural artificial systems, the areas of which include; metaheuristics (MH) neural networks (NN's) and evolutionary computing (ECO). Autonomous systems offer an advantage to many industries, allowing for machines to explore zones or perform tasks which hold a danger to human health, or simply with greater levels of accuracy or yielding higher performance.

As the need for higher performance grows to complete tasks ever growing in complexity parallelisation is key, generating a surge of interest around the field of Swarm Intelligence (SI). SI is defined by the emergent complex self-organisational behaviours of autonomous agents who as a single agent sport a low level of intelligence (Tosun, 2014). While much of the research in SI is highly theoretical, there are a variety of nature-inspired algorithms which have been proposed over the last decade.

Many of such algorithms are defined in the area of MH's, an area focused on high level procedures designed to search for an adequate solution to some optimisation problem given incomplete datasets (Celso, 2006). MH's focus on solving NP-hard problems, that of which once a solution is found it may be applied to any problem of a similar nature (Talbi, 2009). While globally an optimal solution for these problems is not guaranteed to be discovered these algorithms aim to provide acceptable solutions in a satisfactory time frame using stochastic optimisation techniques. Some of the most successful algorithms proposed include:

- Ant Colony Optimisation (ACO)
- Particle Swarm Optimisation (PSO)
- Artificial Bee Clustering (ABC)
- Ant Colony Algorithm (ACA)

1.1 The ACO Metaheuristic

As an emerging area in AI, SI was first inspired by the biological systems of social insects. The most famous experiment contributing to the founding of this area is that of the Double Bridge Experiment by Goss et al (1989). Goss et al's findings from experiments with real ants demonstrate how a system governed by simplistic rules can evolve complex emergent behaviours naturally from the use of stigmergic indirect communication, such as the ants selfcatalysing shortest path optimisation.

Inspired by the findings of Goss (1989), Dorigo (1999) proposed an algorithm to solve the infamous Travelling Salesman Problem in 1992, this was known as the Ant System (AS). AS has since been redefined in 1999 as the ACO metaheuristic by Dorigo & Gianni (1999) to unify the workings of ACO allowing a simpler implementation for solving discrete optimisation problems (Dorigo & Gianni, 1999). ACO is most suited to problems in which properties of the environment change over time, meaning that the optimisation process must be able to adapt with these changes, or where computational architecture is distributed.

Over the years there have been several implementations of ACO varying in success rates. These problems include but are not exclusive to:

- The Job-Shop Scheduling Problem
- The Vehicular Routing Problem
- Load Balancing in Cloud Based Systems
- Assignment Problems (Generic Assignment Problem (GAP), Quadratic Assignment Problem (QAP)
- Image Segmentation
- Data Mining (Clustering and Feature Selection)

1.2 Motivation

Ants have always been intriguing creatures; they are small yet can accomplish comparatively mammoth tasks. The idea of creatures, of which possess only the most basic of intelligence, are able to collaboratively build large complex architectural structures and always seem to have a sense of order has always been fascinating. SI offers a unique insight into the microscopic world, utilising the power of natural instincts possessed by some of the most organised creatures on the planet has the potential to solve some of the largest known problems. Applying methodologies as such provides machines with an imitative intelligence which can be exploited to deal with a range of adaptive issues. Adaptive solutions as such are highly sought after in todays world, from training neural networks to calculating purchasing patterns, the natural world has solved many of these issues.

2 Introducing Swarms

Swarm intelligence (SI) is an evolving area of AI related to the introduction of evolutionary computing and genetic algorithms. SI takes inspiration from nature to discover solutions to problems which the biological world has already solved. The problems tackled in SI are usually those of a non-deterministic polynomial time hardness (NP-hardness). This implies that an algorithm for solving one of these problems can effectively be applied to solving any problem of a non-deterministic nature.

Much of the research in SI has taken place over the last two decades, as interest in search optimisation, load balancing and clustering has increased due to dealing with large amounts of data. As such, much of the research is highly theoretical with some implementations as computational experiments.

This study aims to explore the implementable practicality of SI with a focus on algorithms inspired by social insect behaviours, specifically focusing on the clustering and data-mining capabilities.

2.2 Metaheuristics

In order to understand the potential of SI it is important to first define the overall function of a metaheuristic algorithm (MHA), where they fit and the advantages which they provide over traditional methods.

An MHA functions at a high level. MHA's are designed to search for an acceptable solution within a given search space usually represented in a graphical space. MH's are able to function adequately, providing acceptable localised solutions, without the need for complete datasets (Glover & Kochenberger, 2003).

Comparatively, while algorithms in other methods of evolutionary computing provide optimal solutions, MH's do not guarantee a solution which meets the global optima and may settle for one which meets the local optima (Jaszkiewicz & Branke, 2008), thus they are best utilised in confronting complex combinatorial problems.

Many MH's implement a form of stochastic optimisation, defining a global minimum/maximum of statistical functions. As such they work on solving and improving candidate problems iteratively (Ombach, 2015).

2.2.1 Problem Independence

The nature of MH's is independent from the problems to which they are applied. Instead of providing direct solutions, they instead provide a set of strategies for which to develop heuristic optimisation algorithms (Sörensen & Glover, 2013)

3 Origins of Swarm Intelligence

Originally introduced in 1989 via cellular robotic systems (Garg, Gill, Rath & Amardeep, 2009), the first cellular systems consisted of a finite number of autonomous agents operating within a finite nth dimensional search space with limited local communicative capabilities to complete some predefined global task (Wang and Benig, 1989).

Bonabeau, Dorigo and Theraulaz (1997) argue that the term SI creates unnecessary restrictions and redefine the definition of "Swarm intelligence, 1999" to encapsulate any algorithmic designs or problem-solving devices which have been inspired by the social behaviours of insect colonies. This argument also became supported by Flake (1999) who defines SI as that of the emergent complex behaviours which occur from groups of simple autonomous agents.

Most algorithms operating under SI operate as a collective behaviour of decentralised selforganisation, as such there are no leaders and no global plan to follow. Humans tend to imagine swarms of bees, flies or other insects; however, this definition has been applied to a wide variety of natural behaviours in many differing organisms: chickens, flocks of birds, schools of fish, colonies of ants, hives of bees, cells.

From these, arise instinctive globally complex behaviours from seemingly simplistic rule sets, for instance the complex manoeuvres performed for predatorial avoidance. These emergent behaviours are unsupervised, each agent has a probabilistic behaviour and a local perception of their environment. In a system governed by such local rules accompanied by the interactions each agent shares with one another elegant solutions emerge to complex problems (Karaboga & Akay, 2009). While these behaviours appear deliberate from a global perception, this outcome is unknown to the agent at the local level.

3.1 Swarm Intelligent Systems

Swarm Intelligence Systems (SIS) utilize this local interaction in creative displays; a school of fish must stay close together for safety yet be able to avoid collisions whilst disbanding and re-grouping to avoid obstacles. According to Bonabeau, Dorigo and Theraulaz (1999) these behaviours can be attributed to four characteristics of a swarm:

- Positive Feedback
- Negative Feedback
- Probabilistic Fluctuations
- Multiple Interaction

3.1.1 Positive Feedback

Positive feedback is necessary to promote the prospect of these collective behaviours. It is used to reinforce fragments of good solutions that contribute to the overall goal. Positive feedback is usually accomplished via an entropic loop. These loops push the system away from equilibrium increasing entropy. In social insects this feedback loop is accomplished via the use of pheromones and acts as a self-catalysing system.

Ants begin this behaviour via foraging. As the nest is invoked scout ants begin with random movements seeking out the nearest food source. As the scout's wander, they deposit pheromone along the trail, this pheromone varies in strength dependent on if food has been discovered and sometimes on the quality of the food source. This autocatalytic process attracts more ants to the trail who in turn strengthen the trail further (Bundgaard, Damgaard, Dacara & Winther, 2002).

3.1.2 Negative Feedback

Negative feedback acts as a counterweight to positive feedback, modelled as the evaporation of pheromone along the ant's trail. Ants must be continuously moving over a path to reinstantiate the pheromone as it evaporates, otherwise the trail will evaporate over time (Bundgaard, Damgaard, Dacara & Winther, 2002).

Negative feedback prevents the solutions becoming stuck in sub-optimal solutions; since solutions are able to be forgotten over time ants can get out of cyclic loops and exhausted supplies will no longer be visited.

Due to the time it takes for the ants to travel, longer paths have a higher probability of being forgotten in favour of shorter ones, due to the higher frequency ants are able to pass over these paths in relativistic time.

3.1.3 Probabilistic Fluctuations

Self-organisational behaviours are reliant on random probability. In order to develop more creative solutions some randomness is essential. Innovation and creativity evolve from random fluctuations; as how particles are able to pop in and out of existence via quantum fluctuations (Muthiah-Nakarajan, Mithra Noel, 2016), swarms are able to exploit probabilistic fluctuations to discover new pathways to food which may be closer to the nest than that which is already known (Bonabeau, Dorigo & Theraulaz, 1999).

3.1.4 Multiple Interaction

Shared interaction is the final requirement for self-organising behaviours to emerge. Agents must be able to make use of their own experiences and that of other agents to improve on solutions. Ants use of pheromones is a form of stigmergic communication, by altering their environment they are able to relay data across both space and time (Bundgaard, Damgaard, Dacara & Winther, 2002).

4 Self-Organising Systems

Systems which exhibit organised behavioural patterns are often assumed to occur by intelligent design. In such a system the actions performed in an order such that global intelligent behaviour appears pre-meditated (Flake, 1999).

The theory of emergent behavioural organisation transpires from studies in ecology extending into computer science, the core concepts of which can be applied throughout all systems of a self-organising nature and can even apply at the universal scale (Renyue Cen, 2014). Organisation itself is defined via measurements, the higher the entropy of a system the more information which is required to describe the system.

In order to sustain itself over time, a system must acquire the ability to change its operational environment; many self-organising systems are able to remedy damage to continue operation. An example of this acknowledged by Dutta (2018) known as the network effect, predicts that as more users join a social network, the more useful the network will become up until reaching a critical mass. However, a single user removing themselves from the network is soon replaced or re-routed preserving the overall functionality of the social network itself. This network grows and adapts as any other self-organising system via a positive feedback loop.
Diversity and innovation amongst an initial population are necessary to promote the probability of self-emergent behaviour, in systems of agents with low-levels of intelligence this can be modelled by integrating inherent randomness as part of the overall local automata.

5 Ant Colony Optimisation

Inspired by the findings of Goss (1989) ant colony optimisation (ACO) was first proposed by Dorigo (1992). Based on the natural foraging behaviours of ant colonies, ACO is a multiagent system best suited to problems of a discrete NP-hardness. ACO is adaptive and can deal with systems which change over time, over the years many variations of ACO have been proposed, however there is no globally unified process for implementing the algorithm, though a design pattern was proposed by Dorigo and Gianni (1998).

6 Artificial Ants

Artificial ants (AA's) are the key concept in ACO, modelled after biological ants. The AA's work as a collective intelligence system enabling them to complete tasks much larger than a single ant can achieve.

All colony interactions are handled at the local level, modifying their environment via a pheromone matrix to relay information.

The iterative nature of AA's allows for improvement upon candidate solutions ensuring an adequate minimal cost is achieved for the problem at hand from an initial starting position (either the nest or a random node) (Dorigo, 1992). They build these solutions by applying positional transformations within a discrete environment. AA's may exploit heuristic information along with their own pheromone trails to improve upon solutions. AA's, however, only deposit pheromone upon acquiring a complete solution.

Evaporation of the pheromone is modelled as a means of cyclic prevention, with AA's evaporation taking place at the end of the cycle, before candidate solutions are updated (Dorigo & Stützle, 2004).

AA's begin with initial probabilities which may vary dependent on the set weighting, the idea being that these parameters are tweaked to find an optimal solution. Often the initial algorithm is instantiated with an equal probability of 0.5 when dealing with shortest path optimisation (Fig. 1).

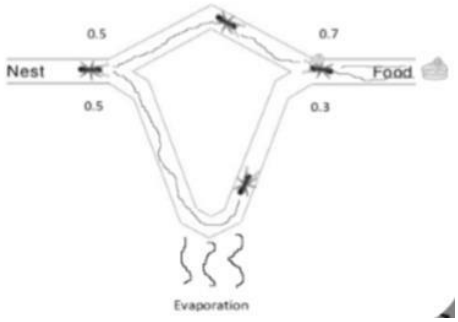

Fig. 2 ACO Probability Traversal
Based on (Goss, 1989)

The ant traversing the shortest path will always arrive before the ant taking the longer path, therefore the probability of selecting the shorter path upon the first return trip is significantly increased (Goss, 1989).

7 Solution Construction

AA's have a higher probability of following stronger pheromone trails. Whilst they still may choose a weaker trail a weighted probability ensures more ants will select the solution with additional pheromone. ACO solution construction can be modelled via a simple equation (Fig. 2) where; *P* denotes the amount of pheromone with *i and j* nodes between which any ant (*k*) may transpose.

The parameter for regulating the influence of pheromone is characterised as *T* with alpha (α) and beta (β) representing the heuristic values controlling the desirability of an edge (*n*) (Shekhawat, Poddar & Boswal 2009). The calculation overall calculates the probability of a single component within the solution being selected.

$$\rho_{ij}^{k} = \frac{[t_{ij}]^{\alpha} * [n_{ij}]^{\beta}}{\sum l \in N_{l}^{k}[t_{il}]^{\alpha} * [n_{il}]^{\beta}}$$

Fig. 3 ACO Solution Equation
(Shekhawat, Poddar & Boswal 2009)

7.1 Local Pheromone Update

After each successfully constructed solution the pheromone update function runs. AA's utilise pheromones varying the levels of pheromone deposited on a path dependent on the overall cost of the solution, where the pheromone on any particular edge is denoted as T_{ij} when multiplied with delta (Δ) this gives the quantity of deposited pheromone (Fig. 3) which in turn can be derived by a secondary equation (Fig. 4).

$$T_{ij} = (X - \rho)T_{ij} + \Delta T_{ij}$$

`

Fig. 4 Pheromone Update Equation
(Shekhawat, Poddar & Boswal, 2009)

This equation represents a superior implementation comparative to the original AS system. The inclusion of evaporation, conveyed as (*P*), is not present in the original solutions proposed by Dorigo (1992).

$$\Delta T_{ij}^{k} = \begin{cases} X/C_k & \text{If } ij \text{ was used by agent} \\ 0 & \text{Else} \end{cases}$$

$$T_{ij} \leftarrow T_{ij} + \sum_{k=1}^{m} \Delta T_{ij}^{k}$$

Fig. 5 Pheromone Update Equation 2
(Shekhawat, Poddar & Boswal, 2009)

The total cost of constructing the solution can be described via C^k, while *X* is typically set to 1 the parameter can be tweaked to an optimal value via trial and error to adjust the amount of deposited pheromone, this is useful in some use cases which require the value to be adjusted via a value (e.g. the length of the agent's journey).

After applying the local pheromone update rule, the global rule dictates the point at which the pheromone evaporates from the path. This is established after each agent has successfully applied the local pheromone update rule (Brownlee, 2011)

8 ACO: Artificial Double Bridge

Before implementing the ACO algorithm a problem statement must be defined in a graphical format. This means that any dataset which is used must first be converted into a graphical search space before any implementation is possible. For this experiment, inspired by the original experiment performed with biological ants by Goss (1989), Matlab is used and the graphical data created using set values for each node.

The pseudocode must then be derived from the equations (Fig. 5); *tau* can be modelled as a percentage based parameter for controlling the amount of pheromone deposited, eta can be determined via the equation in which the array of edges is divided by 1, reversing this equation can also be used to discover the longest path.

```
for iteration -> n
    set initial population;
    set initial pheromone concentration;
    initialise pheromone matrix;
    calculate initial edge desirability;
    set evaporation rate;

    create_colony;
    calculate_fitness;
    update_elite_fitness;
    update_pheromone_matrix;
end
```

Fig. 6 ACO Compact Pseudocode

To initialise the pheromone matrices *tau* is set to a matrix containing the mean distance of all edges in the graph, alpha and beta are assumed to be 1.

Firstly, the main loop must iterate up until iteration *n* within this loop the first population is initialised beginning from a random node selecting the next node traversal via a roulette wheel operation (Fig. 6) (Rebekka & Malarkodi, 2016).

```
for n -> pop_density
    select initial node;
    add ant to node;

    for n+1 -> max_nodes
        previous_node = 0;
        calculate traversal probabilities;
        select next node from probability;
    end
end
```

Fig. 7 ACO CreateColony Pseudocode

After the probabilities are calculated and the AA's have attained a complete solution the colony fitness is evaluated holding the elite ant back as the optimal solution for this iteration.

Lastly, the pheromone matrix is updated where *tau* is equivalent to the pheromone matrix and *rho* is set to equal the evaporation rate equation.

Similar to Genetic Algorithms, the performance of the colony is evaluated via the use of a fitness function. This fitness function calculates the Euclidean Distance between each node and evaluates the shortest path as the highest fitness value (Fig. 7).

```
for n -> tour.length
    calculate_euclidean_distance(nodes);
end

set_fitness(ant(n));
if(ant.fitness < elite_fitness)
    elite_fitness = ant.fitness;
end
```

Fig.8 ACO FitnessFunction Pseudocode

Finally, the pheromone matrices can be updated via the pheromone update equation (Fig. 3, Fig. 8) and evaporation can then take place.

```
for n -> population_density
    for n -> number of nodes
        calculate tau for nodes +1/ant(n) fitness
    end
end
```

Fig. 9 ACO Pheromone Update Pseudocode

8.1 Initial Test

Testing the algorithm performs as expected using Matlab a simplistic graph is first created with an obvious shortest path. The AA's have a choice of a single longer path which may also allow every node to be traversed but at a higher cost. In this implementation the length of each path is the cost of said path, therefore taking longer to traverse (Fig. 9).

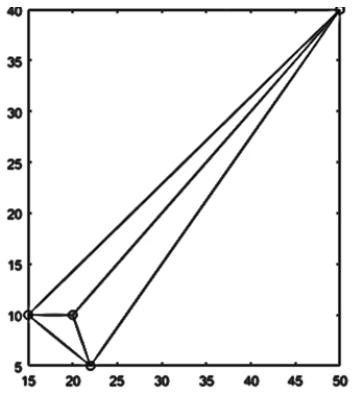

Fig. 9 Artificial Split-Branch (ASB)

The AA's are set to a population of 2 with the maximum number of iterations as 500, as this is a simple solution, the AA's are able to find an adequate solution in a reasonable timeframe even with a minimal initial population.

This solution takes only an average of 5 generations for the results to converge into an optimal shortest path solution (Fig. 10).

Generation	Shortest Path
1	101.3048
2	101.3046
3	100.3046
4	100.8506
5	100.8506
n….500	100.8506

Fig. 10 Basic ASB Results

8.2 Complex TSP Test

The TSP test is devised to increase the complexity of the problem the AA's need to solve, whilst retaining the same use-case scenario. The only change to the algorithm is the data supplied to the graph (Fig. 11) along with an initial population increase to 10.

Due to the increased complexity derived from the randomisation of values it is not possible to visually determine the shortest path from the graph alone. If this was a network in which an engineer wished to uncover the shortest routing path ACO would prove to provide an adequate solution in a much shorter timeframe than humanly possible.

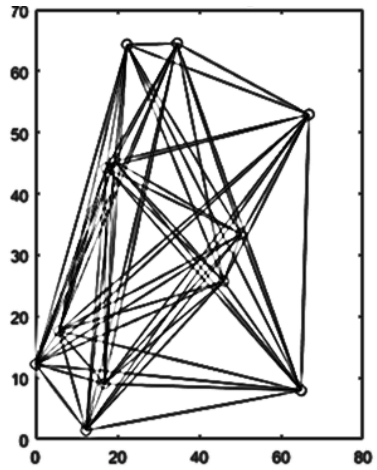

Fig. 11 ACO Simulated TSP Problem

When dealing with more complex problems ACO's parameters can be fine-tuned. Through experimentation it is possible to execute solution with earlier convergence than in the first execution. This can occur via additional parallelisation (increasing the population) or lowering/heightening the evaporation rates, alpha and beta may also be fine-tuned to deliver differing results.

8.3 Complex TSP Experiment1

Population:	10
Evaporation Rate:	5%
Initial Desirability:	1
Pheromone Level:	1

Fig. 12 Experiment1 Parameters

To clarify the algorithm scales, the initial parameters are kept from the ASB experiment (Fig. 12), the average convergence of the distance occurring around the 40th generation on average.

Generation	Shortest Path
1	322.3863
2..3	302.9756
4..5	284.3803
6..8	263.2433
9..26	255.5531
27..38	244.6762
39..500	238.585

Fig. 13 Expriment1 Results

While the algorithm takes many generations to execute in this circumstance, the overall time taken to execute to an optimal solution is around 10 seconds. While this is not ideal, as larger datasets would increase the complexity and therefore the run time of the algorithm, the algorithm does provide an adequate solution in a smaller time period than calculating the path manually.

8.4 Optimising Parameters

As mentioned previously, ACO allows for the adjustment of various parameters. To test the convergence rates a number of tests are conducted on the same TSP graph, with adjusted parameters. This allows the algorithm to be optimised allowing for better solutions to be found within less iterations.

8.5 TSP Doubled Population

Population:	20
Evaporation Rate:	5%
Initial Desirability:	1
Pheromone Level:	1
Generation	Shortest Path
1	326.236
2	279.7968
3..4	265.4357
5..7	261.6959
8..11	259.3213
12	259.0526
13..24	246.9188
25..40	241.9144
41..500	238.585

Fig.14 Experiment2 Results

While doubling the population increases the number of AA's traversing the graph, the average convergence remains the same. This is due to the fact that while doubling the population increases the number of ants to traverse the graph, the evaporation rates remain the same, implying that most of the AA's are still attracted to similar routes.

8.6 TSP Increased Evaporation and Desirability (IED)

An increase in population provides more parallelisation for solving the problem. This alone however is not enough to contribute to a faster convergence rate as the random probability of choosing a shorter path remains the same. By adjusting parameters to increase the initial desirability of each edge the longer edges become more apparent to the AA's sooner. By also adjusting the evaporation rate these longer paths are swiftly forgotten by the AA's lowering the convergence rate from an average generation of 40 to 4 (Fig. 15).

The increased evaporation and desirability (IED) experiment demonstrates how adjusting ACO parameters can greatly affect the performance of the algorithm. These parameters require adjustment dependent on the type of problem that is being faced along with the complexity of the problem.

Experiment1	
Population:	20
Evaporation Rate:	15%
Initial Desirability:	2
Pheromone Level:	1
Generation	Shortest Path
1..3	255.5005
4…500	238.585
Experiment2	
Population:	20
Evaporation Rate:	15%
Initial Desirability:	2
Pheromone Level:	1
Generation	Shortest Path
1	291.7974
2	267.2409
3	255.5005
4	246.9188
5…500	238.585
Experiment3	
Population:	20
Evaporation Rate:	15%
Initial Desirability:	2
Pheromone Level:	1
Generation	Shortest Path
1..2	255.5005
3	254.598
4	238.585

Fig. 15 IED Experiment Results

9 Clustering

Clustering is the process of partitioning sets of data into meaningful subclasses based on supplied features. It is used in many professions in the modern world. From research, to medical diagnostics and even business applications, clustering helps understand the patterns within large datasets in a visual format.

Clustering is an unsupervised learning problem, as such it is difficult to solve due to the possibility of incorrect or incomplete data. Clustering can also be computationally expensive to solve especially when applied to large datasets (Williams, 2015).

Many algorithms have been created to deal with the clustering problem, all of which rely on heuristic information, however, there is yet to be a generic solution to the clustering problem.

Ant-based clustering applies a variation of the ACO algorithm to the clustering problem in an attempt to discover classification patterns without supervision.

The improved Ant Colony Algorithm, (ACA) proposed by Deneubourg (1991) and improved upon by Lumer and Faieta (1994), presents one such method of applying the clustering methods using the behaviours of ants as a basis.

Much like ACO, ACA applies a positive feedback loop along with distributed parallelisation (Maimon & Rokach, 2010) and pheromone. Deneubourgs experiments performed in 1991 using Phaidole Pallidula ants illustrate the natural clustering behaviours performed by realworld ants. These species of ants cluster the dead into compact piles, it is this behaviour which ACA applies to AA's allowing for the clustering of data via ant colony methodologies.

ACA differs from ACO in application. It instead aims to discover clusters within sets of data by utilising the ACO concepts of stigmergic local communication to discover patterns and cluster data in groups of nearest neighbours (Fig. 16).

```
Procedure ACA
    Place every item X_i on a random cell of the grid;
    Place every ant k on a random cell of the grid unoccupied by ants;
    iteration_count ← 1;
    While iteration_count < maximum_iteration
        do
            for i = 1 to no_of_ants                    // for every ant
              do
                if unladen ant AND cell occupied by item X_i,
                then compute f(X_i) and P_pick-up(X_i);
                pick up item X_i with probability P_pick-up(X_i)
                else if ant carrying item xi AND cell empty,
                  then compute f(X_i) and P_drop(X_i);
                  drop item X_i with probability P_drop(X_i);
                end if
                    move to a randomly selected, neighboring and
                    unoccupied cell ;
            end for
            t ← t + 1
    end while
    print location of items;
end procedure
```

Fig. 16 ACA Psuedocode
(Maimon & Rokach, 2010)

The algorithm begins by placing every datum (*X*) on the graph along with every ant (*k*). The main loop then iterates over the colony checking if the randomly wandering AA's are holding a node. A function then checks the density of nearby objects along with the similarity regarding features. The AA then decides based on probability whether to pick the node up or drop the node dependent on if the ant had already held a node. Finally, the AA is moved to a randomly selected neighbouring location and the location of each node updated.

9.1 ACA Basic-Clustering Experiment

To utilise ACA it is important to first define the data. Contrary to ACO, ACA also requires the AA's to be placed at random within the search space. During the implementation process Matlabs *normrnd* function can be utilised to quickly generate dummy locations for the nodes.

```
function [] = dropAnts(antArr)

    for i = 1:colonySize
        running = true;
        while(running)
            xpos = ceil(rand*population);
            ypos = ceil(rand*population);
            if((xpos ~= 0 && ypos ~= 0) && antArr((xpos), (ypos)) == 0)
                antArr((xpos),(ypos)) = i;
                antPos(i,1) = xpos;
                antPos(i,2) = ypos;
                running = false;
            end
        end
    end
end
```

Fig. 17 ACA dropAnts

The dropAnts method (Fig. 17) simply populates the graph space with AA's, the same methodology can be applied to populate nodes simply exchanging the *antArr* matrix for that of the *dataArr* matrix and switching the positional parameters. The resultant graph shows the plotted data points (o) with the initially randomised positions of the AA's (*) (Fig. 18). All points plotted to the graph via Matplot are stored in matrices of zeros, any other number in the matrix is representative of an occupied location.

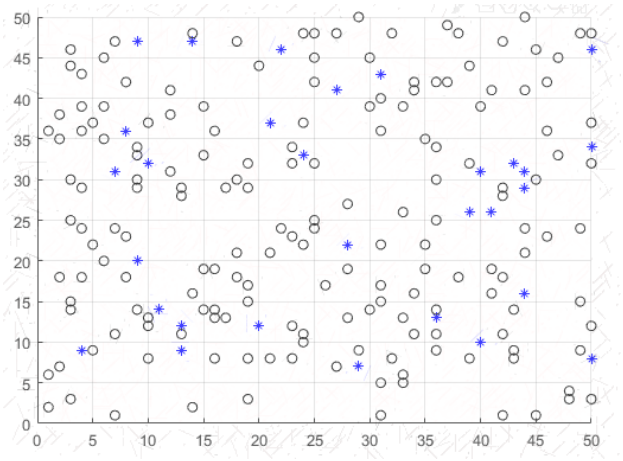

Fig. 18 Initial Nodes and AA's

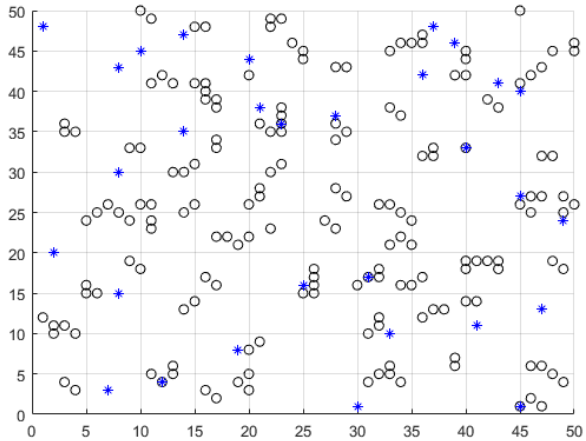

Fig. 19 ComplexCluster ACA: Iteration447

After a period of inherent randomness, the AA's must select a path from a pre-determined selection modelled via a 2d array containing *x* and *y* positional values.

The selected path is limited to a position which does not exceed the largest or smallest value of the colony, is not already occupied by other AA's. If the AA also happens to be carrying a node, it is unable to move to any position which contains a new node.
Ants carrying nodes are tracked via an alternative matrix. The distance between these ants and any other surrounding nodes is calculated via the Euclidean distance (Appendix 3), the probability of the AA picking up/dropping the node is then calculated via the pheromone levels and density of nodes/AA's within the given area, after which the movement matrix is updated providing all plausible areas of which the AA may next occupy. While measuring the success of ACO can be determined per generation via the distance which has been traversed, clustering is best measured visually via matrices data.

Tests with fewer nodes indicate larger populations of AA's increase performance slightly due to the additional parallelisation of movement and the increased likelihood of AA's wandering into either a node or pheromone.

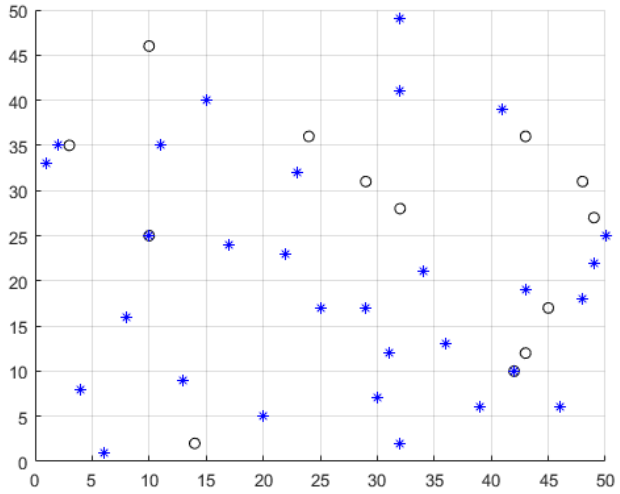

Fig. 20 Small Data ACA

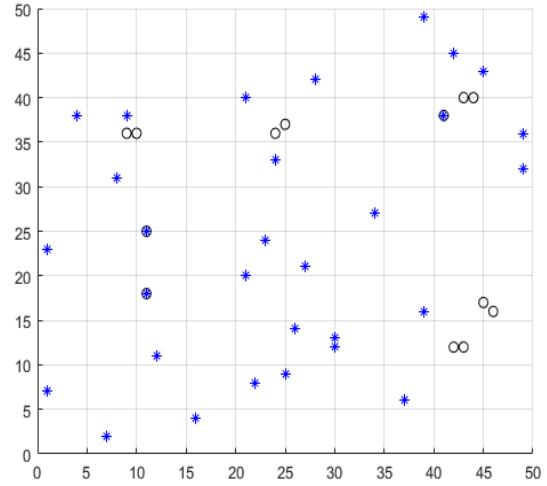

Fig. 21 Small Data ACA: Iteration 411

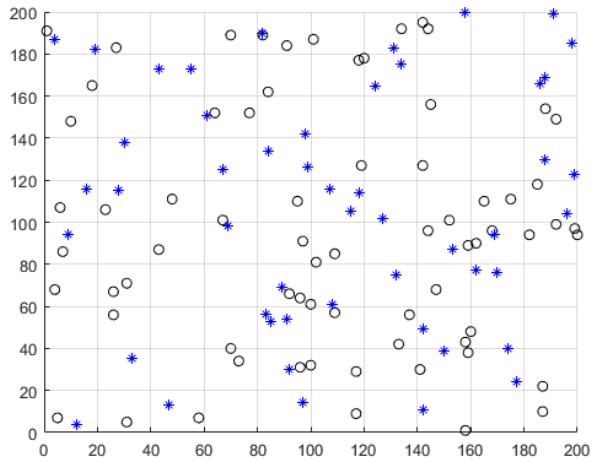

Fig. 22 High Ant Count Large Data

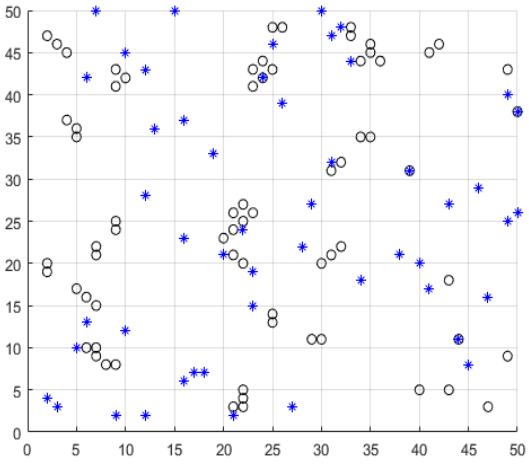

Fig. 22 High Ant Count Large Data Iteration: 403

10 Artificial Bee Colony (ABC)

Artificial bee colony (ABC), proposed by Dervis Karaboga (2005), is a novel approach to the clustering problem; taking inspiration from another social class of insect (bees). ABC is an optimisation algorithm, utilising the same control parameters and methodologies as ACO with the inclusion of additional globalisation and official role delegation. Artificial bees (AB's) are able to utilise different roles within the hive (D. Karaboga, G. Karaboga, Gorkemli & Ozturk, 2012), as such the algorithm tends towards a higher performance rate than ACO due to the global and local interactions between roles acquiring information form the environment at a faster rate.

ABC can be directly applied to the classification problem, as demonstrated by Karaboga & Ozturk in comparison with particle swarm optimisation (PSO) and a multilayer perceptron neural network (MLP). While MLP is shown to supersede ABC in terms of processing and reliability, ABC offers a greater generalisation possibilities to the generic clustering problem (Karaboga & Ozturk, 2011).

11 K-means

K-means, proposed by Stuart Lloyd (1957), is a centroid based partitional clustering algorithm taking a set of data points as input. A centroid is a point (*k*) initially selected at random to act as the centre of a cluster, each iteration of the k-means algorithm iterates through nodes (*x*) and calculates the Euclidean distance of x_i in regards to the location of *k*, *x* is then assigned to the cluster (*j*) which has the minimum distance to the centroid *k* (Chen, 2019). Whilst K-means is not bio-inspired it is the current most well known and popular algorithm for solving clustering problems due to its simplicity, reliability for adequate solutions and the speed at which it is able to process data. As this project is based on nature inspired algorithms the following k-means experiments are conducted using the built-in kmeans function in Matlab (Matlab, 2019).

The Initial data for the first test (Fig. 23) defines clear human readable clusters. This test ensures that the algorithm is functioning correctly as visualised in Fig. 24. K-means sports an efficiency of O(n*k*I*d) where *n* is the number of points, *k* is the number of clusters, *I* the iterations and *d* the number of attributes (Tan, Steinbach, Kumar & Ghosh, 2002). On average k-means convergence occurs around 4 iterations yielding acceptable results, improving on higher iteration counts.

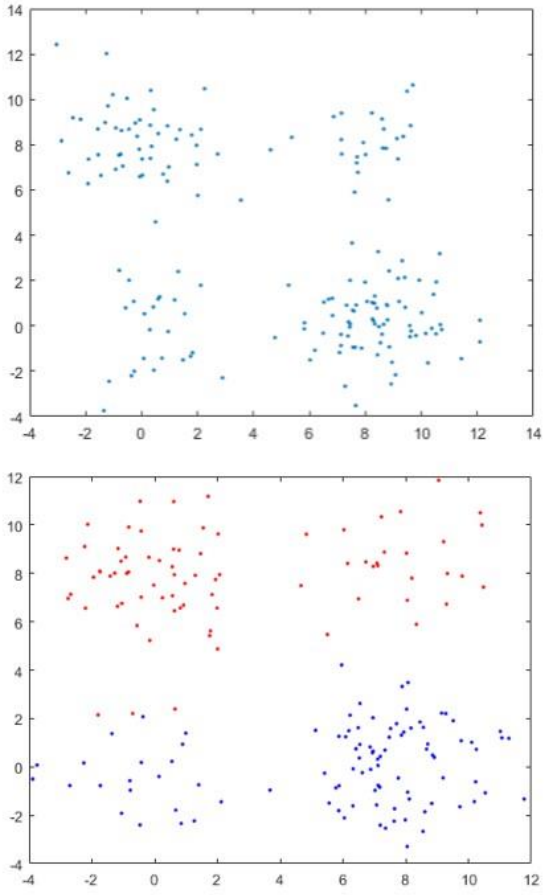

Fig. 23 K-means Clustering: Clear Cluster

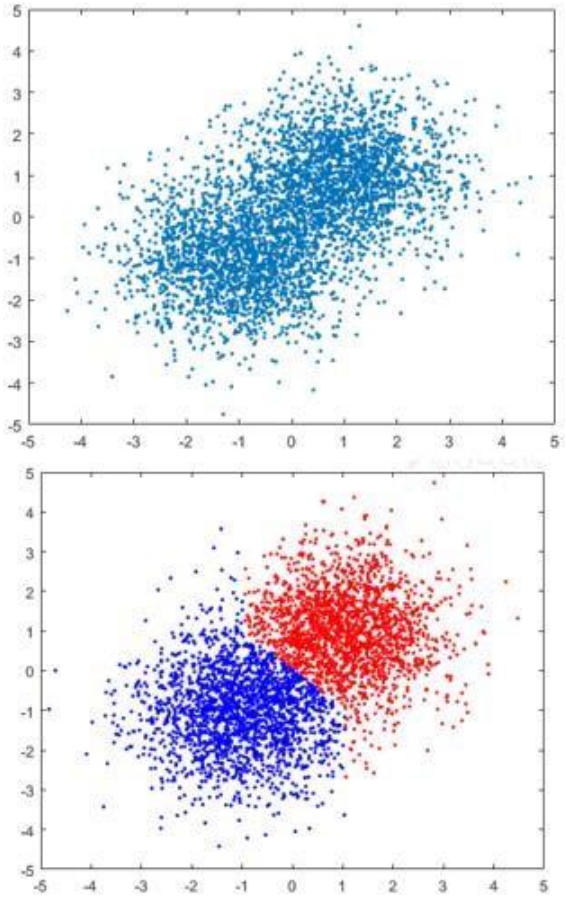

Fig. 24 K-means Clustering: Complex Cluster

12 Centroid Based Ant Clustering

ACO and k-means both have implicit advantages and disadvantages. K-means is fast for processing but is not a generic solution to the clustering problem, whilst the naïve ACO implementations take large amounts of time and iterations to attain an adequate solution comparative to the run time of k-means. Utilising the idea of k-means, ACO is able to perform clustering tasks in a shorter timeframe.

12.1 Algorithm

While the algorithm for ant-based clustering (AC) works the same as ACC, the implementation also utilises the k-means idea of random selection. Initial solutions are built over the generative tours (Fig. 24) as in the basic ACO.

```
i = 1;
while i <= generation_limit
    tau = tau0./repmat(sum(tau,2),1,clusters);
    tours = zeros(population, dataset_items+delta);
    best_tour = [];

    % Begin Tours
    for ant = 1 : population
        for item = 1 : dataset_items

            r = rand();
            if r > pd
                r2 = rand();
                c = 0;
                t_size = sort(tau(item,:));

                for j = 1: t_size
                    c = c + t_size(j);
                    if r2 < c
                        disc = find(tau(item,:) == t_size(j));
                        % Update Tours
                        if size(disc,2) == 1
                            tours(ant,item) = disc;
                        else
                            tours(ant,item) = disc9floor(rand()*size(disc(2)+delta));
                        end
                        break;
                    end
                end
```

Fig. 24 Ant Clustering

Pheromone trails are then explored, and a weighted average assigned to the tours based on desirability. The main difference occurs when utilising the fitness function (Fig. 25). In this implementation the fitness is derived from the centroid's Euclidean distance to the surrounding node density, roulette wheel is still utilised throughout the algorithm for each decision-making process.

```
    % Calc Fitness with Centroid
    centroid = (weighted_average'*dataset_items)./repmat(sum(weighted_avg,1)',1,features);
    for item = 1 : dataset_items
        ed = sqrt(sum((dataset_items(item, :) - centroid(solutions(ant,item),:)).^2));
        tours(ant,end) = tours(ant,end)+ed;
    end
end
```

Fig. 25 Centroid Fitness

12.2 AC Iris Experiment

The Iris-setosa (Appendix 5) dataset is utilised in csv format for this experiment. Since the algorithm has only been prepared to be used with numerical data it is unable to retrieve features from other datasets including alternative forms of data. The beginning parameters are selected at random, utilising the default ACO parameters with a population of 100 for 2000 generations.

Dataset:	Iris-setosa
Population:	100
Evaporation Rate:	5%
Initial Desirability:	1
Pheromone Level:	1
Iterations:	2000

Clusters:	2

Fig. 26 Starting Parameters

In this solution, the results converge at an average of iteration 61 on small to mid-sized datasets (Fig. 27). While AC's processing is superior in comparison with ACC clustering, it is still outperformed by k-means and offers a similar quality of results when working with larger numbers of clusters, nodes or agents (Fig. 28).

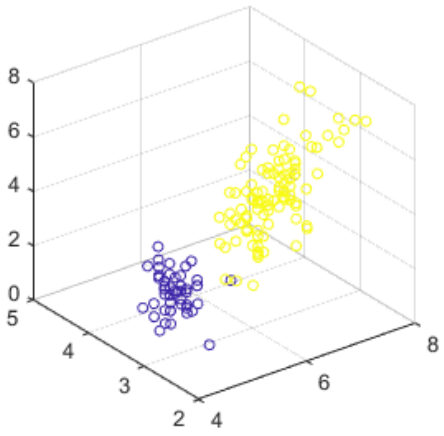

Fig. 27 2 Iris-Setosa Cluster

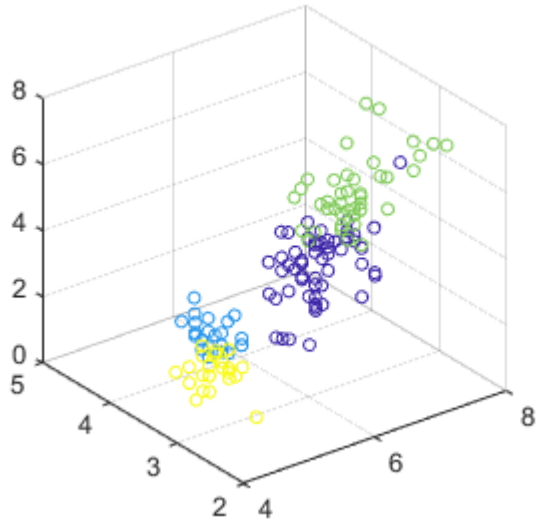

Fig. 28 4 Iris-Setosa Cluster

AC's performance suffers due to random fluctuations in fitness. While applying centroid fitness improves overall performance, the resulting clusters from datasets with more complex patterns can yield unreliable results as demonstrated in Fig. 29 where *x* is generation and *y* is fitness.

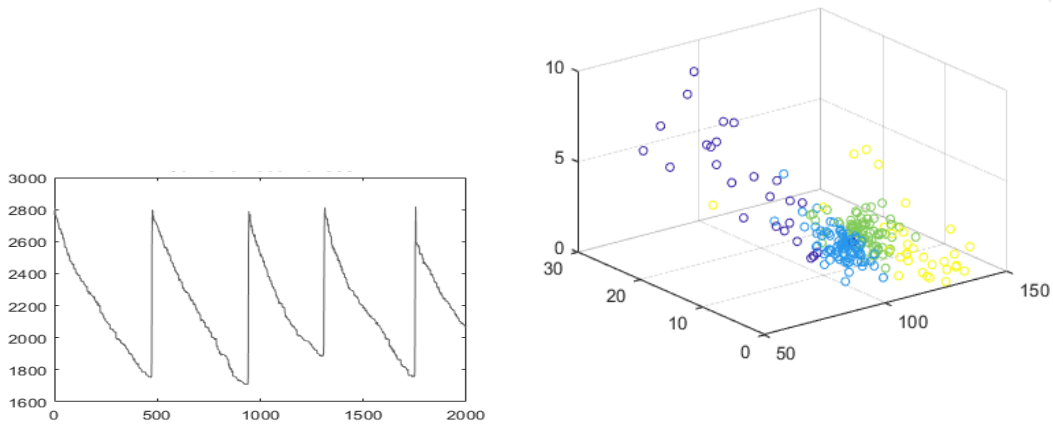

Fig. 29 Thyroid Dataset: Fitness Fluctuation and generated clusters

The fitness changes drastically between iterations, this may be resolved with a new method of calculating an average weighting in the future.

13 Classification

Feature extraction is a key part of classification. It is important to be able to select good features with which to classify data. Without the ability to extract features it is impossible to process the data further (Aghdam, Ghasem-Aghae, Basiri, 2009). Datasets are often stored as Csv or Attribute-Relation File Format (ARFF). ARFF files were originally developed for use with the machine-learning tool Weka by The University of Wakaito. ARFF files contain data along with the description of said data describing attributes.

While loading Arff files into python can be done with the external scipy library, Java proves a more difficult task requiring the use of a custom ArffReader (Fig. 30). The extract method then parses the file, utilising keywords from the data to attain the relation, attributes and data and store them into arrays for later use (Fig. 31). The data must also be represented via a feature class so that individual datums may be stored an retrieved via a custom object array.

```
public ArffReader(File file) {
    try {
        BufferedReader b_reader = new BufferedReader(new FileReader(file));
        // System.out.println(getExtension(file));
        if(!getExtension(file).equals(".arff")) {
            showErrorDialog("File must be arff format!");
        } else {
            // JOptionPane.showMessageDialog(new JFrame(), file.getName());
            // Read data line by line
            extract(b_reader);
        }
    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

Fig. 30 ArffReader Main Method

```
private void extract(BufferedReader b_reader) {
    try {
        for(String line = b_reader.readLine(); line != null; line = b_reader.readLine()) {
            if(line.startsWith("@")) {
                // remove @
                if(line.toLowerCase().contains("relation")) {
                    this.r_name = line.substring(line.indexOf("")+1);
                } else if(line.toLowerCase().contains("feature")) {
                    line = line.substring(line.indexOf("")+1);
                    this.feature_names.add(line.substring(0, line.indexOf("{"))
                            .trim().replaceAll("'", ""));
                    line = line.substring(line.indexOf("{")+1, line.indexOf("}"));
                    String[] sub = line.split(",");

                    for(int i = 0; i < sub.length; i++) {
                        sub[i] = sub[i].trim().replaceAll("'", "");
                        this.features_list.add(sub);
                    }
                } else if(line.toLowerCase().contains("data")) {
                    getFeatures();
                    }
                }
            }
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

Fig. 31 ArffReader Extraction

14 Fuzzy Logic

Fuzzy Logic (FL) has gained popularity in improving the way we utilised datasets. FL allows for logic systems to develop solutions by reproducing the natural human thinking process (Rouse, 2019). While propositional logic (PL) defines a series of statements with either true or false outcomes, FL allows for definitive circumstances in the cases where intermediate responses are more appropriate. For instance, a classifier wants to classify weather as hot, warm, cold and freezing. In conventional systems a specific drop-off would need to be stated; i.e. anything between twenty and thirty degrees is hot, but anything between ten and twenty degrees is warm. This does not function as natural temperature regulations in organisms such as lizards do as there is an instant point where the classification suddenly changes. FL systems allow for this to be corrected as intermediate responses can be given allowing for a smooth transition between differing states, this can be particularly useful in classifications and medical diagnostic in ailments such as diabetes (electricaltechnology, 2018).

With more research a fuzzy implementation of the AC algorithm would be possible, an example of this would be the C-means algorithm, developed by Dunn in 1973. While the Cmeans algorithm is almost identical to k-means it also contains a fuzzifier function (electriclTechnology, 2018). Applying a fuzzy AC algorithm would generalise the implementation further, allowing for an arbitrary complexity, whilst it can currently be considered generic to all datasets consisting of numeric data a more advanced implementation could theoretically process any dataset.

15 Conclusion

The clustering problem is a highly complex issue. The k-means algorithm is popular due to its simplicity and performance which is difficult surpass without a large loss in performance or unreliable clustering using traditional methods. Nature inspired algorithms are able to cluster and process data effectively, however, the complexity of applying some of the algorithms and the time taken to study and learn the implementation given the niche area of research the k-means algorithm is a more practical solution for programmers and businesses that are trying to acquire good results in a short time frame. If an algorithm was to be chosen, the best algorithm to implement is the AC implementation, this algorithm is simple enough to learn, provided the necessary experience and knowledge of ACO and k-means is present, and provides adequate results in a reasonable time-frame comparative with many of the other algorithms. It is important to note that the algorithms implemented within this study are all naïve versions of the algorithm due to time-restrictions and the amount of research required to attain the skills and knowledge necessary to carry out these solutions.

Overall, the time taken to learn the mathematics, utilise the tools and develop the algorithms is currently not worthwhile for professionals seeking to simply process data, A proposed fuzzy solution could combine the idea of the C-means fuzzifier, this would require more indepth research into the implementation and mathematics of fuzzy sets. An alternative to this could be to attempt to implement a neural network solution to the classifier problem, instead

using nature inspired algorithms discussed throughout this study to train the neural networks or adapt/adjust weights.

Throughout the course of the study many algorithms have been implemented and tested via Matlab. The overall aim has been completed; however, the implementation of the Java application is incomplete, in the future more time would need to be delegated to implementing the overall application. The next step forward from this study would be to continue this implementation in java and convert the completed AC algorithm as a classifier. The main issues arose towards the end of the project (Appendix 1). Progress has been tracked via an online calendar throughout the course of the project, however, due to unforeseen circumstances and last minute failures when utilising Matlab tools, some of these milestones were delayed.

While the process of developing the algorithm has been successful, time management has been an issue towards the end of the project. Future development would rely on focusing time more on a single implementation rather than many to ensure a complete application, this could be achieved by cutting down on research time and developing more of a focus towards implementing a single well-rounded solution to the overall problem rather than implementing and testing numerous algorithms. The clustering problem provides unique challenges, especially to those fresh to machine learning concepts. The mathematics involved in deriving the ACO algorithm in particular procured a large amount of time to fully understand and derive.

The main goal of the project has been reached, with some open-ended tasks for future implementation and the proposal of integrating a fuzzy based system to further develop the ant centroids solution.

Ethical Review Form

STATEMENT OF ETHICAL ISSUES AND ACTIONS

If the answer to any of the questions above is yes, or there are any other ethical issues that arise that are not covered by the checklist, then please give a summary of the ethical issues and the action that will be taken to address these in the box below. If you believe there to be no ethical issues, please enter "NONE".

As this research uses simulations the risks are minimal, The main issues are that of the use of Open Source software. As to which all items used within the project will be declared and referenced.

The aim of the research is to evidence the ability of existing algorithms in search and rescue operations, this will be performed in simulations, however the simulations should function correctly and efficiently for any whom may wish to add to this area in the future.

The social considerations are minimal, the project is heavily research based, the only potential issues arise if it were to be implemented in the real world, thorough testing is still needed as to minimize the potential for impact if this research was implemented in the real world.

STATEMENT BY THE STUDENT

I believe that the information I have given in this form on ethical issues is correct.

Signature: Luke McCann Date: 16.10.2018

AFFIRMATION BY THE SUPERVISOR

I have read this Ethical Review Checklist and I can confirm that, to the best of my understanding, the information presented by the student is correct and appropriate to allow an informed judgement on whether further ethical approval is required.

Signature: Date: 26/10/18

SUPERVISOR RECOMMENDATION ON THE PROJECT'S ETHICAL STATUS

Having satisfied myself of the accuracy of the project ethical statement, I believe that the appropriate action is:

The project proceeds in its present form	✓
The project proposal needs further assessment by an Ethical Review Panel. The Supervisor will pass the form to the Ethical Review Panel Leader for consideration.	

RETENTION OF THIS FORM

- The Supervisor must retain a copy of this form until the project report/dissertation is produced.

References

Tosun, O. (2014). *Artificial Bee Colony Algorithm*, , 179-180. doi: 10.4018/978-1-4666-5202-6.ch018.

Celso, M. (2006). Metaheuristics and Applications to Optimization Problems in Telecommunications. *Handbook of Optimization in Telecommunications*, , 103-128. doi: 10.1007/978-0-38730165-5_4.

Talbi, E. (2009). *Metaheuristics From Design To Implementation*. Retrieved from http://ie.sharif.edu/~so/Metaheuristics.pdf

Goss, S., Aron, S., Deneubourg, J.L., & Pasteels, J.M. (1989). Self-organized Shortcuts in the Argentine Ant . *et al. Naturwissenschaften*, (76) 579-581 . doi: https://doi.org/10.1007/BF00462870

Dorigo, M. & Gianni, D.C. (1999). Ant colony optimization: A new meta-heuristic. *2* (1477), . doi: 10.1109/CEC.1999.782657.

Dorigo, M. & Stützle, T. (2004). *Ant colony optimization*, Retrieved from https://ieeexplore.ieee.org/document/4129846.

Dorigo, M. (1992) Optimization, Learning and Natural Algorithms. (Ph.D. Thesis), Retrieved from iridia (ftp://iridia.ulb.ac.be/pub/mdorigo/journals/IJ.10-SMC96.pdf)

Glover, F. & Kochenberger, G.A. (2003). *Handbook of Metaheuristics*. United States: Springer.

Jaszkiewicz A., Branke J. (2008) Interactive Multiobjective Evolutionary Algorithms. In: Branke J., Deb K., Miettinen K., Słowiński R. (eds) Multiobjective Optimization. Lecture Notes in Computer Science, vol 5252. Springer, Berlin, Heidelberg doi: https://doi.org/10.1007/978-3-540-88908-3_7

Ombach, J. (2015). Lecture Notes in Computer Science. *Interactive Multiobjective Evolutionary Algorithms*, doi: 10.4467/20838476SI.14.001.3018.

Garg, A., Gill, P., Rathi, P., Amardeep, ., & Garg, K.K. (2009). International Journal of Recent Trends in Engineering. *An Insight into Swarm Intelligence*, *2* (8), Retrieved from https://pdfs.semanticscholar.org/f3ec/1b3793146f17920d02fd7d43dc0bf946b0f0.pdf.

Bonabeau, E., Theraulaz, G., Deneubourg, J.L., Aron, S., & Camazine, S. (1997). Trends in Ecology and Evolution. *Self-organization in social insects*, 188-93. Retrieved from https://www.sciencedirect.com/science/article/pii/S0169534797010483.

Li, L., Xu, G., Zhao, G., Dong, Y., & Wa

ng, D. (2011). *Cloud Task Scheduling Based on Load Balancing Ant Colony Optimization*, Retrieved from https://ieeexplore.ieee.org/document/6051750.

Dutta, A. (2018, 08 06). What Is The Network Effect? Why Is It Valuable? [Web log post]. Retrieved from https://www.feedough.com/network-effect/..

Cen, R. (2014). The Astrophysical Journal Letters. *American Astronomical Society logo American Astronomical Society logo Institute of Physics logo Institute of Physics logo A publishing partnership TEMPORAL SELFORGANIZATION IN GALAXY FORMATION*, *785* (2), Retrieved from http://iopscience.iop.org/article/10.1088/2041-8205/785/2/L21.

Muthiah-Nakarajan, ., Venkatarman, ., Noel, ., & Matthew, . (2016). Galactic Swarm Optimization: A New Global Optimization Metaheuristic Inspired by Galactic Motion. *Applied Soft Computing.*, (38), 771-787. doi: 10.1016/j.asoc.2015.10.034.

Wang, J. & Beni, G. (1989). International Journal of Recent Trends in Engineering. *An Insight into Swarm Intelligence*, , . doi: 10.1109/ISIC.1989.238706.

Karaboga, D. & Akay, B. (2009). Artificial Intelligence Review. *A History of Metaheuristics*, Retrieved from https://link.springer.com/article/10.1007/s10462-009-9127-4.

Bundgaard, M.B., Damgaard, T.C.D., & Winther, W.W. (2002). *Ant Routing System*, Retrieved from https://pdfs.semanticscholar.org/fecc/ba6e82f40f17831e6fd3f133d291e01f7a9d.pdf.

Flake, G.W.F. (1999). *The Computational Beauty of Nature*, doi: 10.2307/2589369.

Shekhawat, A., Poddar, P., & Boswal, D. (2009). *Ant colony Optimization Algorithms : Introduction and Beyond.*Retrieved from http://mat.uab.cat/~alseda/MasterOpt/ACO_Intro.pdf.

Rebekka, B. & Malarkodi, B. (2016). Multiple Input Multiple Output Detection Using Roulette Wheel Based Ant Colony Optimization Technique. *World Academy of Science, Engineering and Technology International Journal of Electronics and Communication Engineering, 10* (3), 444-450. Retrieved from https://waset.org/publications/10004949/multiple-input-multiple-output-detection-using-roulette-wheel-basedant-colony-optimization-technique.

Brownlee, J.B. (2011). *Clever Algorithms: Nature-Inspired Programming Recipes*. Retrieved from http://www.cleveralgorithms.com/nature-inspired/index.html

Rouse, M. (2019, fuzzy logic [Web log post]. Retrieved from https://searchenterpriseai.techtarget.com/definition/fuzzy-logic.

Williams, A. (2015, 11 18). Clustering is hard, except when it's not [Web log post]. Retrieved from https://searchenterpriseai.techtarget.com/definition/fuzzy-logic.

Stuart P. Lloyd. Least squares quantization in pcm. IEEE Transactions on Information Theory, 28:129–137, 1982.

Karaboga, D., Karaboga, G., Gorkemli, B., & Ozturk, C. (2012). A comprehensive survey: artificial bee colony (ABC)algorithm and applications. *Artificial Intelligence Review, 42* , . doi: 10.1007/s10462-012-9328-0. .

Tan, ., Steinbach, ., kumar, ., & Ghosh, . (2002). *The k-means algorithm.* Retrieved from http://www.cs.uvm.edu/~xwu/kdd/Slides/Kmeans-ICDM06.pdf.

Adghdam, M.H., Ghasem-Aghae, N., & Basiri, M.H. (2009). Text feature selection using ant colony optimization. *Expert Systems With Applications, 36* (3), 6843-6853. doi: https://doi.org/10.1016/j.eswa.2008.08.022.

electricaltechnology. *Introduction to Fuzzy Logic*. Retrieved from https://www.electricaltechnology.org/2018/02/fuzzy-logic-system.html. 2018

Citations generated via: https://library.hud.ac.uk/pages/apareferencing/