Master's thesis

2019

Eirik Baug, Andreas Norstein

NTNU
Norwegian University of
Science and Technology
Faculty of Information Technology and Electrical
Engineering
Department of Computer Science

Master's thesis

Eirik Baug
Andreas Norstein

MAIM: A Novel Island Based Evolutionary Classification Algorithm

August 2019

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

MAIM: A Novel Island Based Evolutionary Classification Algorithm

Eirik Baug
Andreas Norstein

Master of Informatics
Submission date: August 2019
Supervisor: Pauline Catriona Haddow

Norwegian University of Science and Technology
Department of Computer Science

Abstract

The immune system is arguably one of nature's most highly adaptive, distributed and self-organising systems. It has the property of being able to recognise anomalies, something that deviates from the common rule. When immune principles are applied in algorithms they are effective solvers of both optimisation and pattern recognition problems. However, such algorithms typically do not fill any defined niche where it is the best tool for the job, as many of the immune system's natural properties are not sufficiently utilised. Further, one such property is the immune system's inherent distributed nature which the work herein exploits in order to enhance classification efficiency and accuracy over that of a more traditional Artificial Immune System (AIS). To this end a novel hybrid classification algorithm MAIM is proposed, combining AIS with an Island Model Genetic Algorithm (IGA). Consequently, the proposed algorithm employs a distributed AIS population with partially isolated sub-populations that communicate through exchanging genetic material.

The following work thoroughly investigates the properties of the individual techniques and components chosen for the proposed algorithm and their likely applicability in achieving the performance enhancements sought. Subsequently, key features of the model are presented and evaluated through testing in terms of their contribution to accuracy and efficiency. As a result, model properties are discovered and limitations investigated, resulting in components being revised in order to further enhance model performance. Consequently, MAIM is shown to employ a computationally efficient architecture while simultaneously possessing a generalisation ability on par with several state of the art algorithms.

Sammendrag

Immunsystemet er uten tvil et av naturens mest tilpasningsdyktige, distribuerte og selvorganiserende systemer. Det har egenskapen til å kunne gjenkjenne unormalheter, altså noe som avviker fra den vanlige regelen. Når immunprinsipper brukes i algoritmer, løser de effektivt både optimaliserings- og mønstergjenkjenningsproblemer. Imidlertid fyller ikke slike algoritmer noen definert nisje der det er det anses som det beste verktøyet for jobben, ettersom mange av immunsystemets naturlige egenskaper ikke blir tilstrekkelig utnyttet. Derav utnytter dette arbeidet immunforsvarets iboende distribuerte natur for å forbedre klassifikasjonseffektiviteten og nøyaktigheten i sammenligning med et mer tradisjonelt kunstig immunsystem (AIS), samtidig som det beveger seg videre mot en mer definert nisje for AIS-algoritmer. I det følgende arbeidet, foreslås en ny hybrid klassifiseringsalgoritme MAIM, som kombinerer AIS med en øy-modell (IGA). Den foreslåtte algoritmen benytter en distribuert AIS-populasjon med delvis isolerte underpopulasjoner, som kommuniserer gjennom utveksling av genetisk materiale.

Dette arbeidet er en grundig undersøkelse av egenskapene til de nevnte teknikkene og hvordan de på kan kombineres for å oppnå de ønskede ytelsesforbedringene. Deretter presenteres og evalueres nøkkelfunksjoner i den foreslåtte modellen MAIM, med tanke på dens bidrag til klassifiseringsnøyaktighet og effektivitet. Videre blir begrensninger av modellen undersøkt, som igjen resulterer i videre revideringer av modellen for ytterligere forbedringer. Til slutt viser MAIM til å ha en svært effektiv arkitektur, som har en generaliseringsevne på nivå med moderne algortimer.

Preface

The following thesis is the resulting work from a master thesis conducted at Norwegian University of Science and Technology in Trondheim, Norway, in the period of 12.08.2018 - 05.08.2019.

We would like to thank our supervisor Pauline Catriona Haddow for excellent guidance throughout the project. We would also like to thank our good friend Martin Stigen for letting us use parts of his user interface code in our algorithm simulator.

Eirik Baug and Andreas Norstein
Trondheim, August 4, 2019

Contents

List of Figures

List of Tables

Abbreviations

NTNU	=	Norwegian University of Science and Technology
GUI	=	Graphical User Interface
MAIM	=	Multiple Artificial Immune Model
GA	=	Genetic Algorithm
IGA	=	Island-Model Genetic Algorithm
MR	=	Migration rate
MF	=	Migration frequency
IS	=	Immune System
AIS	=	Artificial Immune System
AB	=	Antibody
AG	=	Antigen
RR	=	Recognition region
LFS	=	Local Feature Selection
IP	=	Island population
P	=	Total population
F(b)	=	Fitness of AB b
F(P)	=	Fitness of population P
NA (*na*)	=	Not Available
KNN	=	K-Nearest Neighbours
SD	=	Standard deviation
PCA	=	Principle Component Analysis

Chapter 1

Introduction

A novel hybrid bio-inspired algorithm for supervised classification: *MAIM - Multiple Artificial Immune Model*, is proposed herein. MAIM is a population-based distributed artificial immune system, combining the strengths of artificial immune system algorithms and island-model genetic algorithms. This thesis provides insights into the design decisions made in the light of the state of the art, testing, evaluation and revision of the proposed algorithm.

The following chapter will elaborate on the background and motivating factors in section 1.1. Section 1.2 elaborates on the goals and research questions. In section 1.3 the research method is introduced. Section 1.4 presents the structured literature review process. Section 1.5 shows the overview of the preliminary process. Finally, the structure of the thesis is presented in section 1.6.

1.1 Background and Motivation

The *immune system (IS)* is the product of millions of years of evolution. It is a highly adaptive system consisting of two main sub-systems; the *innate* and *adaptive* IS. On one hand, the innate IS targets anything it deems to be a pathogen and does not change during the lifetime of the host [18]. On the other hand, the adaptive IS targets specific intruders, the *antigens (AGs)*, through the application of its *antibodies (ABs)*. Further, the adaptive IS evolves specialised ABs which, upon repeated AG exposure, become increasingly proficient at recognising specific AGs. Through such adaptive processes of AB specialisation, the IS is capable of learning [6].

Artificial Immune Systems (AIS) are population based and evolution driven algorithms, that solve computationally heavy problems of a wide variety of types by employing IS principles of learning, memory and adaption. However, as stated by Hart et al [15], AISs lack a defined application niche and should be focused on applications that exploit the inherent nature of the IS, which, among others, include its distributed properties. Some recent works [19, 38, 8] have successfully used IS principles in classification, as classification is an inherently distributed task suitable to the distributed nature of AGs and ABs.

Island Model Genetic Algorithms (IGAs), on the other hand, were originally designed as distributed algorithms where sub-populations or islands separately evolve and communicate through exchanging genetic material. Further, such algorithms have been shown to have a natural tendency to exploit the separable nature of problems [40] and improve results through enhanced convergence and exploration [17, 12]. Studies have also shown that it is possible to achieve linear speedup for non-linear problems [27].

IGAs have mostly been applied to optimisation tasks. However, they have shown to be proficient at enhancing distributed *genetic algorithms (GAs)* [40], which in many ways are similar to those of AIS classification. Subsequently, the following work is situated within the field of biologically inspired computation and proposes a new hybrid classification algorithm, MAIM, that combines the AIS and IGA techniques in a distributed artificial immune system.

1.2 Goals and Research Questions

This section defines the goal statement and research questions that will be investigated in this work. The research goal of the work is the following:

Goal *How can artificial immune systems be combined with the island model to create an accurate and efficient novel hybrid classification algorithm.*

The desired enhancements by combining AIS and IGA include improvements in both efficiency and accuracy, which means the algorithm should simultaneously be faster and more accurate than the AIS model implemented running without an island structure. Additionally, the distributed AIS proposed in this work should also be able to perform on par with other state of the art AIS classification algorithms. However, in the event that such results are not achieved, the goal is to understand why the results were below expectations and propose, implement and revise model components.

Much of the work in enhancing the classification process and result therefore lies in exploiting the inherent separate nature of the problem solving process achieved from evolving several different populations with IGA. Further, the distribution of population and evolution, which is a significant feature of the IGA technique, must effectively be used in the proposed algorithms favour. Consequently, parts of the work concerns identifying how IGA can best be applied to an AIS in order to improve the overall performance. Subsequently, a thorough investigation of background and state of the art literature will be conducted in order to determine such properties. Additionally, by focusing on and exploiting the inherent distributed nature of the IS, another goal of this work is to further AIS algorithms towards a more defined niche (see section 2.2.2). The research questions explored are as following:

Research question 1 *How can the island model combined with artificial immune systems be used to increase the accuracy of the classification?*

Research question 2 *How can the island model combined with artificial immune systems be used to increase the efficiency of the classification process?*

Research question 3 *How does the migration configuration used impact the accuracy results of the algorithm?*

Both accuracy and efficiency enhancements are sought in the proposed algorithm. Subsequently, it is natural to look at these as separate research questions or sub-goals. As a result, the experiments will first focus on evaluating accuracy and efficiency, in separate experiments, as reflected in the experimental plan in chapter 4. Finally, additional experiments will investigate the effects of IGA migration, i.e. the exchanging of genetic material between sub-populations, on the accuracies achieved.

1.3 Research Method

The research method applied was first and foremost an analytic process. As the research objective is something that requires a novel combination of AIS and IGA, the two techniques were first investigated individually through a structured literature review. Further, the techniques were analysed by looking at the different components and properties of the techniques, their behaviour and impact on the result. Through this, promising properties of both the techniques were investigated in terms of how they could potentially positively affect the efficiency and accuracy of the proposed algorithm.

The knowledge gathered from the literature search was used to select and justify the design decisions of the algorithmic model. After completing the model it was implemented with the addition of a visual interface for viewing the progress and results during and after a run of the algorithm. Additionally, an experimental plan was developed to test important aspects of the model, as explained in section 1.2, in order to best answer the research questions. Finally, it was discussed to what degree the overall goal were achieved, what contributions were made and how the work could be further extended in the future.

1.4 Structured Literature Review Protocol

The following research questions and a strategy have guided the structured literature search in finding relevant literature, substantiating the work in this thesis. Further, this includes questions that have guided the search, as well as inclusion criteria that must be fulfilled for the article to be included and quality criteria to assess the value given to the research.

1.4.1 Research Questions

As the project description was not set at the start of the project, the guiding research questions for the literature review have changed somewhat throughout the course of the search. However, one question that has been an important factor during the whole process was: "How can biologically inspired methods be fully integrated with machine learning in a hybrid classification algorithm?". Further, the intention, from the beginning, was to angle the project toward machine learning combined with bio-inspired methods. Additionally, it became clear that most hybrid systems, investigated during the research phase, employed a two-step process. In such methods already established machine learning techniques were applied separately from a bio-inspired optimisation method used in the pre- or post-processing stages of the algorithm.

As the project got more focused, the search changed towards questions targeted on specific techniques like artificial immune systems, island-model genetic algorithms and generally hybrid methods of bio-inspired algorithms. After deciding where to focus the research, the following questions has guided the literature search:

- How can classification in artificial immune systems be enhanced?
- Under what conditions does use of the island model positively affect convergence and result?

On one hand, for AIS, further research was conducted into antibody shapes and techniques such as local feature selection for enhancing classification. On the other hand, for IGA, the research was focused towards migration policies, island topoligies and applications the island model were shown to improve the performance of the algorithm.

1.4.2 Research Strategy

In attempting to answer the research questions a few quality and inclusion criteria had to be defined in order to asses the article's value for the research, as well as keywords used when searching. These factors have been summarised in table 1.1 and was used to identify relevant articles. Further, if an article was from a journal, the journal should hold a respectable score according to international standards to ensure that the information is credible. Finally, more recent articles, preferably as recent as 5 years or less, were given greater value.

Identifying	**Keyword used when searching (comma separated):** • Artificial immune system, island based genetic algorithm, machine learning, classification, feature selection, migration, antibody shapes, clonal selection.
Qualifying	**Inclusion Criteria** • Articles have to have IGA or AIS as their main research topic. • The articles have to appear relevant from only reading the abstracts and conclusion. • The articles should include a further work section that elaborates on possible improvements
Evaluating	**Quality Criteria** • The algorithms or techniques being presented in the research are compared against other algorithms known to be proficient in its field of research. • The article also needs to provide sufficient background information to understand the relevant topics presented. • The research must clearly state its purpose and it must be evident that it actually contributes to the field.
Including	Research published by respected sources found on sites like IEEEXplore, ScienceDirect, Google Scholar, etc...

Table 1.1: Article Selection Criterias.

1.5 Preliminary Process Overview

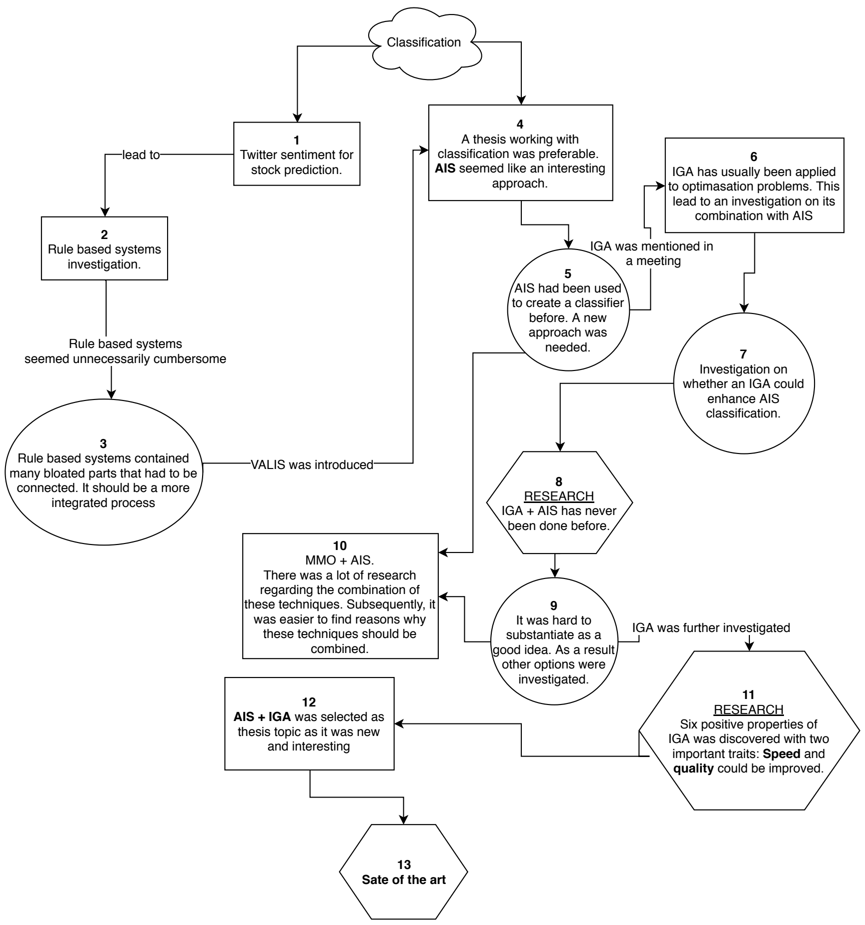

Classification
1 Twitter sentiment for stock prediction.
lead to
2 Rule based systems investigation.
Rule based systems seemed unnecessarily cumbersome
3 Rule based systems contained many bloated parts that had to be connected. It should be a more integrated process
VALIS was introduced
4 A thesis working with classification was preferable. AIS seemed like an interesting approach.
5 AIS had been used to create a classifier before. A new approach was needed.
IGA was mentioned in a meeting
6 IGA has usually been applied to optimasation problems. This lead to an investigation on its combination with AIS
7 Investigation on whether an IGA could enhance AIS classification.
8 RESEARCH IGA + AIS has never been done before.
9 It was hard to substantiate as a good idea. As a result other options were investigated.
IGA was further investigated
10 MMO + AIS. There was a lot of research regarding the combination of these techniques. Subsequently, it was easier to find reasons why these techniques should be combined.
11 RESEARCH Six positive properties of IGA was discovered with two important traits: Speed and quality could be improved.
12 AIS + IGA was selected as thesis topic as it was new and interesting
13 Sate of the art

Figure 1.1: Preliminary process overview.

Through the selection of different research topics, a goal was developed iteratively as reading and different ideas progressed, with research leading into several different sub-fields of biologically inspired computing, as shown in figure 1.1. Further, initial research started by looking into sentiment analysis of tweets on Twitter, optimised by a genetic algorithm to predict the movements of the stock market in the United States [33]. The work of the article was focused heavily on the sentiment analysis which were implemented using a support vector machine classifier and not a bio-inspired method. However, the system also employed a bio-inspired rule based model for buying and selling stocks, which seemed unnecessarily complex as it required several steps of processing using both a machine learning and a genetic algorithm. After a thorough literature review, the results showed that most research that combined machine learning and bio-inspired techniques were heavily module based and few systems combined the techniques in a fully integrated system.

The main module would typically be a standard machine learning approach, while the steps of pre- or post-processing would be optimised with some from of genetic algorithm. Therefore, the conclusion was that it would be more interesting to fully integrate bio-inspired concepts with machine learning to create a proper evolution driven bio-inspired machine learning algorithm. Subsequently, research into AIS classification algorithms was conducted through the recently published VALIS algorithm [19]; a classification algorithm based on principles from the adaptive immune system. During this investigation it was concluded that in general AIS algorithm encounter some limitations in terms of not sufficiently exploiting the properties of the IS.

The idea of combining AIS with IGA was conceived as AIS seemed to have many properties exploitable by distributed systems. Subsequently, the techniques were further researched, but no studies combining the techniques were found. Consequently, it was concluded that the island model had mostly been used for optimisation problems, which made it difficult to substantiate that combining IGA with AIS would enhance the performance of the algorithm.

Considering no research was found on the combination of AIS with IGA, or IGA in machine learning methods in general, a conclusion was made that it would be more sensible to look into other techniques that could possibly be used in enhancing classification algorithms. Consequently, research into *multi-objective-optimisation (MOO)* was conducted, which had previously been combined with AIS and therefore made it easier to substantiate researching. However, the idea of combining AIS with IGA was considered to be a more unique approach, which in turn lead to another round of research into IGA. Subsequently, a list of positive IGA properties was produced, where it was concluded that the technique, under the right conditions, could enhance the quality of the result and the efficiency of the algorithm. Finally, as a result of promising articles found on the second research sequence into IGA, a decision was made to select the research objective as *the creation a hybrid classification algorithm based on a combination of the IGA and AIS techniques.*

1.6 Thesis Structure

Subsequent chapters will be presented as following: Chapter 2 contains the background theory, where basic concepts are introduced and explained, followed by the state of the art for the IGA and AIS techniques. Chapter 3 ties the state of the art into the chosen model and presents the proposed algorithm. Chapter 4 displays the algorithm simulator and contains the experimental plan, results and evaluations. Finally, chapter 5 presents discussion and goal evaluation, contributions and future work.

Chapter 2

Background Theory and State of the Art

The following chapter elaborates on the background theory of the thesis in section 2, as well as presenting and discussing the state of the art of the of IGA and AIS techniques in section 2.2. Finally, a summary of the state of the art is included in section 2.2.3.

2.1 Background Theory

The background theory employed in this thesis will be presented in the following sections. These include introductions to the project's main topics; starting with genetic algorithms in section 2.1.1 and followed by the island model in section 2.1.2. Furthermore, section 2.1.3 introduces classification and section 2.1.4 immune systems.

2.1.1 Genetic Algorithms

Genetic Algorithms (GAs) are a type of iterative search algorithms commonly used in optimisation problems that draws inspiration from biological evolution. Each iteration of the algorithm is referred to as a generation in nature. These algorithms operate on a population of individuals where each individual represent a possible, or parts of a possible, solution to the problem. One such solution consisting of a set of genes, i.e. its *genetic material*, is called a *chromosome*.

An example of the processing cycle in a GA is presented in figure 2.1. The algorithm starts by creating an initial population of more or less randomly generated individuals. Further, it uses this initialisation to start its evolutionary cycle, marked with green on the figure. On each iteration individuals are selected for breeding, where stronger individuals have a greater chance of being selected. This selection process is referred to as *parent selection*, and the breeding process itself is called a *crossover*, where typically two individuals are combined into one

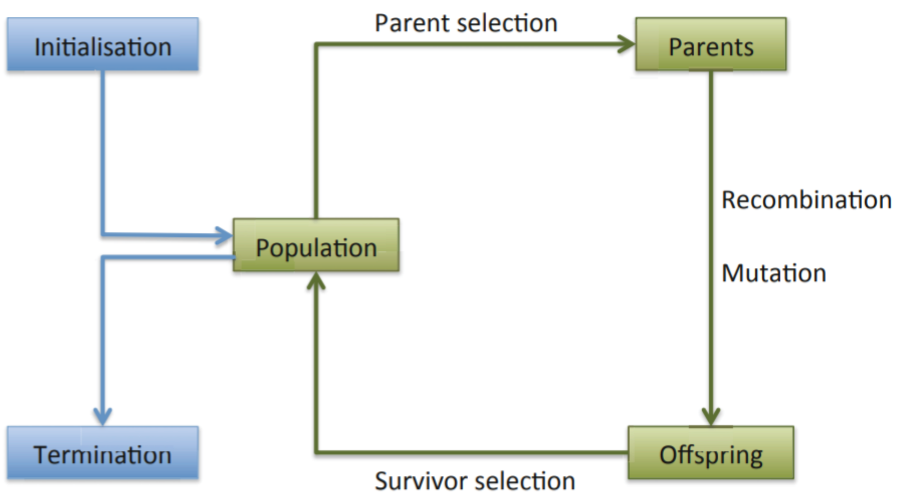


Figure 2.1: Overview of a GA (adapted from [9]).

or more offspring. One common way of performing a crossover is referred to as a *uniform crossover*, where the offspring's genes are randomly selected from its parents, typically with each gene having an equal probability of selection. However, the crossover process is not something that is necessarily implemented in a GA as some algorithms only employ a *mutation* process. Mutation is typically applied to the offspring after a crossover and commonly consist of randomly changing parts of the individuals genetic material or, alternatively, more informed, heuristic based changes. Similarly to a uniform crossover, a *uniform mutation* is commonly applied where every gene has an equal probability of being mutated. The goal of using older individuals to create or change new ones is not that all offspring need to be stronger than their parents. However, the population on average should improve over consecutive generations [9]. Further, the strength of these individuals are calculated as a *fitness* score based on how well they provide a solution to the problem being solved [10].

The number of individuals in a population usually remains constant over the course of the search, meaning that a population size of n individuals always makes it to the next iteration. If more than n individuals remain at the end of the GAs iteration, *survivor selection* is typically employed. Here, individuals are selected for the next generation, where the fitter individuals are more likely to survive, until n individuals are selected for survival [10].

The process of selection, crossover and mutation are repeated iteratively until the algorithm creates a sufficiently strong individual or the maximum number of iterations is reached. These tasks are often regulated by several parameters which typically consist of, among others, crossover and mutation rate, maximum iterations and population size. Much of the success of GAs lies in the choosing and tuning of parameters [10].

Selection Methods

Selection of parents and survivors can be done in several different ways, but generally each individual has some probability of being selected based on its fitness. Different methods provide varying degrees of *selection pressure* and *genetic diversity* [10].

On one hand, the level of selection pressure indicates the amount of individuals that will be selected for reproduction, which means that with a high selection pressure the algorithm may converge fast and create individuals of high fitness. However, this increases the risk of the population becoming homogenised and the algorithm suffering from *premature convergence*. When this occurs the population has converged to a point where consecutive iterations does not improve a solution that is unsatisfactory [10].

On the other hand, genetic diversity refers to the level of which the individuals in the population differ from each other, where a high diversity means that the population contains many individuals of vastly different chromosomes and fitness values. Further, the risk of premature convergence decreases when diversity increases. However, too much diversity in the selection process may cause the algorithm to diverge if not enough of the fittest individuals are being selected, making it similar to random searching. As some level of both selection pressure and genetic diversity is desirable, GAs typically need to make compromises between them, as achieving a high level of both is often not feasible as increasing one counteracts the other [10].

An example of a selection method is *Fitness proportionate selection* which gives every individual a probability of being selected as its fitness value divided by the cumulative fitness of the whole population [10]. Based on this, fitness proportionate selection will select a set of individuals of given size, s, from a provided set of selectable individuals called the *selection pool*. This is illustrated in figure 2.2, where individuals are depicted as boxes with letters and the size of the boxes correspond the fitness of the individual. In the example the selection process moves successively through the list from A to E until an individual is selected. On each individual, the probability of it being selected is the sum of the cumulative probability of all the previously evaluated individuals that was not selected and the probability of the individual currently under evaluation being selected. If the last individual in the sequence is reached it will be selected with a 100% certainty. When an individual is selected, it is removed from the selection pool and moved to the set of selected individuals. When this happens and s individuals have not yet been selected, the cumulative fitness of the remaining selectable population is recalculated and the selection process repeated.

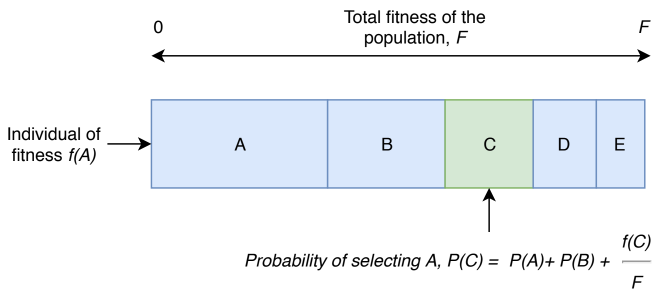

0
Total fitness of the population, F
F
Individual of fitness f(A)
A
B
C
D
E
Probability of selecting A, P(C) = P(A)+ P(B) + f(C)/F

Figure 2.2: Illustration of fitness proportionate selection.

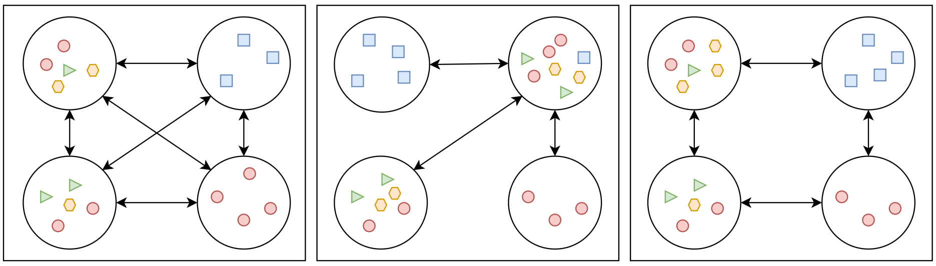
Fitness proportionate selection usually provides the best compromise between selection pressure and genetic diversity when the population contains individuals of significantly varying fitness values. Otherwise, every individual would receive a similar chance of being selected and the evolutionary search will degrade to just random search. However, in this approach no individual should have a much higher fitness than all the others, as such an individual will be selected almost every time, meaning that it may end up dominating the reproduction process and cause premature convergence [10].

Tournament Selection is another selection methods that selects a tournament set which consists of a given *tournament size* randomly selected individuals. Furthermore, the individual with the highest fitness value in the tournament set is returned from the selection process. Subsequently, a larger tournament set means a greater probability of the individual being selected having a high fitness value relative to the rest of the population. Additionally, as each tournament only selects one individual the number of tournaments needed is equal to the number of individuals being selected. This method is usually able to achieve a good compromise between selection pressure and genetic diversity depending on the size of the selection pool and *tournament size* used [10].

2.1.2 The Island Model

The *Island-Model Genetic Algorithm (IGA)* or just *island model* is a method designed for enhancing population based optimisation techniques by employing a distributed population scheme. The technique is usually implemented with n sub-populations which in the IGA context is referred to as *islands*. Each island runs a population-based GA, initialised with a population that is evolving partially isolated from the other islands. The different islands can communicate through *migration* of individuals. This means that an island can both send and receive individuals from other islands [40].

In [17] Huang states that there are six important parameters to handle when using the IGA technique. The first is to decide the *number of islands* or sub-populations to employ and the second is the *population size* for each island. I.e, figure 2.3(a) illustrates four islands with populations varying from four to six individuals. The third important parameter is the *island topology*, which defines the



(a) Fully-connected (b) Star-shaped (master-slaves) (c) Bidirectional ring

Figure 2.3: Common Island Topologies.

connection between the islands. These connections defines the authorised migration routes for each islands, i.e where each island is allowed to send its individuals. To illustrate this figure 2.3 shows three different island topologies. Here, (a) is a fully connected topology where the migration routes are from everyone to everyone. Furthermore, (b) shows a star-shaped topology, where one island, here referred to as the master island, have migration routes to all the other islands (the slave islands) while the slaves only have routes to the master. Finally, (c) shows a ring-shaped topology, where the migration routes only connects the immediate neighbours of each island. The fourth important parameter is the *migration rate (MR)*, which typically controls what percentage of the island population are to be moved from one island to another during migration. Further, the fifth important parameter is the *migration frequency (MF)* or migration interval, which defines the number of algorithm iterations between each migration. Finally, the last important parameter is the *migration policy*, which is the policy for selecting emigrants during migration. Such a policy could for instance state that either the strongest or the most diverse individuals should be selected for migration.

All the parameters mentioned above are crucial for an IGA, and all of them have an impact on the efficiency of the algorithm and quality of result. However, another aspect that should be taken into consideration, especially if the IGA implementation is done in a parallel fashion, is whether the migration should be synchronous or asynchronous. Synchronous migration means that that all islands migrate their selected individuals at set points in an iteration and waits until migration if finished before continuing. Asynchronous migration on the other hand, means that an island can send and receive individuals at any time, which for instance could be as soon as it find the suited individual(s). Synchronous implementations are usually simpler than asynchronous implementations, but the latter is typically more flexible and efficient [12].

The choice of selecting either homogeneous or heterogeneous islands should be addressed when designing an IGA. In [12] Gong et al. describes a homogeneous island model as a model where all islands adopts the same genetic operations, fitness function and selection strategy. This approach is very straight forward, but it has some known weaknesses. If the physical layer running the algorithm consists of different processors, the slowest processor will be a bottleneck regarding the efficiency of the algorithm. Furthermore, by letting different islands use the same parameters and components, they may not be able to balance local- and global exploration. On the other hand, a heterogeneous IGA design may employ different settings and strategies for the different islands and therefore avoid the weaknesses of a homogeneous design. However, this often comes at the cost of increased algorithm complexity.

Advantages and Disadvantages

The island model has become a well known and accepted strategy for keeping a high diversity within a population. It is known to be very flexible, due to the possibility of having multiple strategies in order to maintain diversity within the islands. By dividing the total population into smaller migrating sub-populations it may vary the level of genetic diversity and selection pressure on each island, depending on the selection scheme used (see section 2.1.1). This in turn may help to achieve better convergence and higher diversity of each sub-population, providing an enhanced exploration of the search space and subsequently better solutions [17, 12]. Additionally, IGAs may be implemented very efficiently by exploiting their inherent parallel nature, due to the fact that sub-populations are evolving mostly independently from each other. This allows for the possibility of running each island on separate computational nodes or computer cores, potentially drastically reducing the runtimes of the algorithm [21].

The island model may suffer from islands becoming homogenised and prematurely converging, often as a result of selection pressure on each island being too high. This typically occurs when the best individuals on an island are selected for migration and sent to all other islands, which over time makes a few strong individuals dominate the gene pools on all the islands. Subsequently, it is important to employ an effective migration scheme tailored to the needs of the algorithm [21].

Biological Similarities

By looking at how evolution works in biology and studying smaller, natural, populations, the famous geneticist Sewall Wright [41] argues that having closed populations breed internally with occasional crossbreeding, might greatly improve the convergence rate of the populations. However, too much inbreeding will lead to extinction and too much mutation will lead to a population of anomalies. Further, Wright also states that how much inbreeding, mutation and crossbreeding is necessary will vary between the populations and it is important to maintain a balance between the factors. Furthermore, these properties of biological populations are also applicable to IGA. Subsequently, each island exploits the rapid convergence of inbreeding. Further, the migration of individuals between the islands ensure that excessive inbreeding will not take place while keeping a certain level of diversity in the populations.

2.1.3 Classification

Classification is the task of categorising. When this is done by an algorithm it is commonly referred to as machine learning. More formally, machine learning is the process by which the algorithm learns from experience with respect to some class of tasks, and a performance measure to assess how well the assigned tasks perform in the chosen task environment. If the program increases its performance measure at its assigned tasks given its experience, it is said to learn. Learning is the foundation of machine learning, and subsequently also classification [26].

Machine learning algorithms can be divided into several types, depending on the problems they solve and how they solve them. Common types include regression, association learning, classification and reinforcement learning. Classification will be the focus of the machine learning aspect of this thesis. As mentioned, classification is the task of categorising, and it is performed given a set of examples with attributes to distinguish, and classes to categorise by. This has many real world applications. An example of this being credit score evaluation in a bank. A bank typically want to assess whether a customer is high or low risk in regards to being able to pay back a loan or not. In other words, the goal is to categorise the customer into one of two different classes; high-risk or low-risk. Each customer is then represented by its set of attributes or *features*. This set of features could for instance include the customer's age, name, savings, profession and so on. The customer can then be represented in n-dimensional feature space, the space containing all possible features of n dimensions, where n is the number of features assigned to each customer [2].

The classification or machine learning task here would then be to find the plane in feature space that best divides the examples into its respective groups. This will usually be done by using a set of pre-labeled training examples; a set of example customer which have already been correctly labelled as one of the two classes. This is referred to as *supervised learning* where the algorithm is provided with a set of already classified examples and based on these attempt to create a generalised function and thus also classify new unseen examples. More specifically, the task is to best approximate the true function that correctly classifies all examples given a set of pre-labelled examples. These learned functions takes an input example, also commonly referred to as a *case*, and returns one or several output classes for that case. Cases used during the supervised learning process are usually divided into *training*, *validation* and *test sets*. The training set is used to iteratively approximate the classification function in the training phase of the algorithm. Furthermore, during training, the validation set, consisting of cases left out of the function approximation, is used to evaluate and select the current candidate function(s) at each iteration. This is done in order to evaluate the function(s) over unseen cases in order to improve the final function's generalisation ability. However, it should be noted that using a validation set is optional as the function can alternatively be evaluated using only the test set. When training is completed a final function approximation is returned and the algorithm transitions to the test phase. Here, the function is evaluated by seeing how well it is able to generalise and classify when applied to the unseen cases of the test set [26].

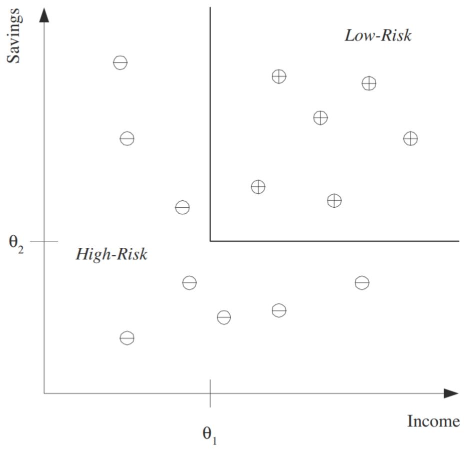


Figure 2.4: Credit score evaluation problem (adapted from [2]).

Going back to the credit score evaluation problem one can do a simple representation of each customer by representing them in a 2-dimensional space with only savings and income as features. The classification algorithm could for instance be the process of finding the respective threshold, θ_i, for each feature, and any customer that is above both thresholds (have enough savings and income) is deemed low risk, as shown in figure 2.4. This can be simplified into a straightforward function such as ”IF income is greater than θ_1 AND savings is greater than θ_2 THEN low-risk ELSE high-risk” [2]. Finding this function may involve several iterations of testing how well the function perform by applying it to the training and/or validation set, calculating an error value based on its performance and using this error to adjust the function. The algorithm terminates when it correctly classifies all examples, detects premature convergence or reaches the maximum number of allowed iterations [26].

When testing the generalisation ability of a classification algorithm, different validation schemes may be employed. One method that is often adopted is the *k-fold cross-validation* approach. Here, a parameter, k is picked and the dataset is split into k equally large parts. Further, the algorithm is ran k times with each part being used as the test set exactly once and otherwise being a part of the training set when not selected. When all parts have had their turn as the test set, an accuracy averaging the k runs is returned as a measure of the algorithm's generalisation ability [26].

Ensemble Learning

One commonly used machine learning technique is *ensemble learning.* Ensemble methods combine multiple weak models into one strong. A weak model is a model that only approximates a part of the complete function the algorithm wishes to learn. The weak models will have high accuracy on one or more parts of the data, but low accuracy when tested on the complete data set. Subsequently, the idea is that multiple weak learners specialising in parts of the problem are together able to create one strong learner approximating the complete function. In such a model multiple weak learners work together to make up for each others' weaknesses [22].

2.1.4 Immune Systems

The *immune system (IS)* is arguably one of nature's most highly adaptive, distributed and self-organising systems [35]. It is has the property of being able to recognise anomalies; something that deviates from the common rule. This ability to differentiate between organisms that do and do not belong in the body, gives the IS its inherent classification properties. Through repeated exposure to the same anomalies the IS is able to adapt and evolve its cells to combat secondary infections at much higher rates than the original infection [18].

The Natural Immune System

The natural IS is the body's natural way of dealing with intruders and otherwise anything that does not fit into the norm of the system. Generally, the IS is split into two subsections of immunity; namely *innate* and *adaptive* immune responses. Unlike the adaptive IS, the innate IS does not target any specific intruder, but rather anything it deems a pathogen (infectious microorganism) [35]. These innate immune responses is the product of millions of years of evolution and does not change during the lifetime of the host [18]. Additionally, it operates as a controlling organ for the rest of the IS, and plays a crucial role in administering and triggering immune responses [35].

While the innate IS is sufficient at protecting the body against microorganisms with certain common molecular patterns it is not able to protect it against anything that has not been observed before. Pathogens evolve while the innate IS does not. This is why in addition to the innate IS the adaptive IS is needed. This part of the IS has evolved to handle what the innate IS is not able to recognise. While the innate IS target any foreign microorganism it is able to recognise, the adaptive system target specific intruders with its variety of immune cells. Furthermore, each cell is able to recognise a certain *antigen (AG)* which are foreign substances like toxins and enzymes, or more specifically, the epitope or antigenic determinant present on the invading microorganism. Upon repeated exposure to such AGs the adaptive IS is in fact able to adapt and evolve new and specialised *antibodies (ABs)*. ABs, with paratopes as AG binding sites, are the parts of the immune cell that are able to recognise and bind to AGs [31].

The adaptive IS employs two different types of lymphocytes as immune cells, namely T-cells and B-cells. T-cells function as the controlling cells of the adaptive IS, as they initiate the attack, while B-cells are the immune cells that produce ABs that actually bind to the AG [35].

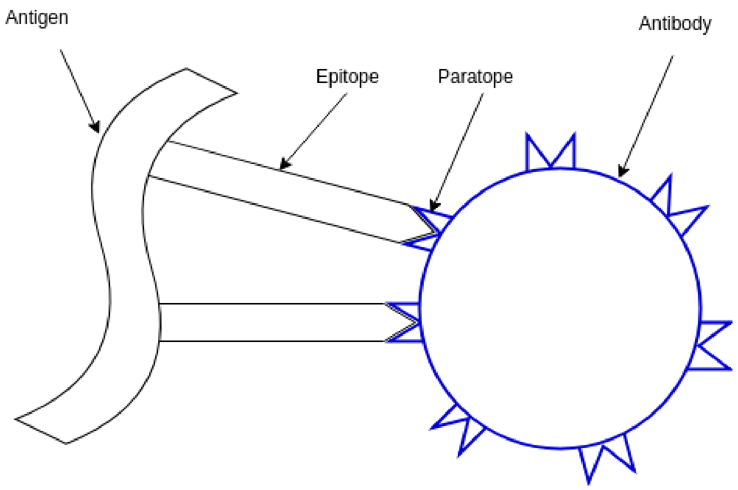


Figure 2.5: Illustration of an AB's connection to an AG

Biological Operations in the Natural Immune System

As mentioned, upon repeated exposure to new patterns in invading microorganisms the ABs in the adaptive IS gradually evolve to recognise these intruders. This is done through a process called *affinity maturation*. *Affinity* is a measure for how well an AB is able to recognise and bind to a specific AG. Higher affinity means higher strength of the bond between the AB's epitope and the AG's paratope [6]. Figure 2.5 illustrates an AB-AG interaction through the AB's epitopes connecting to the AG's paratopes. The binding strength, i.e. the affinity between the AG and the AB, increases with the amount of connections. Furthermore, affinity maturation occurs during exposure to an AG, when B-cells with higher affinity ABs are stimulated by the specific AG to proliferate (divide) with a rate proportional to the corresponding affinity [38]. Subsequently, This either turns the B-cells into what is called plasma cells, which secrets one type of ABs, or *memory cells* that constitutes the immune memory [35]. Further, The memory cells are a type of long-lived B-cell that continue to exist in the body even after the AGs and the original B-cells have died. These usually high affinity cells circulate through the body, and when exposed to antigenic stimulus of the same or similar AG from when they were first created, they differentiate into large lymphocytes that produce high-affinity cells. This in turn combats the returning AG at a much higher rate than during the first exposure. These memory cells allow for the IS to learn and protect against secondary infections [35].

During the proliferation process the cloned B-cells are subjected to affinity maturation, which constitutes some somatic hypermuation [6] along with strong selective pressure that allows the IS to learn. Somatic hypermutation mutate the variable regions of the AB genes affecting its capability of binding to certain AGs, which either increase or decrease the affinity of the AB. During the hypermutation process the B-cells undergo a stochastic alteration of their receptors in an attempt to generate ABs of higher affinity specialising on the invading AG. As mentioned above, B-cells of higher affinity are stimulated to proliferate at greater rates, which in turn creates a selective pressure where more capable (higher affinity) B-cells are produced in greater numbers, while lower affinity cells are gradually phased out. Additionally, the mutation process makes the affinity of these cells converge to a point where there are many and capable enough B-cells to combat the current AG invasion [35]. The theory that only the cells being able to recognise the AG proliferate, and therefore being selected over those that do not, is what is referred to as the *clonal selection principle* [1].

Artificial Immune Systems

Artificial immune systems (AIS) are algorithms that solve computationally complex problems by employing immune principles. They are population based and evolution driven with many similar aspects to GAs (see section 2.1.1). Furthermore, in such systems one or several IS-inspired components and theories are employed. However, so far no system employ all the principles of the natural IS and thus there is currently no algorithm that creates a complete abstraction [15]. The principles used typically have elements in common with evolutionary methods, but at the same time exhibit some peculiarities that make them useful for certain unique applications [10].

One concept typically employed in AISs is *shape space*. Shape space is a common abstraction that allows the system to interpret the process of ABs recognising and connecting to AG in terms of geometric properties of shape and position. This is further visualised in figure 2.6 where AB-AG interactions are represented by their coordinates and distances in shape space. The aim of such an abstraction is to simplify the interaction process, at the same time as some similarity to a biological system is kept. Further, in natural ISs the recognition process is based on the complementarity of the geometric shapes between the surface of the AG and the AB receptors, as well as the electric charge distribution on parts of the surfaces [10].

A simplified view of the recognition process is to assume that an AB recognises and connects to all AGs within some threshold length from its centre. Mathematically, one can abstract and simplify the representation of this interaction by representing the properties of the AG and AB necessary in determining the degree of interaction between them, i.e the binding region of the AB, as a list on n parameters. This list is referred to as the *generalised shape* of the AB, stating its position in shape space. What set of values is used to define the generalised shape also determines the complexity of the AB representation and, subsequently, its interaction with AGs, which in turn is highly dependent on the needs of the AIS model adopted. Using the generalised shape we can place the ABs, as well as the AGs, as points in the n-dimensional shape space, indicated by the black circles and squares in figure 2.6 [10].

Only representing the AGs and ABs as points in shape space is not enough for determining the degree of interaction between them. For this a representation will

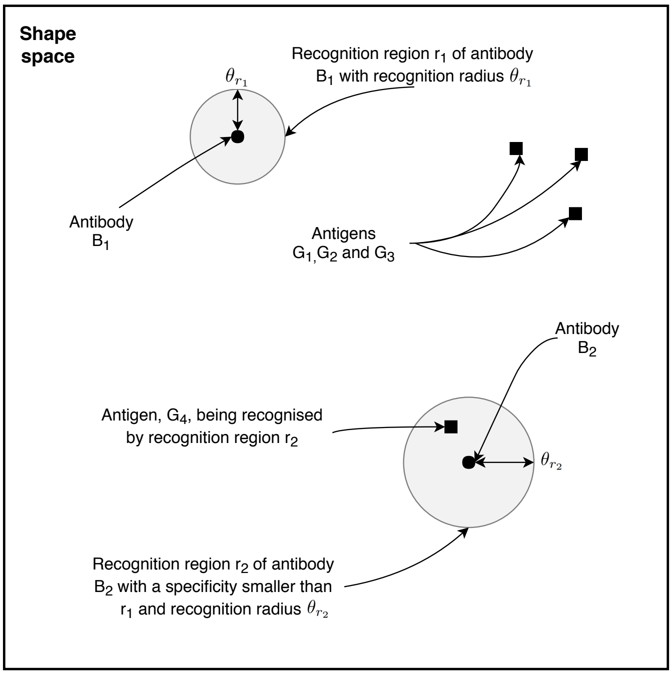

Shape space
Recognition region r1 of antibody B1 with recognition radius θ_{r_1}
θ_{r_1}
Antibody B1
Antigens G1,G2 and G3
Antibody B2
Antigen, G4, being recognised by recognition region r2
θ_{r_2}
Recognition region r2 of antibody B2 with a specificity smaller than r1 and recognition radius θ_{r_2}

Figure 2.6: Visualisation of the shape space.

be needed of how and under what conditions ABs recognise AGs. As mentioned, in an IS the degree of interaction is determined by the complementary of the interacting molecules. However, in an AIS, these physical properties can be ignored, and simply look at the interaction from an abstract point of view. In such a view there is only a need to look at how well the ABs and AGs match each others' position in shape space, namely to what degree their attributes overlap. In determining the degree of overlap the concept of *recognition region (RR)* is employed. In figure 2.6 the RRs are shown as grey circles surrounding the ABs. An AB recognises and connects to all AGs within its RR. To determine what is within the RR of the AB a similarity or distance function must be defined. For some AB, G_i, and some AG, B_j, the distance and therefore the similarity between the pair in shape space, is defined as the function $d(G_i,B_j)$. Here, a zero value means a perfect match, and increasingly higher values means decreasingly similar AG-AB pairs. Alternatively, a complementary similarity function, $s(G_i,B_j)$, can be defined to give increasing values to increasingly overlapping AB-AG pairs [10].

Typically, some threshold, θ_{r_j} is defined for each individual AB, B_j, to determine if an AG, G_i, is within its RR, r_j, where $i = 1, 2, ..., m$ and $j = 1, 2, ..., n$. More precisely; if $d(G_i,B_j) < \theta_{r_j}$ then G_i is said to be within the RR of B_j. This is illustrated in figure 2.6, where the RR of AB B_2 contains an AG, G_4. Subsequently, when the AG is within the RR of the AB they are said to be interacting, and the distance between the two can be used in further calculations for determining the affinity between them. In this example, θ_{r_j} correspond to the radius of a spherical RR, where smaller values means a smaller RR and therefore an AB of higher *specificity*, capable of recognising a more specific set of AGs [10]. The concept of specificity is furthermore illustrated in figure 2.6 where θ_{r_2} is larger than θ_{r_1} and therefore less specific.

What ABs are connected to what AGs and how well these ABs solve the problem addressed by the algorithm determines how the ABs change in order adapt and more correctly recognise as many of the invading AG as possible. Additionally, the total area covered in the shape space by the unified RRs of all the ABs determines what AGs can be recognised. This area is also what is referred to as the coverage of the *immune repertoire*, which here is being the set of all ABs. As there are physical constraints that limit the number of possible configurations for the n parameters defining the AGs, the size of the shape space is also consequently finite. Subsequently, a finite number of distinct ABs with corresponding RRs is in theory able to recognise all possible AGs [10].

2.2 Sate of the Art

The following section presents the state of the art regarding the AIS and IGA techniques. More specifically, configurations and results of IGA will be discussed in section 2.2.1, followed by a discussion of AIS, its implementations, challenges and solutions in section 2.2.2.

2.2.1 Configurations, Applications and Results of the Island Model

The island model has commonly been used for various optimisation problems such as job shop scheduling problems, feature selection and clustering. The technique has shown promising results on both convergence and quality of outcome [40].

Island Model Applications

Corcoran et al. [5] compares different types of GAs, both serial and parallel, including the island model. Their results indicate that the serially running GA provides as good, or equal, results to the IGA version of the algorithm when looking at the smallest problems. However, when compared to the most complex problems, the IGA outperforms the serial GA in both time and quality of result.

A study presented by Rahman [30] compares three different GAs for attribute reduction; serial GA, IGA without migration and IGA with migration. Their results also shows that the IGA with migration outperforms the other two, both in time and quality of the result. Such results are likely due to the parallel and separable nature of IGA. Further, performance is improved by preserving diversity in the different sub-populations, while periodic migration enhances the quality of the selection process [17]. Additionally, Huang states in [17] that distributed GAs like IGA are often more efficient than a serial GAs even when the algorithm runs on a single processor. Subsequently, a serial IGA will often find the optimal solution faster than a serial GA. Furthermore, other studies also show that it is possible to get linear speedups for non-linear problems, using a distributed GAs as shown by Neves et al. [27].

Whitley et al. [40] also states that the island model has a natural tendency to exploit the separable nature of problems. Subsequently, linearly separable problems have especially good convergence rates. Consequently, each island is able to evolve solutions proficient at different parts of a problem, before being exchanged and combined through migration to create even better solutions.

Number of Islands and Population Size

One of the main properties of the island model is the enhanced diversity achieved from employing several smaller and communicating subpopulations (see section 2.1.2). However, the number of islands employed is important for the result and can be decided in several different ways. For instance, the number of islands in a parallel IGA could be the same as the number of available processors, as shown in by Rahman [30]. Alternatively, a dynamic approach like what is presented by Meng et al. [25] could be adopted where islands are created and removed dynamically as needed. Finally, a fixed set of islands set by an algorithm parameter could also successfully be employed. This approach is favoured for its simplicity and being very configurable, as presented by Gozali [13], but comes at the cost of additional parameter tuning.

Areibi et al. [16] presents an IGA for static and dynamic optimisation problems. Their results show clear correlation between the number of islands and quality of the solution, where an increase in islands (and subsequently a higher total population) improves the solution. However, this approach results in increased computational cost as the number of islands increase.

As stated by Whitley et al. [40] the population size of the islands will impact the convergence of the algorithm. In a typical GA, a large population with high diversity will converge slower in comparison to one with a small population. Further, this is also true for the island model, as an IGA where islands have large populations will converge slower than an IGA with smaller populations. Subsequently, it is important to effectively balance the total population size and the number of islands employed, as smaller islands converge faster, while larger islands are able to maintain greater degrees of diversity.

Impact of Island Topology

The island topology employed has a large impact on how fast genetic material is shared between sub-populations (see section 2.1.2). Further, Huang [17] presents promising results using a *directed ring topology* where migration routes form a directed cyclic graph as in figure 2.7. In their proposed algorithm increasing the number of islands subsequently increases the total population. Through this topology, together with other components, the algorithm's runtimes and quality of the result are improved. Further, this topology is very efficient as the number of migration routes grows linearly with the number of islands. It only needs to perform two more migrations for each island added, as opposed to the fully connected model where the number of migrations increase exponentially with the number of islands. Additionally, in the directed ring topology, an increase in the number of islands is also an increase the time it takes to share genetic material from $island_1$ to $island_n$, as it takes a minimum of $n - 1$ migrations for genetic material to reach n from 1. This is illustrated in figure 2.7 where it takes a minimum of three migrations for genetic material to reach island d from island a. Increasing the number of islands may make the algorithm converge slower, while keep heightened levels of diversity within the population. Furthermore, the study presented by Areibi et al. [16] also use a directed ring topology, but where a fixed population size is divided across multiple islands. Also here the topology is shown to improve the quality of the final solution. Further, a directed ring topology is able to vary the level of *genetic isolation* and *interconnection* between islands. Subsequently, smaller islands numbers will make the topology more interconnected, while higher numbers increases the isolation levels because of the extra migrations needed for genetic material to reach all islands.

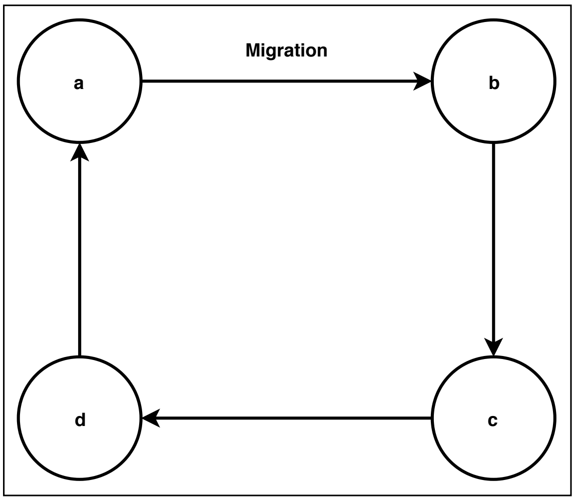


Figure 2.7: Directed ring topology.

Gozali et al. [13] presents promising results with a star-shaped island topology, which is also referred to as a master-slave topology (see figure 2.3 (b)). Three slave islands run three heterogeneous GAs and the *master island* has the role of controlling the migration between the *salve islands.* Due to employing heterogeneous slave islands the islands operate differently and subsequently migration is implemented asynchronously and controlled by the master island. Further, the algorithm is able to keep a high diversity due to the implementation of the master, which keeps the other islands from directly communicating with each other, only receiving selected genetic material through the master.

Another approach to island topology is a dynamic design presented by Meng et al. [25]. Here, an algorithm called DIM-SP (Dynamic Island Model based on Spectral Clustering) is proposed, which is initialised with only one island. Through successive iterations sub-populations create and migrate to new islands based on similarities between their individuals. This, refereed to as central clustering, is repeated until the last iteration, where all islands are merged together. Further, Meng et al. compare DIM-SP to three other IGAs with three different topologies; namely the fully-connected, star-shaped and ring-shaped topologies. Further, DIM-SP achieved the best results, followed by the ring topology, the star-shaped topology and lastly the fully-connected topology. Further, results indicate that the ring-topology generally performs better than star- and fully-connected topologies. This is because the ring-model is able to maintain a higher diversity over time due to its isolating topological features. On the other hand, the other two models show tendencies where the islands prematurely converge and become homogenised, as a consequence of the number of connections between them.

Impact and Approaches to Migration

As mentioned in section 2.1.2, migration corresponds to the sharing of genetic material between sub-populations. However, it is possible to implement an IGA without migration, but studies indicate that it does not perform as well. Subsequently, without migration the sub-populations show a tendency to prematurely converge as not enough diversity is kept [30, 5, 17].

IGAs has proven to perform well with migration [17, 30, 5, 20, 27]. Through migration, IGAs are able to exploit the rapid convergence of small populations, while simultaneously maintaining sufficient amounts of diversity. Consequently, the search space is more thoroughly searched and better solutions may be found [17].

As mentioned, migration has three parameters; MR, MF and policy (see section 2.1.2). However, if the MR and MF are too high, the sub-population will not be able to evolve in an partially isolated environment. Subsequently, successive migrations makes islands increasingly similar to each other, which lowers the level of diversity in the total population. Consequently, also at this point the sub-populations prematurely converge. As a result, both too low and too high MRs and MFs will lead to premature convergence, indicating that a healthy, problem dependent, balance is needed [25, 17].

Migration policies may be implemented in several different ways. One such example is presented by Merelo et al. [3]. Their study suggests to migrate multiple individuals at a time, while having the receiving population decide which individuals should be migrated, in order for the island to promote its own diversity. Subsequently, each island chooses the individuals most different from its own. This policy produces promising results as because of its enhanced diversity. Furthermore, another example of a promising migration policy is presented by Gong et al. [11] where an elitist approach is implemented, resulting in only the best individual on an island being migrated. Migration occurs when a new best individual is found, which at that point is sent to a random neighbouring island. Further, the receiving population replaces its worst individual with the new immigrant. Additionally, Raman et al. [30] presents a similar well-performing elitist policy, where at each island a percentage of the best individuals are selected for migration. On the other hand, the receiving population replaces the same percentage of its worst individuals with the new immigrants. However, as opposed to Gong et al., migration does not occur once a new best individual is found, but rather after a set time, long enough to allow development of several diverse individuals of high quality at each island. Elitist strategies are quite common as indicated by several other studies [16, 29, 17] that successfully employ different elitist migration policies. However, as stated by Huang [17], if an island only sends its best individuals and replace its worst, it may lead to premature convergence. This is increasingly true when migration is performed often. Alternatively, a random migration scheme can be employed that counteracts premature convergence through enhanced diversity, while simultaneously being more effective as a result of reduced sorting needed.

Cantú-Paz [4] presents a study comparing elitist and random migration policies. The study shows that migrating the best individuals and removing the worst generally achieves the fastest convergence. On the other hand, both sending and removing random individuals has the slowest convergence as a consequence of having the highest diversity, but risks never converging at all. Alternatively, combinations of elitist and random approaches can be employed, i.e. sending random and removing the worst or sending the best and removing random. Such policies may be able achieve a reasonable compromise between the approaches, in terms of diversity and convergence. However, what performs better is generally determined by the needs of the problem solving process.

2.2.2 Artificial Immune Systems - Implementations, Challenges and Solutions

AISs have gotten attention in various fields in recent years, as it has been shown to have success in a some application areas. These areas include clustering and classification, anomaly detection, computer security, numeric function optimisation, combinatoric optimisation and learning [35, 15].

Algorithms

A substantial amount of different AIS algorithms have been proposed in literature. They all differ in the amount of IS and biologically inspired concepts employed. However, hypermutation, affinity maturation and clonal selection are important IS concepts that are often implemented [6, 38, 8]. Generally, an AIS for classification seeks to evolve the bests set of ABs able to correctly classify a set of AGs. Subsequently, AISs recognising typically population based and evolution driven algorithms, sharing similarities to GAs (see section 2.1.1) and allowing a wide variety of biologically inspired techniques to be employed.

CLONALG

One of the most iconic AIS algorithms is *CLONALG*, a biologically plausible AIS algorithm which has inspired the work of many subsequent AISs. CLONALG is based on the clonal selection principle (see section 2.1.4) and employ a few key steps inspired from the IS. These include maintaining a set of memory cells capable of recognising and selecting from the generated ABs, cloning and mutation of stimulated ABs (ABs recognising AGs), as well as removal or death of non-stimulated ABs. When ABs are selected they are subjected to affinity maturation (see section 2.1.4) in in order to successively create better ABs [6]. Finally, while CLONALG is known for its ability to solve both optimisation and pattern recognition problems it generally does not perform as well on the latter. However, several alternative CLONALG derivatives have been proposed that improves on the shortcomings of the original [32].

AIRS

AIRS - Artificial Immune Recognition System is a recognised AIS algorithm for classification. AIRS, as well as CLONALG, is heavily based on the idea that immunological metaphors can be used to create an effective learning algorithm, through clonal selection and affinity maturation. Further, AIRS follows the shape space model where every receptor has its own fixed position in feature space and bindings between AB-AG pairs are calculated as euclidean distances (see section 2.1.2). Additionally, it abandons the immune network model where ABs can interact with each other and, as such, the ABS are evolved independently from each other. Furthermore, ABs are cloned at a rate proportional to the strength of their bindings, with stronger bindings giving a higher degree of stimulation to the AB. Additionally, during the training loop, the least stimulated ABs are continually removed as better performing ABs emerge from a continuous process of cloning and mutation. These properties has been shown to perform well for classification in a supervised learning system, performing on par with or better than several established algorithms [38, 39].

VALIS

VALIS - Vote-Allocating Immune System is an AIS algorithm that operates by allowing all ABs that contain a given AG within its RR to connect to said AG. Further, when a connection is realised, a class assigned to each AB is cast as a vote for the what class the AG belongs to, as a means of classification. Consequently, the different AB classes cast are the same as the classes in the data set used. Further, the strength of the AB's connection with the AG determines the impact of the vote and the class with the highest tally, when all connected ABs have cast their votes, is the predicted class of the AG. Furthermore, at every iteration of the algorithm the evolved ABs are evaluated by their fitness score. For this VALIS employs a flexible fitness function consisting of several components involving accuracy, AB-AG interaction and resource sharing, which changes the dynamics of the evolution according to how they are weighted [19].

Other AIS algorithms like CLONALG and AIRS does not rely on independent voting based on binding weights, but rather on the most common class of the k-nearest neighbours (KNN). Therefore, VALIS employs, arguably, a more biologically plausible method as it relies on local AB-AG interactions and not distance-based sorting of AGs. Subsequently, in a voting process, only ABs that are actually connected to the AG is used when classifying it, as opposed to looking at all the closest ABs regardless of there existing a connection between them [19]. On the other hand, VALIS uses a crossover approach when proliferating ABs, which in turn is less biologically plausible than the proliferation process of AIRS and CLONALG.

The VALIS algorithm employs a training scheme similar to traditional GAs, where antibodies are created through crossover and mutation and a set population of ABs are selected and employed at each iteration (see section 2.1.1). Additionally, both parent and survivor selection are used during an iteration. On one hand, for survivor selection, the worst individuals are replaced by the newly generated ABs at each iteration. On the other hand, for parent selection, fitness proportionate selection is used (see section 2.1.1) which reportedly gives good results in terms of effective convergence [19].

AISLFS and Implications of Shape Space

In section 2.1.4 the notion of shape space and spherical RRs were introduced. Spherical RRs are simple, yet effective, to use as they can be implicitly defined through a radius. VALIS is an example of an algorithm using hypersphere RRs. However, a spherical RRs causes some issues because as the number of dimensions increase its volume approaches zero. Further, dimensionality problems are common in machine learning, where computation that yields good results in a low-dimensional spaces, becomes intractable in higher dimensions. In fact, it is such a common problem that it is often referred to as *the curse of dimensionality* [23, 8].

The curse indicates that AIS algorithms with spherical RRs may become decreasingly effective when applied to data sets containing a high number of features. Fortunately, some approaches has been proposed to address this problem as in [23, 8]. Further, Dudek [8] introduces an algorithm called *AISLFS - Artificial Immune System Local Feature Selection*, where an AIS using *local feature selection (LFS)* is implemented. LFS operates on the idea of using different subsets of the problem's features in different parts of the search space. The features employed in each AB is a subset of features derived from the complete shape space of the problem. This means that each AB operate in an specialised l-dimensional subspace where l is the length of the subset feature vector, derived from the complete feature space of n features where $n \geq l$. Further, the AB only considers this feature subset when classifying cases. Subsequently, the use of subspaces decreases the impact of curse of dimensionality, as most ABs operate on lower than n features. Further, this is illustrated in figure 2.8 where a set of ABs is first shown to use the whole two-dimensional shape space (a), and another set is shown to have different sub-spaces of either 1 or 2 dimensions, using different parts of the shape space (b).

The subspace abstraction is somewhat closer to a real IS as each derived feature vector represents the specialised receptor of a B-cell. In biology, the epitope is an

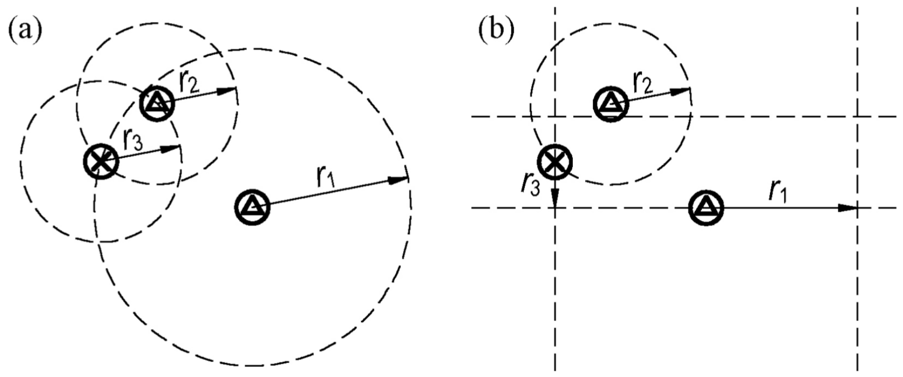


Figure 2.8: All ABs (small circles) visualised with their RRs (dashed lines) of radius r_k where k = 1,2,3. (a) All ABs are using the whole 2-dimensional shape space of features {1,2}. (b) One AB of radius r_2 using the whole shape space {1,2}, another, r_1, using just {1} and the last one, r_3, uses just {2} (Adapted from [8]).

arbitrary discontinuous region on a the surface of the AG, with potentially large differences between epitopes in the area of the AG covered [23]. Therefore, the specialised receptors may also vary widely in the area covered. Subsequently, not representing each AB as a configuration of all the available features, but rather as a subset of related, but sufficiently different features, creates a biologically closer model [8].

The AISLFS algorithm employ no set population size and simply evolve memory cells of different feature subsets recognising AGs until convergence. Additionally, AISLFS employ an *apoptosis* mechanic where redundant ABs (ABs of equal feature values) are removed from the population after training. This mechanic, along with LFS, allow AISLFS to achieve good results on datasets with large and difficult feature spaces. Additionally, the LFS mechanic enables the algorithm to reduce the amount of data needed for classification by up to 99% [8].

The AIS Niche

As mentioned, AIS algorithms have shown promise by being able to solve increasingly complex problems of a wide variety of types. However, according to Hart et al. [15] there is little distinct value added to these fields by AIS. Furthermore, it is stated that that the value of using AIS over other methods is not clear, as there are no known problems that cannot be solved to an equal or better degree by other techniques. For instance, when used for classification or clustering, an AIS provides features like feature extraction, recognition and learning. While these are necessary features for any machine learning task, it is also something that is already well-performing and integrated into many other techniques.

Because of this, the AIS field has no clear niche where it is the best tool for the job [15]. However, the authors suggest some improvements that would move AIS closer to their biological counter-part. To this end, a set of principles in order to fulfil a more defined, unique niche is presented. These principles include a combination of properties from the biological IS that have yet only partially been implemented in an AIS. The principles are the following:

1. It should be *embodied*, the IS does not act in isolation.
2. It should exhibit *homeostasis* - a relatively stable equilibrium between the components.
3. It should benefit from interactions between innate and adaptive IS.
4. It should consist of multiple, heterogeneous interacting, communicating components.
5. It should contain components that can be easily and naturally distributed.
6. It will be required to perform life-long learning.

Ideally all the above principles should be implemented in an complete AIS, but work arguably still remain in fully achieving each one. The fifth principle is especially important for this work, as the proposed algorithm exploits the inherent and naturally distributed nature of an AIS. Some work has already been done on this in regards to AIS classification by Watkins et al. [37]. In their work they present a parallel implementation of AIRS where training data is scattered over several processors. While accuracy improvements from this were small, the approach achieved good improvements in terms of parallel efficiency. Additionally, Watkins [36] takes the AIRS implementation one step further by implementing a fully distributed version of the AIRS algorithm. Subsequently, the training process on different parts of the data set is completely decentralised. While the improvements from distributing the evolution were subtle, it was shown that certain segments of the network became more proficient at correctly classifying certain classes over others. Some segments excelled at classifying cases while others were hopeless. Futhermore, Hart et al [15] argues that a distributed approach is both necessary to create a distinct niche for AIS classifiers and at the same time, much more biologically plausible than the non-distributed alternative.

Alternative Recognition Regionss

Spherical RRs are easy and efficient to use because of their simplicity in definition and computation. However, several other geometric shapes may be employed for the ABs RR. Other shapes have shown to give good results under certain conditions for optimisation problems, as shown by Hart [14]. Furthermore, different AB RRs may perform better on some problems than others. This is illustrated in figure 2.9, where an AB with a cross-shaped RR is able to separate the two classes of data points, while the AB with a circular region is not.

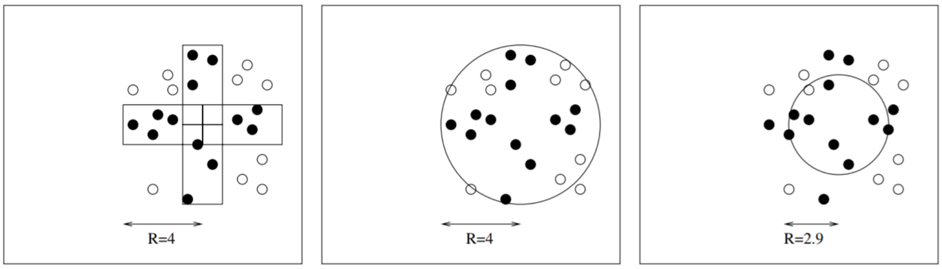


Figure 2.9: Example problem where the choice of RR shape is important (adapted from [14]).

Hart [14] further evaluates different RRs in a classical AIS, as well as in an idiotypic immune network, where ABs are able to interact with both AGs and other ABs. Further, it is shown that different RRs result in networks with different sizes and dynamics. Additionally, these networks show a varying degree of being able to tolerate AGs, given the different ranges of their recognition radius. Subsequently, the size and shape of the AB RR may affect both results and convergence of the algorithm.

For AIS classification, research has shown to give good results by employing dynamic RRs as in Ozsen et al. [28], where evolvable elliptical regions are used. Here, three different mutation operators are employed on the RRs that consist of changing the centre, length and orientation of the elipsis. The resulting algorithm shows no great improvements in regards to solving linearly separable data sets, but on complex nonlinear data it appears to perform well and sometimes even better than other algorithms in regards to both training times and accuracies. Furthermore, another example of an algorithm using alternative RRs is AISLFS, which successfully employ a variety of different AB RR shapes that include spheres, cubes and cylinders [8]. However, many of these are only employed for a certain subset dimensionality as some shapes are inefficient when applied in higher dimensions (see section 5.3.2).

2.2.3 Summary of State of the Art

AIS algorithms can be adapted to solve classification problems of both linear and non-linear nature. Such classification algorithms generally consists of evolving a set of ABs able to recognise, adapt and interact with a set of AGs. Several algorithms have been proposed that implement parts of the IS in different, yet similar, ways and it is shown that AIS algorithms are able to perform just as well than several established classification methods on some problems. However, some AIS algorithms are dependent on evolving ABs in higher-dimensional spaces where a connection is only realised between an AB-AG pair when the AG is within the hyperspherical RR of the AB. This property has the unfortunate effect of increasingly subjecting AIS algorithms to the curse of dimensionality, as the dimensions in the dataset increase, leading to the RR's volume approaching zero. However, these issues can for be addressed through employing dimensionality-reducing methods, such as LFS. Finally, while AISs have successfully been adapted to several different problems, they lack any defined niche. However, this could be solved by exploiting principles inherent to the IS, such as employing naturally distributed components.

IGA has successfully enhanced the performance of several algorithms. This is done through maintaining enhanced diversity and exploration by employing several separately evolving and migrating sub-populations. As a result, convergence and exploration can be regulated through the number of islands and migration parameters employed. Through this, IGAs are reportedly able to achieve both better results and lower run times than its non-distributed counterparts. However, not all IGAs enhance performance as worsening results may be a consequence of migration and high levels of selection pressure making each island homogeneous and prematurely converge. Subsequently, good results typically come from effective compromises between genetic diversity and selection pressure on the islands.

Chapter 3

Algorithmic Model

The following chapter introduces the algorithmic model of the proposed algorithm. Section 3.1 introduces how the IGA will be combined with AIS through topologies, algorithm flowchart and migration policies, as well as reasons for selecting the components used. Furthermore, section 3.2 presents the configuration of the AISs on each island, consisting for parameters, initialisation, chromosomes, crossover and mutations, selection mechanisms, AB and population fitness evaluations and, finally, voting and classification.

3.1 Combining IGA and AIS

In principle, to create a hybrid algorithm involving AIS and IGA, each island runs its own AIS that evolves its population of ABs independently, except for the periodic swapping of genetic material (ABs) through migration. This is done in order to exploit IGA properties of enhanced efficiency and exploration (see section 2.2.1) in order to improve AIS classification. Subsequently, as the proposed algorithm evolves a several interacting AISs, it is named *MAIM - Multiple Artificial Immune Model.*

3.1.1 Topology

The key consideration in the choice of the topology was to select a computationally efficient architecture that could provide flexibility between genetic isolation and interconnection. Figure 3.1 illustrates a novel two layered master slave directed ring topology chosen for the MAIM algorithm.

A directed ring topology was chosen for the first layer. These islands are the *slaves* in the model. The second layer is a single island which acts as the *master.* Unlike the standard master slave topology, slaves may migrate to other slaves directly rather than through the master. The slaves are responsible for exploration of the search space through selection, crossover and mutation of the ABs of their island populations, IPs. The master is responsible for combining the IPs to form

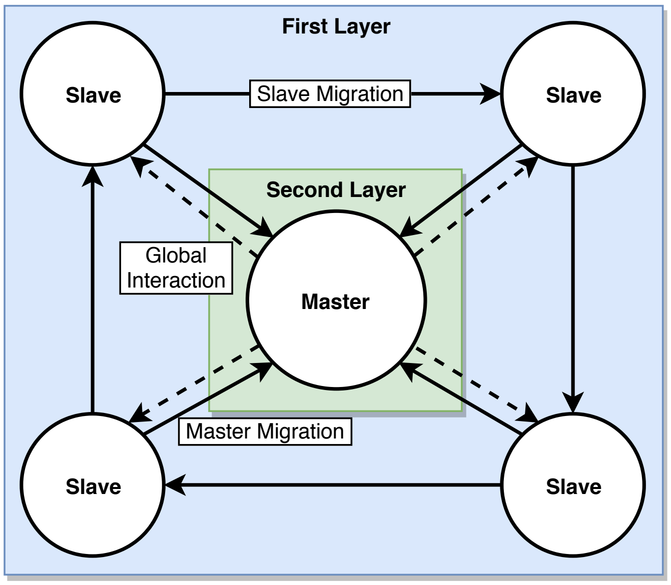


Figure 3.1: Directed ring topology with master island.

the new population P and applies fitness to the population, $F(P)$ (see section 3.2.7). The master island holds a copy of the best population found so far. This novel approach of combining two established topologies was selected in order to achieve the benefits of both: The master-slave topology, for its ability to direct the search and combine solutions, and the directed ring model for its advantages of efficiency, diversity and flexibility (see section 2.2.1).

Traditionally in an IGA with N islands there are $(N \cdot IP)$ potential solutions at every iteration. However, in AIS, one individual is an AB and it takes a population of ABs to create a complete solution, meaning that there would be N solutions. However, in this work, the population P is spread across all the slaves, where each island holds (P/N) individuals and there is only one solution at any iteration. Each island contributes to the combined solution P by evolving a partial solution. By evolving only partial solutions the potential issue of smaller sub-populations prematurely converging is greatly reduced while simultaneously enabling increased efficiency due to having smaller IPs.

It should be noted that there is a second fitness evaluation, which is the fitness of an AB, $F(b)$ (see section 3.2.7), that is calculated for each AB on every island. Optionally, through a global $AB\_interaction$ parameter, the master can guide the selection of ABs at each island (see section 3.2.7). There is no partial fitness function, i.e. fitness of the island population and, therefore, the term partial solution is used loosely. The slaves may be said to act as *weak learners* creating partial solutions whilst the master may be thought of as a *strong learner*, making it similar to an ensemble method (see section 2.1.3).

3.1.2 Proposed Algorithm Flow Chart

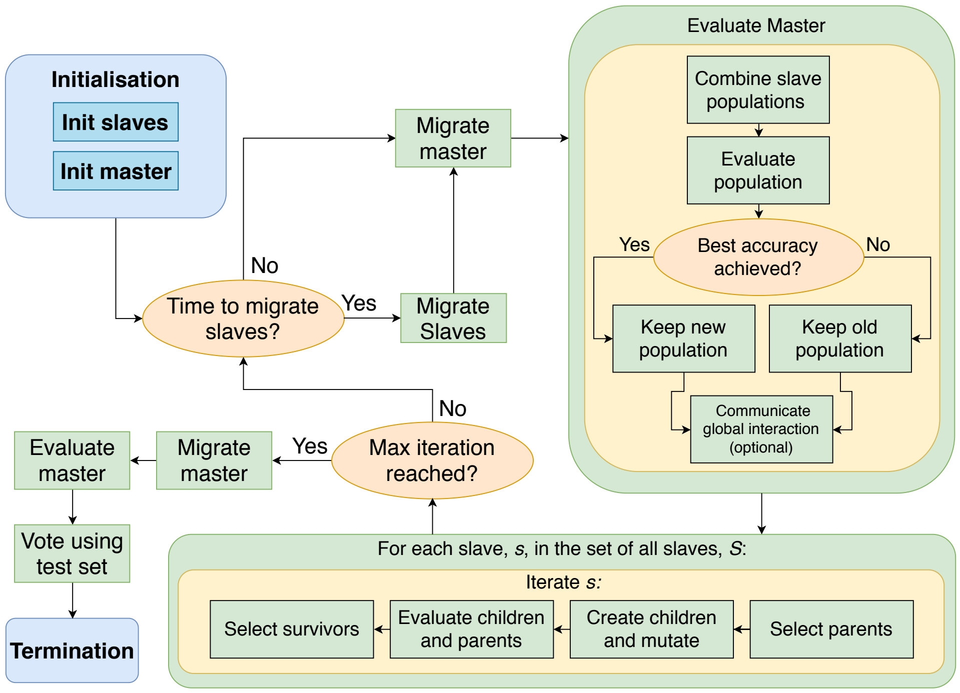


Figure 3.2: Flowchart showing the proposed MAIM algorithm.

Figure 3.2 presents an overview of the MAIM algorithm. During initialisation, parameters controlling the master and the islands are applied (see section 3.2), and an initial population of ABs is randomly generated for each island. AGs, being training samples, are split into complete training and test sets. The complete training set is further split into training and validation sets. Every slave is supplied with the training set as their AG population, while the master is supplied with the validation set. Therefore, slaves are tasked with evolving an AB population using the training set, while P is evaluated by the master using the validation set.

After initialisation, the sub-populations are migrated to the master, combined to form the *current population* and evaluated. The evaluation involves a voting process that assigns classes to AGs in the validation set based on the ABs in P and the classes assigned to all AGs are compared to their true classes to calculate $F(P)$ (see section 3.2.7). The current population then becomes the best population so far. Optionally, the global *AB_interaction* parameter (see section 3.2.7) from the master is sent to the slaves to affect the next iteration of the AISs. Further, during the parallel iterations of the AISs (on every slave island), parent ABs are selected, crossover and mutation are applied and the individual ABs of both the parent and the child population are evaluated through $F(b)$ (see section 3.2.7). A new IP is then selected from the combined parent and child population for each island (see section 3.2.6). If the maximum iterations is not reached and it is not time to migrate between slaves (see section 3.1.3) the slaves again migrate their IPs to the master and the loop continues as before. Note that the evaluation at the master includes a comparison between the current population and the best so far where the best population is updated with the current population if it achieves a higher $F(P)$.

If it is time to migrate, as indicated by the MF (see table 3.1), all islands exchange x individuals (see section 3.1.3), before migrating their IPs to the master. When the *maximum iterations* is reached, a final migration between the slaves and the master is needed, the process of evaluate master is followed to ensure that the best population and its fitness is stored which subsequently is the final the solution. Finally, voting is conducted for the test set using the best population, and the resulting fitness is returned as a measure of the algorithm's generalisation ability upon termination.

3.1.3 Migration

As illustrated in figure 3.1, there are two different migration policies employed in the algorithm. These are the migration between the slave islands and migration of all the sub-populations from the slaves to the master. However, in this work it should be noted that copies of individuals are migrated and not the individuals themselves. Both migration policies are synchronous, meaning that all the islands migrate individuals between themselves in parallel and all the islands migrate their individuals to the master in parallel. A synchronous policy over a potential asynchronous policy was chosen for simplicity.

Slave Migration

As stated, many IGAs employ an elitist migration. However, such a strategy assumes that each individual represents a solution to the task in hand. In this work, an individual is one AB, one component of the complete solution. The quality of the individual may be assessed in terms of the accuracy of the AB or its fitness (see section 3.2.7). However, even if the AB is of high quality it does not indicate that it is a good solution to the problem, as it is only one part of a whole. Subsequently, there is likely little to gain from an elitist migration strategy based on these values as a component of low quality might very well be vital parts of the solution. Consequently, given the challenges of elitist migration and the benefits of random migration in terms of efficiency and enhancing diversity (see section 2.2.1), random migration was chosen between slaves.

Generally, slave migration on an islands involves cloning of x randomly selected individuals and transfer of the cloned individuals to the neighbouring island connected through migration. Further, two different random slave migration polices are proposed and tested for the algorithm.

The first alternative operate by having each slave island select random ABs and send them to its receiving neighbour without deleting anything. In the other policy a random set of ABs are selected for migration and another random set is selected for deletion. This is illustrated in figure 3.3 where the green individuals are selected for migration, the red individuals are selected for removal and the blue individuals are selected for both migration and removal. This is to illustrate that the selection of individuals to migrate and to remove, are two separate selection processes which both are random. The number of individuals to migrate, x, is defined by MR as explained in table 3.1. To keep a stable population size, the number of individuals to remove will always be the same as the number of migrants.

The second alternative is a policy presented specifically for the this work and relies on migrating x random individuals, but without deleting any. Subsequently after a round of migration each island has a population of $IP+x$. Thus $IP+x$ ABs are migrated to the master providing $N \cdot x$ more genetic material for the evaluation of $F(P)$ (see section 3.2.7) at this iteration. Thus, if the current P is better than the best P so far, the updated best will hold $P + (n \cdot x)$ ABs rather than P. In the following iteration of the AIS, each IP has the population $IP + x$ at the start but only IP ABs are selected for survival and the islands return to populations of size IP on the next iteration with the master returning to P the next time the best P is updated. Further, this approach was proposed because each island contain one partial solution to the problem, consisting of all its ABs, and not IP different solutions as in a traditional IGA. Subsequently, deletion of ABs may have severe effects on the partial solution by potentially removing vital ABs. Consequently, a random policy that simply increases the population for one iteration was instead proposed. The different policies are tested in section 4.5.3.

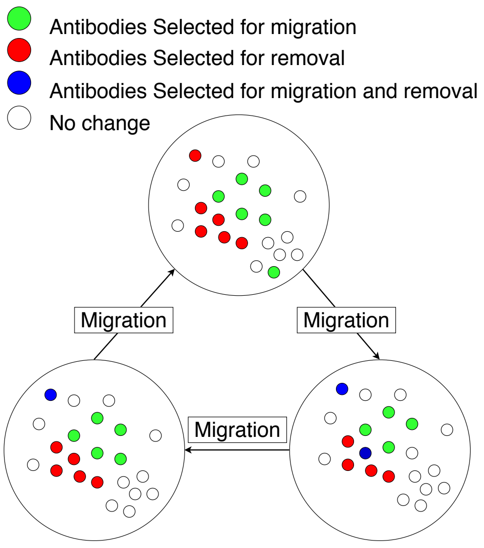

Antibodies Selected for migration
Antibodies Selected for removal
Antibodies Selected for migration and removal
No change
Migration
Migration
Migration

Figure 3.3: Visualisation of migration between slave islands.

Master Migration and Recombination

The master migration and recombination process is visualised in figure 3.4. The master island is here incorporating all the subpopulations from all slave islands into one complete solution. As mentioned, the master island contains both the current best population and the newly gathered one. After recombination the new population or solution is evaluated and if it attains a higher accuracy over the validation set than the best population, it is kept as the new current best.

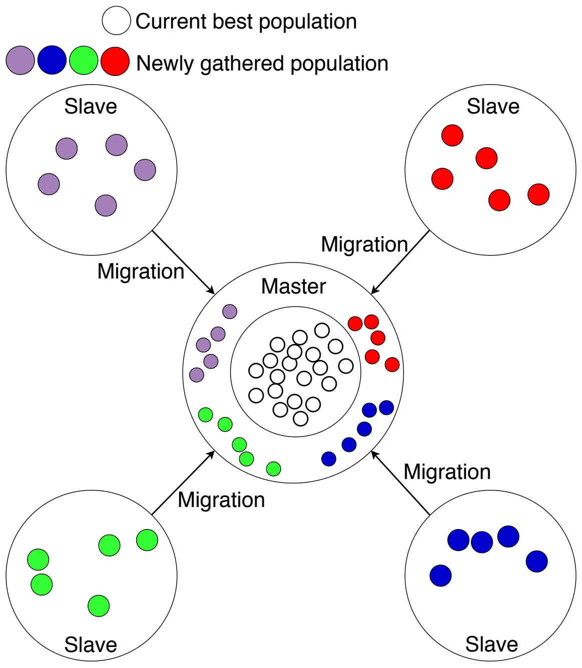


Figure 3.4: Visualisation of migration from slaves to master.

3.2 MAIM Components and Configurations

The AIS employed is inspired from the work in VALIS. More specifically, this means that it evolves a population of ABs at each iteration in order to classify a set of AGs. The AGs are created from the dataset passed to the proposed algorithm, as one AG is created from one data sample. Further, the AGs are initialised with normalised feature values between 0 and 1. AGs remain static over the course of the algorithm.

3.2.1 MAIM Parameters

The parameters of the MAIM algorithm are as presented and explained in table 3.1.

IGA Parameters	
Number of Slave Islands (N)	The number of slave islands to initialise and employ. The master island will always be initialised in addition to these.
Population Size (P)	The total population size. The master island have the full population size, while each slave island have a population size of: $$IP = \frac{P}{N} \tag{3.1}$$
Migration Rate (MR)	The percentage of the island's population that are to be selected for migration. For example, if the given island's population is 100 and the MR is 0.1, the number of ABs selected for migration is 10.
Migration Frequency (MF)	The rate at which migration occur between slave islands, which is every X iteration as determined by equation 3.2. $$X = \frac{1}{MF} \tag{3.2}$$
AIS Parameters	
Mutation Rate	The chance that a newly created AB is mutated.
Tournaments	The number of tournament rounds used in tournament selection (see 3.2.6).
General Algorithm Parameters	
Iterations	The total number iterations.
AGs	Set of AGs with normalised feature values created from the data set provided.
Feature-class Intervals	A set of feature intervals, i.e. the lowest and highest possible feature values, for each feature-class combination in the dataset (see section 3.2.3).
Classes	The set of different classes in the data set.
Training and test split (TS)	The percentage split of all AGs between the test and complete training sets. TS AGs are assigned to test and $1 - TS$ to training.
Validation split (VS)	The percentage split of the complete training set between training and validation sets. $1 - VS$ AGs are assigned to validation and VS to training.

Table 3.1: MAIM parameters.

3.2.2 Chromosome Structure

There are two different chromosome types employed, one for the AG and one for the AB. For the AB the chromosome consists of one feature vector, a class and a recognition radius, as visualised in figure 3.5. A hypersphere RR was selected for its efficiency and simplicity. Further, the RR is defined by the recognition radius while the feature vector corresponds to its position in feature space. Any AG which has a euclidean distance from an AB less than the recognition radius of the AB is said to be within its RR. Finally, the class determines what class the AB belongs to, as well as what class it will cast during the voting process (see section 3.2.7).

Similarly, the structure of the AG chromosome consists of a feature vector and a class label determining the true class of the AG, as MAIM is a supervised learning algorithm. Further, the AB and AG feature vectors are always of equal lengths, meaning that the structure of the AG chromosome will look like the feature vector and class in figure 3.5, but without the RR.

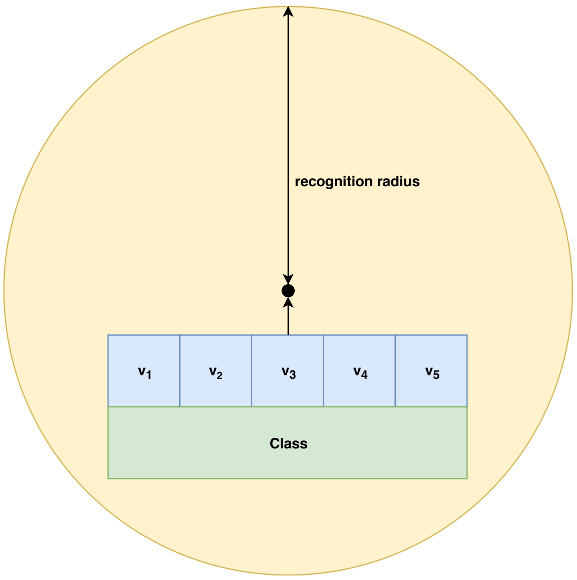


Figure 3.5: Visualisation of the AB chromosome.

The *affinity* between an AB, b, and an AG, g is defined as the inverse euclidean distance between them as long as the AG is within the RR of the AB and otherwise zero. This is defined as in equation 3.3 where $d(b, g)$ is the euclidean distance between the AB and the AG and r is the recognition radius.

$$W_{bg} = \begin{cases} \frac{1}{d(b,g)}, & \text{if } d(b,g) \leq r \\ 0, & \text{otherwise} \end{cases} \tag{3.3}$$

3.2.3 AIS Initialisation

As mentioned, each AIS is initialised with IP, ABs (see equation 3.1). Upon creating a new AB in the initialisation procedure, a random class from the set of AG classes (the different classes provided with the dataset used) is selected. All ABs have a fixed amount of features and each feature value is randomly selected from within a preset interval for each feature and class combination. Here, each feature has an interval for each class. Further, this means that if the classification problem has three different classes there are three different intervals for each feature, corresponding to each of the three classes. These intervals starts as 10% below the smallest value for that feature-class combination and ends at 10% above the highest value for the same feature-class combination. In this way the AB is either initialised somewhere within the sub-space where the AGs of its given class resides or slightly outside in order to create a smarter an slightly less random initialisation.

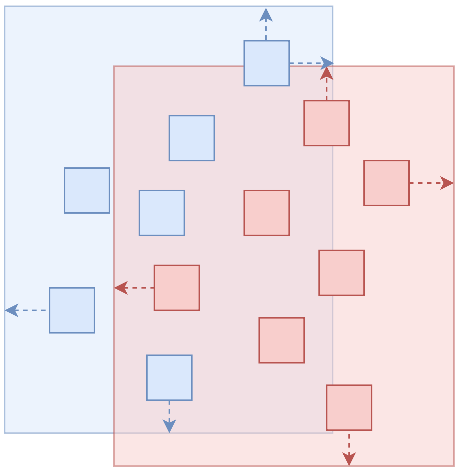


Figure 3.6: AB initialisation example.

Figure 3.6 illustrates the class initialisation area. AGs are shown as small squares and their class is depicted by their colour. Furthermore, the large squares corresponds to the sub-spaces of each class. Subsequently, ABs of class red will be initialised randomly within the large red square while ABs of class blue will be initialised in the large blue square. Also as shown in the figure, these sub-spaces are calculated by the AGs possessing the largest and smallest values of each feature, indicated by the arrows.

An AB's recognition radius is set as the distance to a random AG of its class. Originally in MAIM, ABs were not guaranteed to interact with AGs after initialisation as their recognition radius were simply set as a random value within the sub-space of its class. However, during experiments (see section 4.5.1) the smarter initialisation was shown to generally be a better choice.

3.2.4 Crossover

As stated in section 2.2.2, crossover is not common in AIS. It is implemented in VALIS but not in AIRS, CLONALG and AISLFS. However, if mutation alone was implemented in the proposed algorithm the effect of migration between the islands would have little impact on the evolving IPs as ABs from different islands would not interact and combine. Crossover enables the migrating individuals to spread their genetic material through the other IPs.

A uniform crossover is employed for its simplicity where two parents create two children as visualised in figure 3.7. The first child, shown as *Child 1* in the example is created from combining random features from both parents. Further, the second child will be the complement of this, shown as *Child 2* in the figure, where the features from the parents not picked for the first child is used to create the second child. Similarly, the recognition radius for the first child is taken randomly from from one of the parents and the second child receives the remaining recognition radius that was not selected. Lastly, in this work only ABs of the same class are selected for crossover with each other through a restrictive form of tournament selection (see 3.2.6), meaning that the classes of the parents are equal and therefore both children receive this class. This approach is further investigated in section 4.2.2.

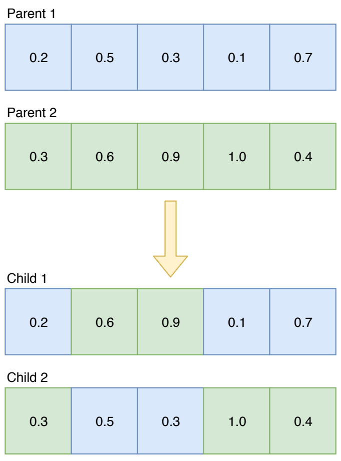


Figure 3.7: Uniform crossover example.

3.2.5 Mutation

A uniform mutation operator is employed for its simplicity, which is controlled by the *mutation rate* parameter as shown in table 3.1. Furthermore, when the mutation operator is applied every feature value in a feature vector of n dimensions has a chance of being mutated with a probability of $1/(1+n)$. When a feature value is selected for mutation it is multiplied by a random real number between 0.1 and 2 based on results of section 4.2.1. Additionally, the recognition radius of the AB will be multiplied by a number within the same interval. This process is illustrated in figure 3.8 where features selected for mutation are highlighted in green and subsequently mutated, incidentally resulting in one feature having its value increased while the other is decreased.

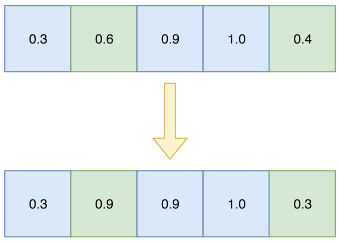


Figure 3.8: Uniform mutation example.

3.2.6 Antibody Selection

In the proposed algorithm both parent and survivor selection for ABs are employed.

Parent Selection

For parent selection, a tournament selection approach has been chosen for its properties and adjustable parameter (see 2.1.1). The selection process is controlled by the *tournament size* parameter of table 3.1. Additionally, crossover is only performed between ABs of the same class, which means that each tournament selection process only creates and selects from tournament sets with ABs of the same label. The class to select an AB from is therefore passed as a parameter to the tournament selection process. This means that in order to select two parents of the same class, the tournament selection process will have to be conducted two times with the same class. Further, in order to efficiently select ABs of a specific class, ABs are split into separate selection pools through a hashmap structure using classes as keys and same-class AB arrays as values.

The first selected parent will be passed as a parameter to the tournament selection process for the second parent. This is done in order to avoid selecting the same parent twice. Finally, according to the *clonal selection principle* only ABs that interact with at least one AG should be selected for reproduction. In the event that the selection process returns a non-interacting AB (i.e. affinity is null, see equation 3.3) it will subsequently be discarded and a completely new AB will be created and initialised to at least interact with one AG through the AB initialisation process explained in section 3.2.3. This approach was selected to avoid potentially running multiple tournaments where the winner has an affinity of null.

Survivor Selection

For survivor selection, a fitness proportionate selection approach has been chosen (see 2.1.1) after reportedly achieving good results in VALIS. The selection process selects *IP* individuals from a combined selection pool of parents and children. Additionally, in order to enhance diversity by removing redundant ABs, an *apoptosis* process similar to the one used in AISLFS (see section 2.2.2) is simultaneously conducted. If, during the selection process, an AB is selected for survival that is the same (i.e. contains exactly the same feature values) as an already selected AB it will subsequently be removed entirely form the population and not included as a survivor. This is implemented through calculating, storing and comparing hash values of the feature vectors.

3.2.7 Fitness Evaluation and Classification

Fitness evaluation occurs in two stages; fitness of the individual ABs b, $(F(b))$, and the fitness of the population P, $(F(P))$.

Antibody Fitness Evaluation

$(F(b))$ is adapted from VALIS for its flexibility and ability to encourage exploration. It constitutes three main components; *sharing factor*, *weighted accuracy* and *AG interactions*. The fitness is calculated as in equation 3.4:

$$F(b) = \frac{SharingFactor(b) \cdot WeightedAccuracy(b)}{AG\_Interactions(b)} \tag{3.4}$$

The main fitness function components include several smaller components which are summarised in table 3.2.

b	The AB getting its fitness calculated.
b'	One specific AB.
B	The set of all ABs.
g	An AG.
G	The set of all AGs.
G_b	The set of all AGs with *true class* being the same class as AB b.
W_{bg}	The affinity between AB b and AG g (see equation 3.3).
k	The number of different classes in the classification problem.

Table 3.2: Fitness formula components.

$F(b)$ is applied during parent selection (see figure 3.2), to select the best antibody of each tournament. Following the crossover and mutation, $F(b)$ is calculated for each new AB of the resulting child population. In addition, $F(b)$ is recalculated for each AB of the parent population to ensure that the sharing factor component (see equation (3.7)) reflects the ABs in the combined parent and child population. The updated $F(b)s$ are then applied in the survivor selection process. To avoid a second $F(b)$ update in a single iteration, $F(b)s$ are not updated to reflect the surviving IP on any island. Subsequently, the already existing $F(b)$ values are applied during parent selection on the next iteration. This decision does not affect the fitness evaluation at the master island.

It should be noted that all ABs and AGs have fixed positions in the feature space. Therefore the components *WeightedAccuracy* and *AG_Interactions* (see equation 3.10 and 3.8 respectively), are unchanged during the lifetime of a given AB. Subsequently, to recalculate $F(b)$ for a given AB, only the sharing factor component is recalculated to reflect the updated sharing of AGs at each iteration, between the current ABs in either the IP, when using local AB_interaction or P, when using local AB_interaction.

Sharing Factor

The *sharing factor* calculation of an AB, b, is as presented in equation 3.7. Furthermore, the sharing factor is the sum of all the AB's *interaction shares*, which is the portion of the an AG's total interaction belonging to the AB. This is presented in equation 3.6, which is calculated and summed for every AG to create the sharing factor. Subsequently, for every AG, its affinity with the AB needs to be calculated and squared before being divided by the affinity sum between all ABs in the IP (or P) and the same AG. Further, this is refereed to as the *AB interactions* (see equation 3.5) of the AG.

$$AB\_interactions(g) = \sum_{b' \epsilon B} W_{b'g} \tag{3.5}$$

$$InteractionShare(b,g) = \frac{W_{bg}^2}{AB\_interactions(g)} \tag{3.6}$$

$$SharingFactor(b) = \sum_{g \epsilon G} InteractionShare(b,g) \tag{3.7}$$

The effect of the sharing factor is that, when it is small the AG is shared with a lot of other ABs which is punished. Some AGs may have many AB connections, while others may have none. Therefore, the sharing factor encourages exploration for such AGs through the punishing effect of the *AB_Interactions* component. As a result, ABs connecting to an AG that is already interacting with a large amount of other ABs is not as favoured as connecting to one interacting with less.

Notice that in the sharing factor formula, the affinity of the AB is squared in the enumerator to give the it a greater impact in the calculation and encourage some sharing. Voting requires that ABs to some extent share AGs, so that more than one AB can vote on a given AG and more accurately determine its class. Subsequently, while exploration should be encouraged, sharing should not be completely discouraged. Further, since each AG is shared, the sharing factor provides the AB with its share of the potential reward [19].

In the default in the model the *AB_Interactions* component is calculated locally on each island and reflects sharing in the sub population, IP, of ABs on the island. However, optionally the master can provide global information about the AB connections to any given AG in P. This parameter, the global *AB_interactions*, is then sent from the master to all slaves to replace the local AB_interactions component in the sharing factor calculations. Subsequently, global AB_interactions represents AG sharing across all islands.

It should be noted that when the sharing factor is not used in the fitness calculation the AB population will converge to areas of high AG density, which severely degrades the accuracy of the algorithm, (see 4.2.4).

AG Interactions

The *AG interactions* of an AB, b, is as presented in equation 3.8. This is the sum of the AB's affinities with all AGs.

$$AG\_Interactions(b) = \sum_{g \epsilon G} W_{bg} \tag{3.8}$$

The other fitness function components are divided by the AG interactions, which means that interacting with a lot of AGs while having low sharing factor and weighted accuracy is subsequently punished.

Weighted Accuracy

The *weighted accuracy* of the AB, b, is presented in equation 3.10. The weighted accuracy is calculated as the affinity sum of all the AGs it correctly classifies (found by comparing the AGs' *true class* to the class of b). Further, this is referred to as the *Correct_AG_Interactions* as presented in equation 3.9 which is divided by b's AG interactions (see equation 3.8). Additionally, Laplacian smoothing is performed on the component by adding 1 in the numerator and k, the number of different classes in the classification problem, in denominator. This is performed in order to prevent overfitting.

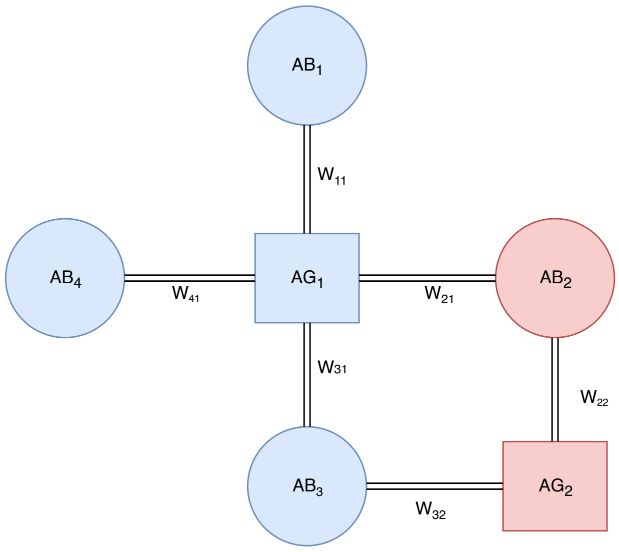
$$Correct\_AG\_Interactions(b) = \sum_{g \epsilon G_b} W_{bg} \tag{3.9}$$

$$WeightedAccuracy(b) = \frac{1 + Correct\_AG\_Interactions(b)}{k + AG\_Interactions(b)} \tag{3.10}$$

The effect of the weighted accuracy term on $F(b)$ is that high-affinity connections between ABs and AGs of the same class are favoured, while high-affinity connections of different classes are subsequently punished. This means that ABs are preferred that reside as close as possible to AGs it correctly classifies while at the same time favouring antibodies with connections to more than one AG of its class, because it reduces the punishing effect of the k term.

It should be noted that the weighted accuracy term is specifically introduced for this work, in order to better suit the proposed algorithm, that allocates a single static class to each AB, as opposed to VALIS that uses class distributions (see 4.5.1) which subsequently changes their accuracy term.

Antibody Fitness Evaluation Example



$$\textbf{Sharing factor of AB}_3 = \frac{(W_{31})^2}{W_{31} + W_{21} + W_{11} + W_{41}} + \frac{(W_{32})^2}{W_{32} + W_{22}}$$

$$\textbf{AG interactions of AB}_3 = W_{32} + W_{31}$$

$$\textbf{Weighted accuracy of AB}_3 = \frac{1 + W_{31}}{2 + W_{31} + W_{32}}$$

Figure 3.9: Visualisation of F(b) component calculations.

Figure 3.9 presents an example of calculating the three main components of the fitness function. In the example, AGs are shown as squares and ABs as circles. Additionally, connections between AGs and ABs are shown as two parallel lines and denoted as a weight, W_{xy}, where x is the AB number and y is the AG number in the connection. Furthermore, different classes are denoted by their colours.

In the figure the fitness components of AB AB_3 is being calculated, firstly by looking at its sharing factor. AB_3 is connected with two different AGs, AG_1 and AG_2. The interaction shares of AB AB_3 for these AGs are therefore calculated based AB_3's affinities with the connected AGs, divided by all the AGs' connections, i.e their AB interactions, and subsequently added together, as shown in the figure.

The next element to be calculated is the AG interactions. In the figure this is calculated by adding all the affinities of AB AB_3, namely W_{31} and W_{32}, together.

Finally, the weighted accuracy for AB_3 is calculated. As shown in the figure, AB_3 is only connected to one AG of its class, AG_1, as AG_2 does not share its colour, meaning that W_{31} is the only correct interaction of the AB. Its weighted accuracy is therefore calculated as $W_{31} + 1$ and divided by $AG\_Interactions(AB_3) + 2$, as 2 is number of different classes in the classification problem.

Voting and Class Assignment

$$V_{g^c} = \sum_{b^c \epsilon B_{g^c}} W_{b^c g^c} \cdot A_{b^c} \tag{3.11}$$

The voting function can be seen in equation 3.11, where b^c and g^c are ABs and AGs of class c, respectively, and A_{b^c} is the weighted accuracy of b^c.

The population of ABs involved in the voting process is the current population P on the master. MAIM employs a voting function for determining what class to allocate to an AG at a particular generation. Every AB, b, of the population competes to allocate its current class to all AGs in its RR. In other words, for each AG, g, all local ABs vote to give it their class. Further, when all local ABs have cast their vote, g is classified as the class with the highest *tally.* For a given class, c, the tally resulting from the vote may be calculated as in equation (3.11). As shown, the vote strength of and an AB, b, of class c for AG g includes the current affinity between b and g in addition to the weighted accuracy, A_b. Subsequently, more accurate ABs will be enabled to have a greater impact on the vote. When all antibodies have cast their votes, the AG is classified as the class with the highest tally, which is the highest V_{g^c}.

An example of calculating vote tallies can be seen in figure 3.10 where the blue circles are ABs of class a and the red circles are ABs of class b. The resulting classification of the AG, AG, will subsequently be determined by which of V^a_{AG} and V^b_{AG} is greater.

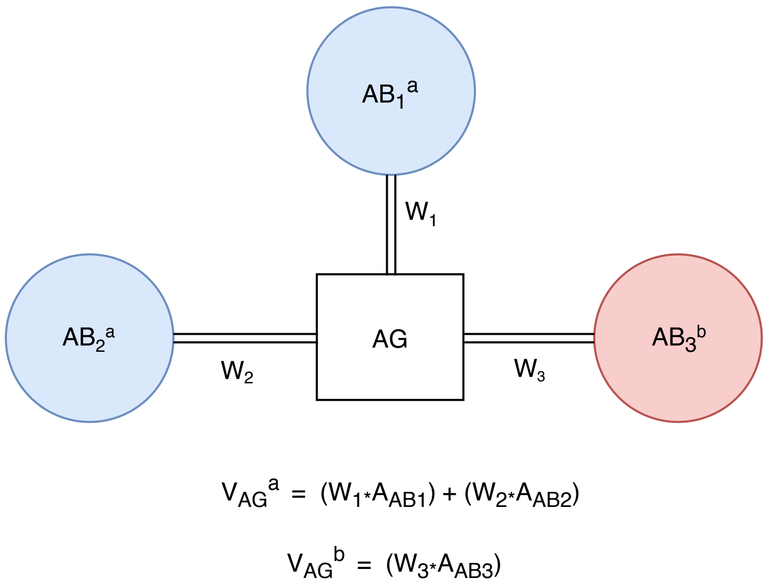

$AB_1{}^a$
W_1
$AB_2{}^a$
W_2
AG
W_3
$AB_3{}^b$
$V_{AG}{}^a = (W_1 * A_{AB1}) + (W_2 * A_{AB2})$
$V_{AG}{}^b = (W_3 * A_{AB3})$

Figure 3.10: Visualisation of voting tally calculations.

One issue with using RRs for voting is that when an AG is not within the RR of any AB it will not be classified. To select a class for a such AGs, kNN is applied. This approach is selected based on tests conducted in section 4.2.3.

Classification Accuracy and Population Fitness

During the voting procedure, classes are allocated to AGs and the allocated class of each AG is compared to its true class included in the data set. Subsequently, population fitness, $F(P)$, corresponds to the classification accuracy of the current population in the master and is expressed as in equation 3.12.

$$F(P) = \frac{t}{T} \tag{3.12}$$

In equation 3.12, t, is the number of AGs correctly classified by the current population of ABs and T is the number of AGs in the data set to be classified.

Chapter 4

Experiments and Results

This chapter will present the algorithm simulator as well as all experiments conducted on MAIM. Firstly the simulator will be explained in section 4.1, followed by preliminary tests conducted during development in section 4.2. Further, in section 4.3 the experimental plan is presented, followed by the experimental setup in section 4.4 and lastly the results are presented in section 4.5.

4.1 Experiment Simulator and Visualisation

A simulator has been created in order for the proposed algorithm to be accurately monitored and tested, which is available at GitHub [1]. This includes a graphical user interface that allows for the user to input the algorithm's parameters and select data set. During the run of the algorithm a graph interface is shown and updated live in order to reflect the current accuracies on slaves and master. Additionally, the AG and AB population at all iterations on all islands can be visualised once the algorithm has finished. Accuracy graphs for a thousand iterations of the algorithm with 3 slave islands, one master island and global AB_interaction (see section 3.2.7) is shown in figure 4.1. Here the respective islands are named with bold text at the top of each graph and the highest achieved accuracy for each island is written below each graph. Iterations where migration takes place between the slave islands are marked in the graphs as red lines while no migration are marked as blue. Classification accuracies are shown from voting using the training set, but optionally the validation set can be shown.

Notice how the slave accuracies fluctuates between iterations as a result of creating new IPs at every run. Additionally, employing the global AB_interaction parameter makes each island vary widely in accuracies as they all search different and smaller parts of the search space, subsequently specialising on a subset of AGs (while being evaluated on all by the simulator). On the other hand, the master only switches populations when a new best P is found, making the graph much more stable. For comparison, a run of the same data set using local AB_interaction is

[1] https://github.com/InfintieEvolution/MAIM

included in figure 4.2. Here it is shown that when only employing local information each island cover larger parts of the complete function, individually achieving higher accuracies.

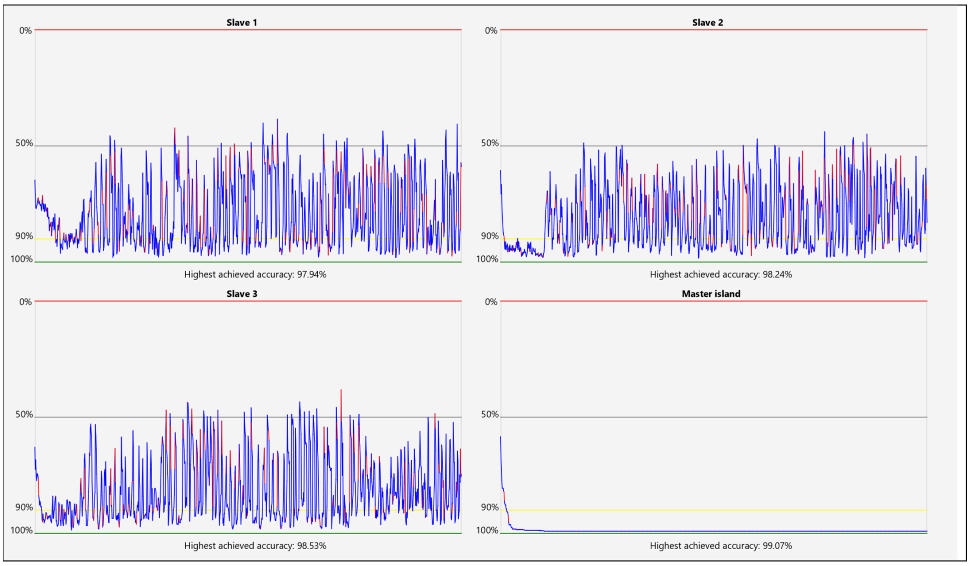


Figure 4.1: MAIM with 3 slaves, 1 master and global AB_interaction.

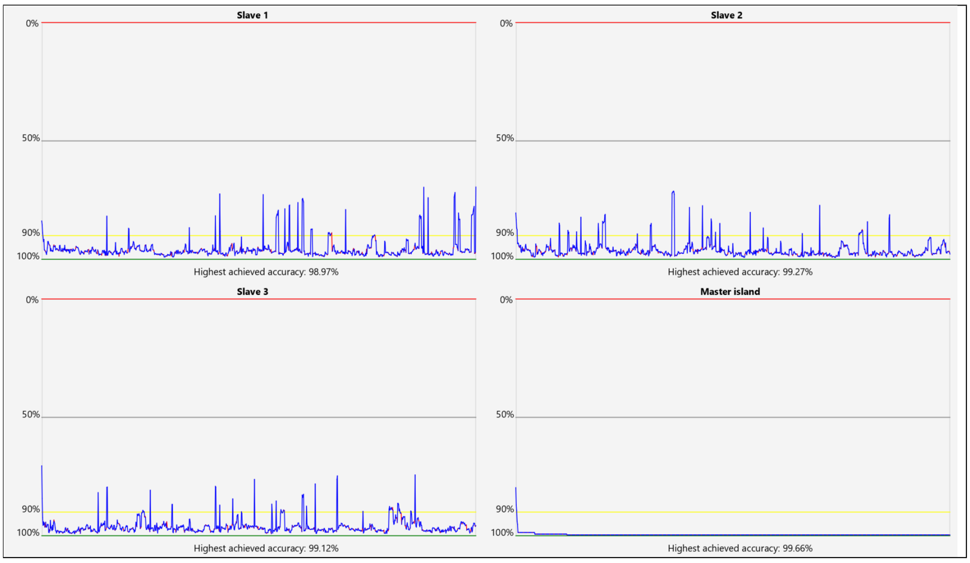
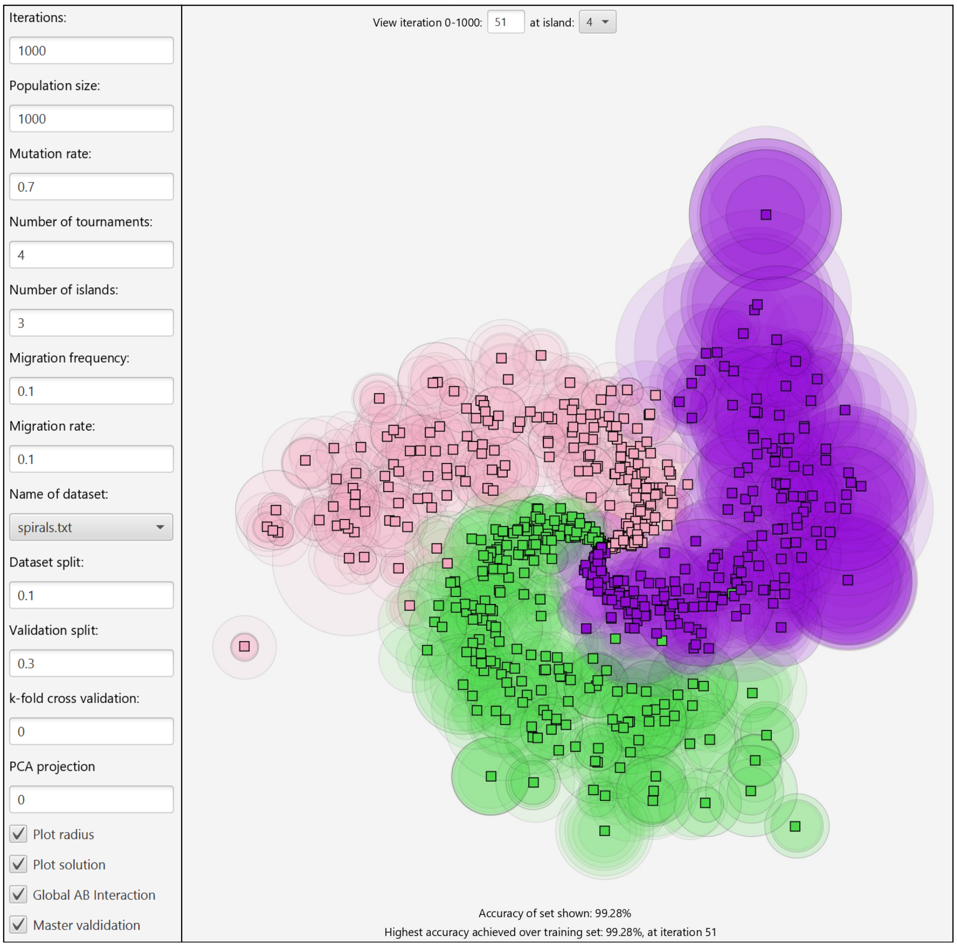


Figure 4.2: MAIM with 3 slaves, 1 master and local AB_interaction.

Figure 4.3 is an example of visualisation of the master-island (corresponding slave island visualisations can be found in appendix .3) population at iteration 51 on an artificially generated two-dimensional spirals data set gathered from the work of VALIS. The image contains 1000 ABs and 680 AGs. AGs are depicted as squares, AB RRs as circles and different classes as different colours. Controls at the top of the image indicates which iteration and which island is currently being viewed. Further, through the controls at the top of the figure, the population of every island can be viewed at every iteration. The master island is here island 4 because the algorithm is ran with 3 slaves. Further, the bottom text indicates the training accuracy achieved at the current iteration as well as what iteration the best accuracy was achieved. In addition, by typing "test" into the "view iteration" field at the top, the population's accuracy over the test set will be shown and visualised. Parameters are input at the left which includes the parameters of table 3.1 as well as some additional parameters for testing and visualisation. Another visualisation, using local AB_interaction can be seen in figure 4.4 where four slaves and their master are displayed.


Iterations: 1000
Population size: 1000
Mutation rate: 0.7
Number of tournaments: 4
Number of islands: 3
Migration frequency: 0.1
Migration rate: 0.1
Name of dataset: spirals.txt
Dataset split: 0.1
Validation split: 0.3
k-fold cross validation: 0
PCA projection: 0
✓ Plot radius
✓ Plot solution
✓ Global AB Interaction
✓ Master valdidation

View iteration 0-1000: 51 at island: 4

Accuracy of set shown: 99.28%
Highest accuracy achieved over training set: 99.28%, at iteration 51

Figure 4.3: Proposed algorithm simulator with user interface and solution example.

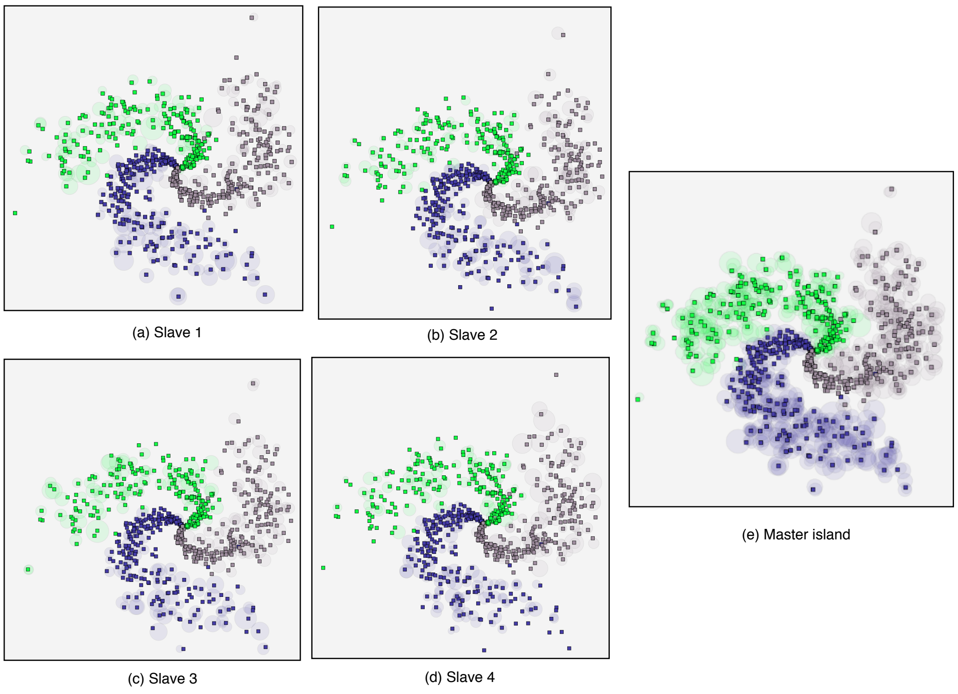


Figure 4.4: Four slave islands and master at iteration 217 using local AB_interaction.

As stated all iterations can be viewed and therefore also the evolution and convergence of the respective slave and master populations. Figure 4.5 shows 4 different iterations of a run of the algorithm on the Wine data set with visualised RRs.

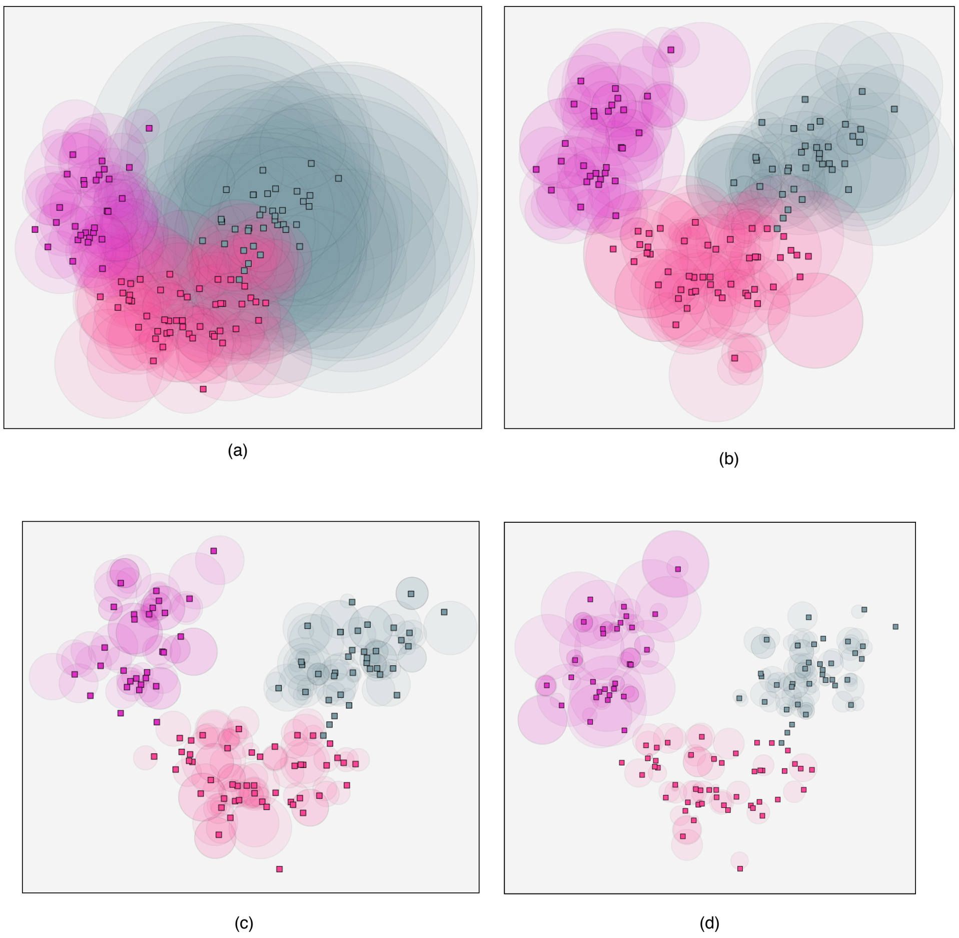


Figure 4.5: Evolution of MAIM on the Wine data set. **(a)** Generation 0 **(b)** Generation 5 **(c)** Generation 50 **(d)** Generation 347

Alternatively ABs can be visualised as small circles with thin lines indicating connections as in figure 4.6. Additionally, four different iterations of the Iris data set is here displayed. Visualisations have been inspired from the work done in VALIS.

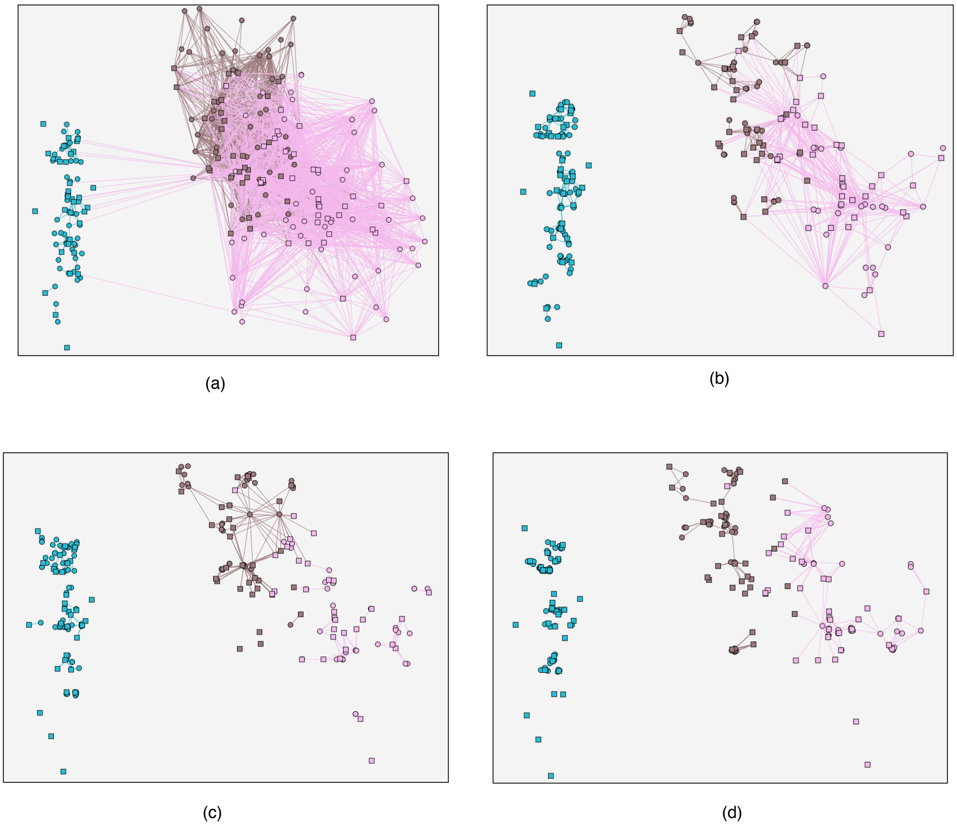


Figure 4.6: Evolution of MAIM on the Iris data set. **(a)** Generation 0 **(b)** Generation 5 **(c)** Generation 45 **(d)** Generation 317

4.2 Preliminary Tests Conducted During Development

This section will constitute experiments conducted while developing the proposed algorithm. These was conducted in order to make sure that the model components would benefit the algorithm. All preliminary tests used the same set of parameters which is defined in table 4.1 and run with ten-fold cross-validation averaging over five runs (see section 2.1.3). Notice that $\# AG$ indicates that the number of AGs in the data set. Furthermore, only the AIS components of the algorithm were tested, likely making the island model unnecessary for these experiments. Therefore, to avoid factoring in the effect of islands, all preliminary tests were conducted running *AIS only*, i.e. without any additional slave or master islands. Subsequently, there are no defined MR and MF for theses tests. The preliminary tests are conducted on 3 different data sets, namely Pima Indians Diabetes (Diabetes), Iris and Wine (see table 4.6), all selected for their different properties. Iris was selected for its small size both in terms of feature space and AG population. Further, Wine was selected for its slightly larger feature space and being easily separable. Finally, diabetes was selected for its larger size and harder difficulty. Finally, results of all accuracy testing in this work is presented as accuracies returned from the MAIM's voting function (see section 3.2.7) with *standard deviations (SDs)* in parentheses.

4.2.1 Mutation Multiplier Range

Parameter	**Value**
Islands	1
Population Size	# AG
Mutation Rate	0.8
Iterations	600
Tournaments	5
Validation Split	0.3

Table 4.1: Parameters used for testing the mutation operator.

Multiple tests were conducted to find the best and most stable value for the range of possible feature multiplier values for the mutation operator, as indicated in section 3.2.5. The results presented in table 4.2 are some examples of results from the tests ran. The different value ranges give quite similar results, with minor individual differences. The individual differences indicates that different data sets might, to a lesser degree, benefit from different ranges in mutation multipliers. It was assumed that a larger multiplier range allows for more exploration and higher, but less stable results because of large range of possible values to select from. However, any range from the ones tested achieved satisfactory results. Subsequently, the tests in this work use the range of $0.1 - 2.0$ as it on average gave the most stable results (0.008 SD).

Value Span of Mutation Operator	**Result**			
	Iris	*Wine*	*Diabetes*	*Average*
0.9 - 1.1	0.945 (0.013)	0.96 (0.015)	0.714 (0.003)	0.872 (0.01)
0.1 - 2.0	**0.947 (0.009)**	**0.948 (0.009)**	**0.72 (0.004)**	**0.871 (0.008)**
0.1 - 3.0	0.947 (0.007)	0.956 (0.016)	0.721 (0.003)	0.875 (0.009)

Table 4.2: Preliminary mutation multiplier testing.

4.2.2 Effects of Crossover Inside and Across Classes

The following test was conducted to identify whether the crossover function should be conducted between ABs of same or different classes. This was proposed in order to achieve faster convergence as same-class parents were assumed to create better same-class children. The results are presented in table 4.3. As shown, no big difference between the two different crossover approaches were found. It should be noted that with the random class crossover the SD is a bit smaller, indicating that the algorithm may be more slightly more stable with this approach. Finally, because of the results showing no great differences, the crossover approach was kept between same classes as in the original model.

Data set	**Result**	
	Equal Class Crossover	Random Class Crossover
Iris	0.953 (0.008)	0.959 (0.007)
Wine	0.948 (0.011)	0.941 (0.006)
Diabetes	0.724 (0.015)	0.725 (0.009)

Table 4.3: Preliminary tests of crossover across classes versus equal classes.

4.2.3 Comparing Voting with the KNN-Rule

Tests following were concocted to investigate the effect using kNN and voting in the propsed algorithm.

Data set	**Results**		
	Voting Only	*KNN(k=5) Only*	*KNN(k=5) and Voting*
Iris	0.929 (0.019)	0.955 (0.011)	0.951 (0.015)
Wine	0.938 (0.014)	0.96 (0.011)	0.958 (0.017)
Diabetes	0.706 (0.019)	0.72 (0.006)	0.725 (0.009)

Table 4.4: Preliminary testing on the difference between the voting function and kNN.

It is clear from Table 4.4 that voting alone results in lower accuracy compared to the other approaches. This is a consequence of AGs that are not in the RRs of any AB and are therefore not allocated a class. As stated in section 3.2.7, the combined approach is a voting approach where kNN is only applied to those AGs that are not assigned a class during voting. As shown, this ensures an accuracy on par with kNN. However, kNN alone is challenged in terms of efficiency.

To investigate run times with the different approaches, the Diabetes data set was applied as it is the largest of the data sets used and therefore is very dependant on an efficient classification strategy. The results clearly showed kNN alone to be inefficient with **6.57** minutes run time, compared to that of a combined voting and kNN of **2.26** minutes and voting alone of **2.15** minutes.

KNN may be said to be a *lazy learning* approach, requiring no training at the cost of being computationally heavy during classification. On the other hand, voting is an *eager learning* approach that is computationally heavy during training, but provides fast classification. AIS, is also an eager learning approach so when combined with kNN, efficiency is negatively impacted. However, when combining MAIM's voting with the kNN, efficiency is similar to that of voting alone and while simultaneously enabling the increased accuracies shown. Subsequently, a combined kNN and voting approach was chosen for its compromise in performance. Additionally, as it is not guaranteed than every AG will always be within an RR it is likely a good idea to use kNN as a backup regardless.

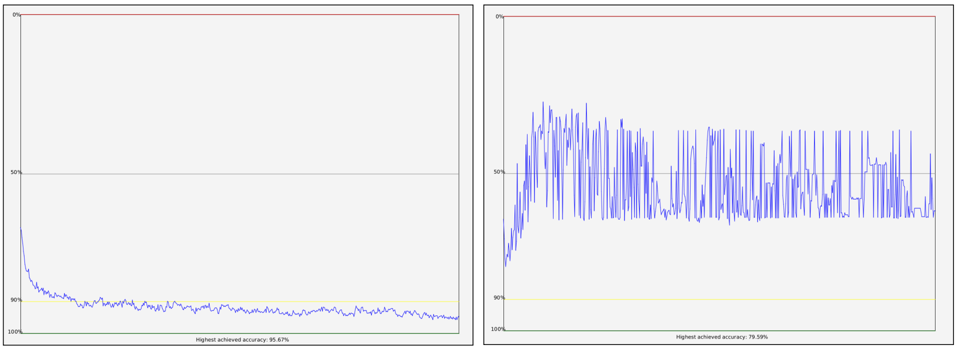
It should be noted that the poor performance of voting only is likely due to insufficient exploration and some AGs not residing within an AB RR. Further work has been done that improves exploration in section 4.5.1.

4.2.4 Effects of Sharing Factor

Data set	**Result**	
	With Sharing Factor	*Without Sharing Factor*
Iris	**0.953 (0.007)**	0.943 (0.019)
Wine	**0.948 (0.011)**	0.938 (0.011)
Diabetes	**0.724 (0.012)**	0.723 (0.009)

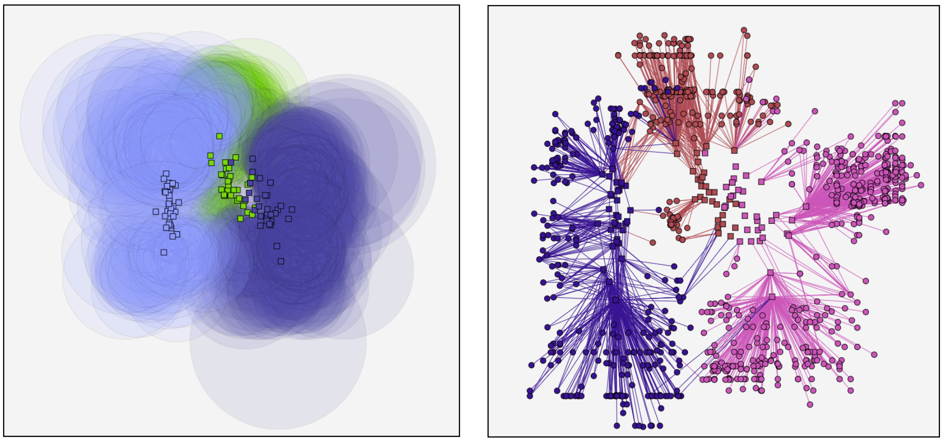
Table 4.5: Preliminary testing of sharing factor.

The effects of the sharing factor (see section 3.2.7) has been found by testing the proposed algorithm, with and without sharing factor. The results are presented in table 4.5. As the results indicate, the algorithm obtains better results with sharing factor than without. Furthermore, The sharing factor has a substantial impact on how the algorithm evolve over time. Figure 4.7 presents two graphs of how the algorithm is evolving with and without sharing factor, showing the algorithm converging to better accuracies with sharing factors while diverging without. Furthermore, figure 4.8 shows a visualisation of how the AB population looks after evolving without sharing factor. By comparing this figure to 4.10, which is the same data set, one can see that the sharing factor has a big impact in how ABs position themselves. From comparing the figures one can see that without sharing factor many more connections are calculated as AG sharing is not punished, allowing AGs to position themselves in areas of high AG density while having large RRs. Consequently, this impacts the efficiency of the algorithm. On one hand, when calculating sharing factor, but not using it in the fitness function, $F(b)$, the average time training time for the Diabetes data set was **07:15 minutes**. On the other hand, when both calculating and using the sharing factor in the fitness function, the average time dropped to **02:26 minutes**. Subsequently, when too many ABs connect to too many of the same AGs it is referred to as *AB congestion*, resulting in exponential amount of connections and calculations. This concept is further investigated and explained in section 4.5.2.



(a) With sharing factor enabled (b) Without sharing factor enabled

Figure 4.7: Visualisation of sharing factor effects on the Diabetes data set.



(a) Iris dataset with antibody radius drawn (b) Iris dataset with antibody to antigen connection

Figure 4.8: 2D solution plots of the Iris data set, without sharing factor.

4.3 Experimental Plan

The experiments present an analysis of some of the key decisions behind the model and their effect on accuracy and/or efficiency. Further, The research questions for this thesis are separated into three areas of enhancing accuracy and efficiency as well as the impact of migration. Based on these questions, the experiments will be conducted in three respective phases.

The first phase, **T1**, will be focused on testing classification accuracy. In order to accurately test the performance of the algorithm, different tests will be done one several datasets with varying properties. Further, the results will be compared to state of the art classification algorithms.

The second phase, **T2**, of testing will be focused towards efficiency. Efficiency in this context corresponds to the training times or run times of the MAIM algorithm. However, the state of the art classifiers introduced do not provide any measures of their efficiency. Subsequently, the efficiency tests will be conducted by running the proposed algorithm with different island numbers and population sizes and comparing the different results with each other. For instance, by comparing the AIS model by itself against MAIM with different island numbers, one can examine the effects of the island model on the AIS.

The third phase, **T3**, will investigate the impact of migration, through MF, MR and policy in terms of accuracies achieved.

Data set	**Samples**	**Attributes**	**Classes**	**Source**
Iris	150	4	3	UCI
Wine	178	13	3	UCI
Pima Indians Diabetes	768	8	2	UCI
Sonar	208	60	2	UCI
Glass	214	9	6	UCI
Heart Statlog	270	13	2	UCI
Ionosphere	351	34	2	UCI
Breast Cancer Wisconsin	699	9	2	UCI

Table 4.6: Data sets used for testing.

The data sets that will be used for testing are presented in table 4.6, and is taken from the UCI Machine Learning Repository [7]. Further, VALIS, ARIS and AISLFS are all tested against some of these data sets. This allows for a comparison of the proposed algorithm with several state of the art AIS classifiers. The properties of the data sets are also varies, spanning from 2-6 classes, 4-60 dimensions and 150-768 samples. The chosen data sets also have a big variation on sizes and difficulties and each one were selected for their unique properties in order to challenge the algorithm. Iris and Wine are both fairly easily separable datasets. However, Iris was selected for having few AGs and a small feature space, and Wine, which is also small, for having a larger feature space than Iris. Further, Breast Cancer Wisconsin (Breast Cancer) is a data set with a large sample size that was selected for being somewhere in between easy and difficult to separate. This is because it consists of a combination of easily classifiable samples while also containing a few that are very difficult. Further, Diabetes and Heart Statlog were selected for being quite difficult data sets with fairly small feature spaces and few classes. Additionally, Diabetes was also selected for having a large sample size. Further, Sonar and Ionosphere were selected for being somewhat difficult while having large feature spaces. However, Sonar has a substantially larger feature space and is generally harder than Ionosphere. Finally, Glass was selected for being a difficult data set with many classes and few cases per class.

4.3.1 Overview of Test Plan

Table 4.7 presents an overview of the test plan. All tests marked with **TL** means that they are conducted with *local AB_interaction*, while the tests marked with **TG** are conducted with *global AB_interaction*. Tests are initially conducted with *local AB_interaction*, but some are redone using *global AB_interaction* in order to evaluate its effects.

Test Overview		Hypotheses
Phase One - **T1**	Testing of classification accuracy with comparisons with classifiers. • **T(L/G) 1.1** - Tests on fairly separable data. • **T(L/G) 1.2** - Tests on more difficult non-separable data. • **T(L/G) 1.3** - Tests on difficult non-separable data with many classes. • **T(L/G) 1.4** - Tests on data with large feature spaces. • **T(L) 1.5** - Tests with new radius initialisation and higher populations and iterations. • **T(L) 1.6** - Test with class distributions.	The proposed algorithm will provide results which are mostly on par with the other classifiers for tests **1.1**, **1.2** and **1.3**. Further, global information, **G**, is expected to perform better than local information, **L**, as it becomes a more informed search. However, since the proposed algorithm does not contain any sort of feature selection, results from tests, **T1.4**, are expected to be worse. Furthermore, results are expected to improve when the MAIM is given better exploration capabilities through initialisation, iterations and population sizes in test **T1.5**. Finally, in test **TL1.6** class distributions are expected to overall perform worse than static classes, as a result of less specialised ABs.
Phase Two - **T2**	Efficiency testing in terms of run times and compared against itself. • **T(L/G) 2.1** - Tested with *P* as *500*, using AIS only, four, eight and twelve islands. • **T(L) 2.2** - Tested with *P* as *1000*, using AIS only, four, eight and twelve islands and compared to a single AIS. • **T(L) 2.3** - Tested with *P* as *1500*, using AIS only, four, eight and twelve islands. • **T(L/G) 2.4** - Tested with *P* as *2000*, using AIS only, four, eight and twelve islands. • **T(L) 2.5** - Tested with *P* as *4000* and *6000* on the diabetes dataset, using AIS only, four, eight and twelve islands. • **T(L) 2.6** - Tested with *P* as *#AGs*, using three and four islands.	The proposed algorithm is expected to perform somewhat better with an island structure than without on **T2.1** and **T2.2**, but not by a large margin. However, the more islands the algorithm are initialised with, the better the efficiency on large populations because of the increased distribution, which will be seen in tests **T2.3**, **T2.4** and **T2.5**. However, with very small population sizes in **T2.6**, the efficiency enhancements will be either be relatively small or, for many islands, cause extra overhead that will make MAIM less efficient. Finally, local information, **L**, is assumed to be slightly more efficient than global information, **G**, because master computation is reduced.
Phase Three - **T3**	Migration testing in terms of MR, MF and policy and their effect on accuracy. • **T(L/G) 3.1** - Parameter sweep on the Wine data set with non-deletion migration. • **T(L/G) 3.2** - Parameter sweep on the Diabets data set with non-deletion migration. • **T(L/G) 3.3** - Parameter sweep on the Heart Statlog data set with non-deletion migration. • **T(L) 3.4** - Parameter sweep with deletion migration on Diabetes, Wine and Heart data sets.	The proposed algorithm is expected to achieve better results with migration than without. Further, it will perform the best with small values for both MF and MR in tests **3.1**, **3.2** and **3.3**. This is because as more and larger migrations occur the likelihood of MAIM never converging increases. Further, local information, **L**, is expected to benefit more from the diversity provided by migration than global information, **G**. This is because global information already makes the islands significantly different from each other. Finally, migration without deletion is expected to perform better than with, in test **T3.4**, as AB deletion may destroy valuable parts of the sub-solutions.

Table 4.7: Test overview and hypotheses.

4.4 Experimental Setup

The following section will describe the experimental setup for the different tests for the proposed algorithm. Generally, in all tests, the AG population is divided into complete training and test sets, proportional to the cross-validation fold, which is always generated with random splits. Further, SDs are presented in parenthesis next to accuracy and efficiency results. Finally, the number behind MAIM is indicating the number of islands initialised for the given test, i.e. meaning MAIM-3 is the test with three slave islands.

4.4.1 Phase One - Accuracy Testing

Accuracy tests will be conducted in order to evaluate MAIM's generalisation ability. In order to accurately measure its performance, test results will be compared against results from AIRS [24], VALIS [19] and AISLFS [8], as well as *MAIM-AIS* which is the MAIM AIS model running by itself. For AIRS, two common configurations is used for comparison, AIRS-1 and AIRS-7, where the numbers indicate how many of the nearest neighbours are used when classifying an AG. Further, the data sets tested are all the sets of table 4.6, which is selected for their different properties (see section 4.3). As described in table 4.7, the accuracy tests will be divided into the six cases from **T1.1-T1.6** with hypotheses as presented in table 4.7. Additionally, in the tests the data sets will be visualised with solutions containing both AGs and ABs after 600 iterations of training by projecting them in two dimensions using *principle component analysis (PCA)*. Further, this is meant to serve as an indication of the degree of separability and layout of the different data sets, which is visualised through the two different visualisations available in the simulator (see section 4.1). Some data sets may be better visualised with one method over the other, which is why they were both included.

The population size for each test will be equal to the number of initialised AGs (see table 4.6). A full list of parameters used can be found in table 4.8. None of the parameters for the test setup are fine tuned for obtaining high accuracies. Further, all accuracy results presented are the average of five ten-fold cross-validation runs. Finally, results highlighted in bold are the best of each column.

Parameter	**Value**
Iterations	600
Total Population Size	#AGs
#Islands(range)	1-12
Mutation Rate	0.7
#Tournaments	4.0
Migration Frequency	0.2
Migration Rate	0.1
Validation Split	0.3

Table 4.8: Parameter setup for accuracy tests.

4.4.2 Phase Two - Efficiency Testing

The efficiency tests will focus on how different number of islands will effect the efficiency of the algorithm, as populations increase. Therefore the tests will be separated into six parts, **T2.1-T2.6**. Each part will be tested with AIS only (MAIM-AIS), four, eight and twelve slave islands. Further, the populations tested are 500, 1000, 1500, 2000, 4000, 6000 and $\#AG$ with hypotheses as presented in 4.7. A full list of parameters used can be found in table 4.9. The tests will be conducted on four data sets; Wine, Iris, Diabetes and Sonar. The sets are chosen for their variability in sizes, both in terms of samples and feature space, as well as varying level of complexity (see section 4.3) in order to see the impact of the different set properties on efficiency. Further, the efficiency tests are conducted with an Intel Core i7-4790 CPU (3.60GHz, 4 cores) processor. Additionally, the run time results are displayed as the average in seconds of five ten-fold cross-validation runs, i.e. each result is the cumulative time of ten runs of the algorithm. Finally, results highlighted in bold are the best of each row.

Parameter	**Value**
Iterations	600
Population Size(range)	500-6000
#Islands(range)	1-12
Mutation Rate	0.7
#Tournaments	4
Migration Frequency	0.2
Migration Rate	0.1
Validation Split	0.3

Table 4.9: Parameter setup for the efficiency tests.

4.4.3 Phase Three - Effects of Migration Policy, Rate and Frequency

The tests for migration effects will be conducted by accuracy parameter sweeping, of MF and MR. Further, sweeping will be conducted for both deletion an no-deletion migration (see section 3.1.3). Further, the sweeping range will span from 0 to 1, and increase in 0.1 increments. The tests will be ran on three different data sets for tests **T3.1-T3.3** in order to see the effect of migration on data sets of varying properties. The data sets selected are Iris for its small size and separability, Heart Statlog for its complexity and Diabetes for both its complexity and large size. Additionally, a test, **T3.4**, is employed to test the effect of migration with deletion. Further, a full parameter list can be found in table 4.10. Finally, accuracies are indicated in by their colour intensity in the heatmaps, corresponding to the average of five ten-fold cross-validation runs. A random split of the data sets is generated for every ten-fold cross-validation run conducted.

Parameters for Master Slave	
Parameter	**Value**
Generations	600
Population Size	#AGs
Mutation Rate	0.7
#Tournaments	4
#Islands	4
Migration Frequency(range)	0.0-1.0
Migration Rate(range)	0.0-1.0
Validation Split	0.3

Table 4.10: Parameter setup for migration tests.

4.5 Experimental Results and Evaluation

The following sections will present and evaluate results of MAIM testing.

4.5.1 Results Phase One - Accuracy Testing

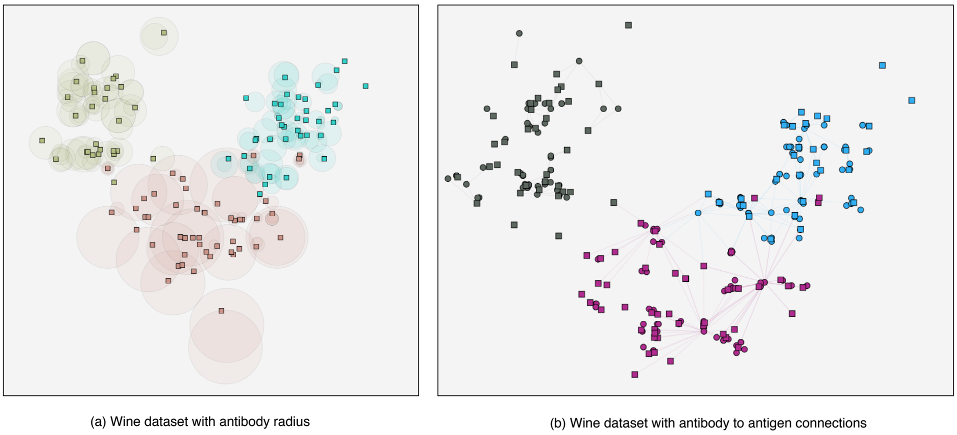
The following section concerns the accuracy results of tests **T1.1-1.5** (see table 4.7) with setups as presented in section 4.4.1. Further, test using only local AB_interaction, **TL1.1-1.5**, will first be conducted followed by tests using global AB_interaction, **TG1.1-1.5**.

Accuracy With local AB_interaction

Algorithm	Wine	Iris
VALIS	0.972 (0.005)	0.956 (0.005)
AIRS-1	*na*	0.96 (0.0560)
AIRS-7	*na*	0.953 (0.055)
AISLFS	**0.9776 (0.0058)**	0.957 (0.0038)
MAIM-AIS	0.958 (0.017)	0.951 (0.0153)
MAIM-3	0.966 (0.006)	0.964 (0.006)
MAIM-4	0.967 (0.003)	**0.965 (0.006)**
MAIM-5	0.963 (0.008)	0.948 (0.011)
MAIM-6	0.969 (0.003)	0.937 (0.008)
MAIM-7	0.967 (0.007)	0.945 (0.014)
MAIM-8	0.972 (0.007)	0.943 (0.012)

Table 4.11: TL1.1 Accuracy test with local AB_interaction, on partially separable data.

The results for test **TL1.1** are presented in table 4.11 which contains data sets where classes are fairly easily separable. First off, for the Wine data set, seen in figure 4.9, the proposed algorithm performs on par with VALIS and AISLFS, performing just as well as VALIS on eight islands, and almost as good as AISLFS. These results are as expected, as MAIM is assumed to perform well on separable data, as IGA is shown to be good at easily separable problems (see section 2.2.1) and from the facts that UCI defines Wine as a ”well behaved” data set, which is good for testing new classifiers [7]. Furthermore, the most interesting thing about this particular data set is that MAIM provided the best results with eight islands. This is the only accuracy test with local AB_interaction where 8 islands have proven to be the best. Additionally, 4 and 6 islands also perform well and accuracies seem to somewhat increase with more islands for this particular dataset. This may indicate that the genetic isolation and convergence provided by employing islands is positive.


(a) Wine dataset with antibody radius

(b) Wine dataset with antibody to antigen connections

Figure 4.9: 2D solution plots of the Wine data set.

To see whether the Wine data set specifically prefers 8 islands or simply a large island number, a test, **TL1.1.1**, for 10 and 12 islands were conducted with results in table 4.12.

Number of Islands	**Accuracy**
10	0.966 (0.009)
12	0.96 (0.009)

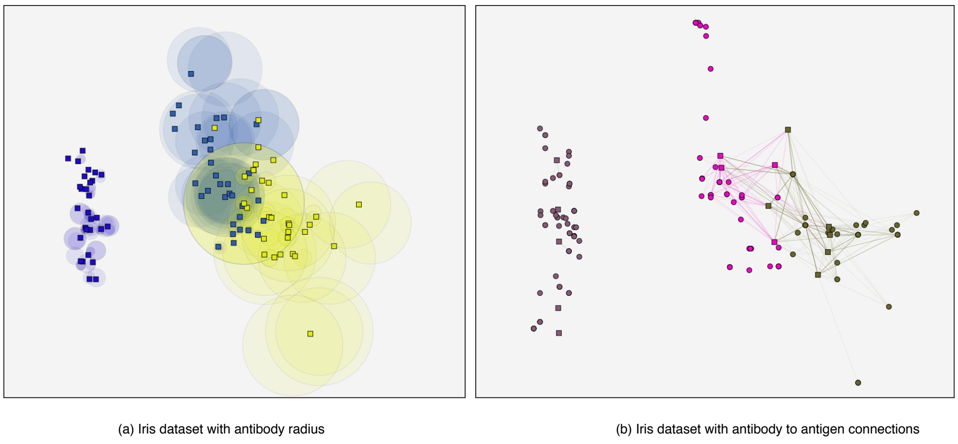
Table 4.12: TL1.1.1 Accuracy tests for the Wine data set employing 10 and 12 islands.

The results does not indicate that increasing the number of islands will further enhance the results. Rather, it seems like the accuracies slightly decrease while SDs slightly increase when population distribution becomes greater. This might indicate that when IPs become too small it makes the results more unstable. To further investigate this another test, **TL1.1.2**, was conducted, with ten islands while doubling the population size from 178 to 356 individuals. The results are as presented in table 4.13 and show that increasing the population slightly increases accuracy while decreasing SD, stabilising the population and subsequently the results. However, the differences are almost negligible and further testing should be done to draw any conclusions. Subsequently, further tests are done on larger population sizes in tests **TL1.3** and **TL1.5**. Furthermore, based on the fact that accuracies do not continue to increase it seems that the Wine data set prefer exactly 8 islands over other numbers, indicating that it provides a good level of genetic isolation for this data set (see section 2.2.1). This could also indicate that these properties are more easily exploitable by easily separable data.

Number of Islands	Total AB Population	Accuracy
10	356	0.969 (0.005)

Table 4.13: TL1.1.2 Accuracy test for the Wine data set employing 10 islands and 356 ABs.

The Iris data set is much like the Wine data set with easily seaparable data. As seen in figure 4.10, one class is linearly separable from the other two which are more overlapping. Also here MAIM presents promising results, beating the other AIS classifiers by a small margin, with both 3 and 4 islands. It is worth noting that for this data set, as opposed to Wine, increasing the number of islands seem to worsen the results. This is possibly due to Iris being the smallest set being tested, with only 150 samples, making it vulnerable to extensive distribution when only 150 ABs are employed. Similarly to test **TL1.1.1** this might indicate that IPs should not get too small in order for an island structure to be effective.


(a) Iris dataset with antibody radius

(b) Iris dataset with antibody to antigen connections

Figure 4.10: 2D solution plots of the Iris data set.

To test this assumption, another test, **TL1.1.3**, was conducted with 8 islands and double P. Table 4.14 presents the results of this experiment where accuracies are shown to increase with larger IPs at the same time as SDs decrease, further indicating that when IPs become too small it may in fact worsen the performance. This is consistent with the results of test **TL1.1.2**. Subsequently, the worsening accuracies can be counteracted by employing larger IPs while concurrently stabilising the results. However, at what point the distribution of the total population becomes too great seems to be vary depending on the data set, as the Wine and other subsequent tests will show that some data sets benefit from more distribution than others.

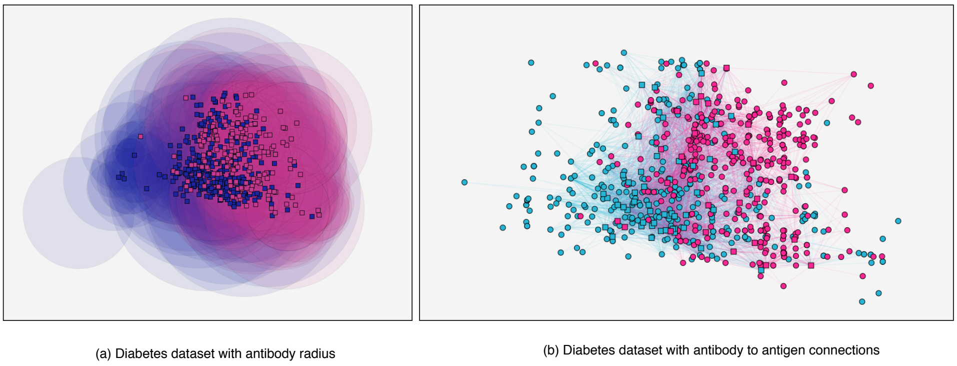
Parameters	**Values**
Islands	8
Population Size	300
Result	**0.965 (0.003)**

Table 4.14: TL1.1.3 Accuracy test for the Iris data set with 8 islands and 300 ABs

Algorithm	**Pima Indians Diabetes**	**Heart Statlog**	**Breast Cancer Wisconsin**
VALIS	*na*	*na*	*na*
AIRS-1	0.674 (0.046)	*na*	0.961 (0.018)
AIRS-7	0.736 (0.035)	*na*	0.962 (0.019)
AISLFS	0.742 (0.009)	**0.8177 (0.0103)**	0.9642 (0.0026)
MAIM-AIS	0.725 (0.009)	0.798 (0.003)	0.962 (0.0013)
MAIM-3	0.751 (0.006)	0.805 (0.006)	**0.969 (0.001)**
MAIM-4	**0.757 (0.006)**	0.813 (0.008)	**0.969 (0.001)**
MAIM-5	0.747 (0.006)	0.807 (0.011)	0.968 (0.002)
MAIM-6	0.742 (0.004)	0.799 (0.001)	0.969 (0.002)
MAIM-7	0.748 (0.004)	0.806 (0.01)	0.969 (0.002)
MAIM-8	0.750 (0.019)	0.802 (0.01)	0.967 (0.003)

Table 4.15: TL1.2 Accuracy tests with local AB_interaction, on non-separable data.

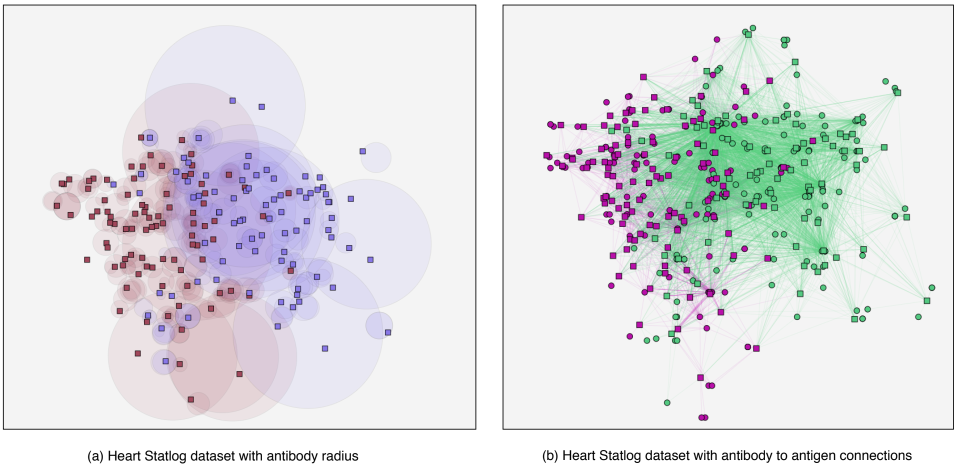
The table 4.15 presents results for test **TL1.2**, on difficult to separate data of varying difficulty. Figure 4.11 is a 2D plot of the Diabetes data set where one can see that the two classes overlap significantly. However, the results for this data set is very promising. MAIM performs the same or better than both AIRS and AISLFS on every island number tested. Further, the best result, MAIM-4, is better than both AIRS and AISLFS by 2.1% and 1.5%, respectively.


(a) Diabetes dataset with antibody radius

(b) Diabetes dataset with antibody to antigen connections

Figure 4.11: 2D solution plots of the Diabetes data set.

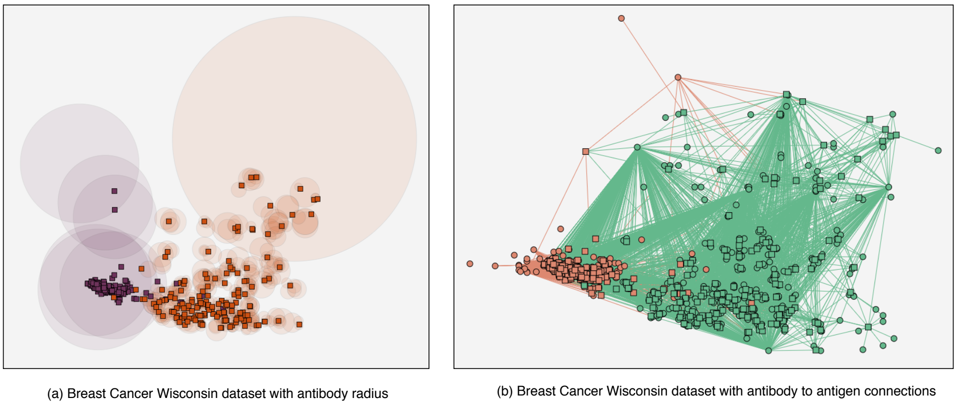
The Heart Statlog data set look a lot like the diabetes data set when plotted in two dimensions (see figure 4.12) with significantly overlapping classes. Further, as seen of the results in table 4.15, MAIM performs well on this data when compared to AISLFS, especially when looking the best-performing island number, MAIM-4. Here, the results of the two algorithms are quite similar, with AISLFS having a slightly higher accuracy, while MAIM-4 has a slightly lower SD.


(a) Heart Statlog dataset with antibody radius

(b) Heart Statlog dataset with antibody to antigen connections

Figure 4.12: 2D solution plots of the Heart Statlog data set.

The Breast Cancer Wisconsin data set is one of the easier data sets where most classification algorithms achieve around 96%. It consists of two classes that are mostly easily separable as shown in figure 4.13. However, it also contain a few cases of the two classes that are almost indistinguishable, making anything higher than 96% hard to achieve. MAIM performs well on this data set, better than AIRS and AISLFS on all island numbers, although only slightly, as seen in table 4.15. Furthermore, 3 and 4 islands is shown perform the best also here.



(a) Breast Cancer Wisconsin dataset with antibody radius (b) Breast Cancer Wisconsin dataset with antibody to antigen connections

Figure 4.13: 2D solution plots of the Breast Cancer Wisconsin data set.

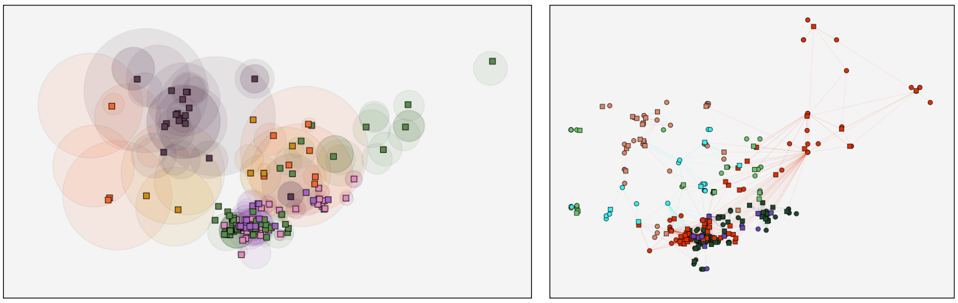
From the results of table 4.15 one can conclude that MAIM perform well on these types of data sets which have larger sample sizes, are harder to separate, contain only two classes and does not have particularly large feature spaces. Furthermore, the results of MAIM is significantly better than the results of MAIM's AIS by itself. This indicates that the properties of the island model is positive for these data sets. Through the enhanced exploration and convergence provided by a migrating island structure, better results can be found. Further, diversity is likely important for these data sets, as they all contain regions with AGs that are particularly hard to correctly classify. Therefore, the island structure employed provide the islands with enough new genetic material to over time create really well-performing ABs for the particularly difficult regions. Additionally, 4 islands performs the best on all data sets, indicating that it provides a good compromise between genetic isolation and interconnection (see section 2.2.1).

Algorithm	**Glass**
VALIS	0.689 (0.024)
AIRS-1	*na*
AIRS-7	*na*
AISLFS	**0.754 (0.0161)**
MAIM-AIS	0.619 (0.021)
MAIM-3	0.640 (0.025)
MAIM-4	0.640 (0.021)
MAIM-5	0.619 (0.019)
MAIM-6	0.646 (0.011)
MAIM-7	0.642 (0.018)
MAIM-8	0.641 (0.019)

Table 4.16: TL1.3 Accuracy tests with local AB_interaction, on non-separable data with many classes.

The test **TL1.3** investigates performance on the Glass data set, containing many classes with few AGs of each class. Glass is a very difficult to solve for the proposed algorithm, with 6 classes and 9 features and where one class only constitutes 9 of the 214 total samples. Additionally, as seen in figure 4.14, the classes are quite overlapping and inseparable. The results of the test can be seen in table 4.16 where MAIM is shown to perform a bit worse than VALIS and substantially worse than AISLFS. This could indicate that MAIM struggles with generalisation when one or more classes contain few samples and particularly when there are also few ABs per AG.

A new experiment, test **TL1.3.1**, have been conducted with larger population sizes. Here, the number of islands is set to 6 as this it what preforms best in test **TL1.3**, population sizes are increased as shown in table 4.17, while all other parameters remain the same. The new results indicates that the assumption could be correct. Every new test has increased the population size by 214 (#AGs) individuals it is clear that the algorithm performs better as the population grows. However, after reaching 1070 individuals, it seems that there is not much to gain by continuing to increase the population. Further, as the population increase, the SDs decrease, which is consistent with tests **TL1.1.2** and **TL1.1.3**, indicating that it is stabilising for larger populations. Further, while MAIM's performance was increased it was not able to perform better than VALIS. This is likely a consequence of VALIS using class distributions instead of static classes, which is further investigated in test **TL1.6**. AISLFS, on the other hand, achieve much better results than both VALIS and MAIM, indicating that the results on Glass significantly benefit from feature selection. However, feature selection is outside the scope of this work, but is proposed as future work in section 5.3.1. Finally, tuning of population sizes shows promising results, which is further investigated in section test **TL1.5**.


(a) Glass dataset with antibody radius

(b) Glass dataset with antibody to antigen connections

Figure 4.14: 2D solution plots of the Glass data set.

Population Size	**Results**
428	0.659 (0.019)
642	0.665 (0.013)
856	0.675 (0.016)
1070	**0.677 (0.013)**
1284	0.674 (0.01)

Table 4.17: TL1.3.1 Accuracies for the Glass data set with 6 islands and increasing population sizes.

Algorithm	**Sonar**	**Ionosphere**
VALIS	0.818 (0.020)	0.928 (0.007)
AIRS-1	0.841 (0.074)	0.869 (0.031)
AIRS-7	0.765 (0.084)	0.886 (0.050)
AISLFS	**0.881 (0.0118)**	**0.9437 (0.0049)**
MAIM-AIS	0.619 (0.099)	0.641 (0.001)
MAIM-3	0.657 (0.067)	0.673 (0.006)
MAIM-4	0.675 (0.085)	0.671 (0.021)
MAIM-5	0.659 (0.085)	0.680 (0.01)
MAIM-6	0.632 (0.008)	0.662 (0.012)
MAIM-7	0.629 (0.017)	0.675 (0.007)
MAIM-8	0.608 (0.031)	0.657 (0.007)

Table 4.18: TL1.4 Accuracy tests with local AB_interaction, on data with big feature spaces.

The test **TL1.4** evaluates the last two data sets; Sonar and Ionosphere. Further, both data sets have quite large feature spaces with Sonar having 60 and Ionosphere 34 features. Both sets have 2 classes. Additionally, as seen in the PCA projection in figure 4.15 and 4.16, the areas of the two classes in both data sets overlap quite a bit.

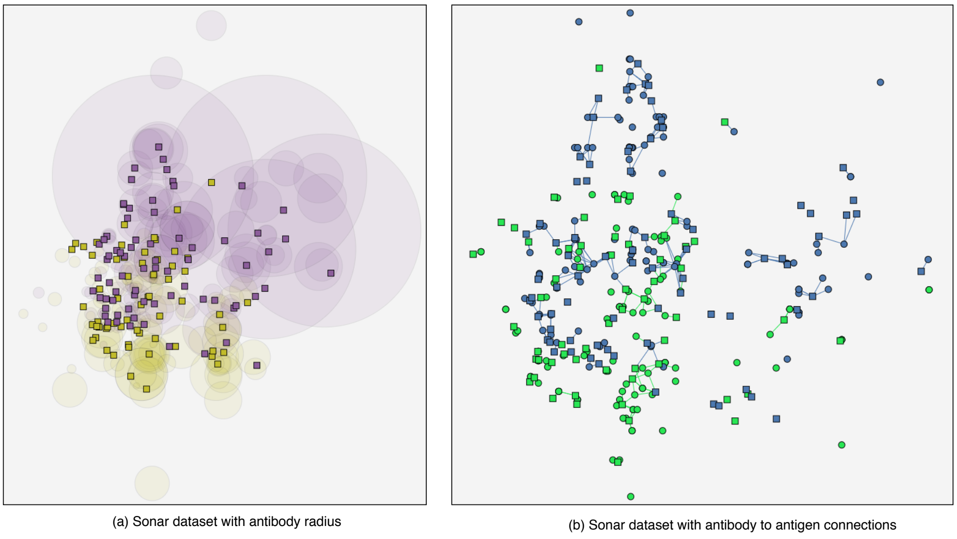

(a) Sonar dataset with antibody radius (b) Sonar dataset with antibody to antigen connections

Figure 4.15: 2D solution plots of the Sonar data set.

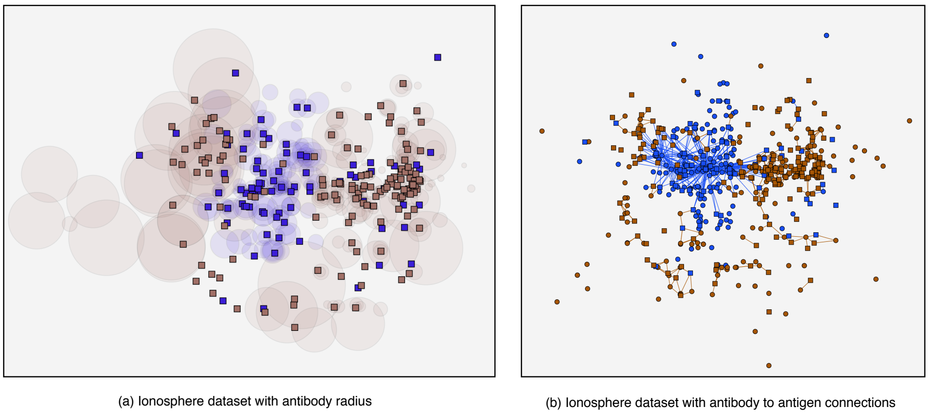

(a) Ionosphere dataset with antibody radius (b) Ionosphere dataset with antibody to antigen connections

Figure 4.16: 2D solution plots of the Ionosphere data set.

The results are presented in table 4.18. As seen, the results of MAIM are very poor compared to the other classifiers. One reason for the poor performance may be due to the lack of any feature selection for the proposed algorithm. However, AIRS and VALIS do not report any feature selection used, indicating that this is not the only cause of the poor performance as they both still achieve good results on these data sets.

There are a few distinct differences between MAIM and algorithms like AISLFS and AIRS. The AISLFS and AIRS algorithms evolve ABs that are targeted at specific AGs. This means that every AG is eventually specifically selected and AB candidates are cloned and mutated to best connect to these AG. On the other hand, MAIM evolve ABs in a way similar to traditional GAs (see section 2.1.1) where a new population is generated and selected at each iteration. Subsequently, ABs in MAIM are not generated and specialised to recognise specific AGs and is only selected based on their fitness. However, because of this approach, as opposed to AIRS and AISLFS, this means that there is no guarantee that every AG receives a connection from an AB. Subsequently, in large feature spaces there is a good chance that some AGs are left out of the final function approximation, as a result of insufficient exploration of the search space. This is further substantiated by the very large SDs on the Sonar datasets indicating very unstable results due to luck of whether all AGs are found or not.

It is interesting that VALIS, which is the algorithm that MAIM is the most similar to, performs much better on these data sets. This might be partially due to VALIS' having an initialisation method that set the radius of a newly generated AB's RR as the distance from the AB to a randomly selected AG. Subsequently, every AB is guaranteed to be connected to at least one AG, and over time, as more ABs are generated, every AG is likely to receive a connection. MAIM, on the other hand, randomly initialises the radiuses of the RRs and is therefore much more dependent on being lucky in having the initial ABs covering enough of the search space. These assumptions are further investigated in test **TL1.5**.

The preceding tests show promising results regarding the classification accuracy of MAIM using local AB_interaction. MAIM generally performs better than the AIS model running by itself. This indicates that the properties of the island model is positive. Table 4.19 gives an overview of minimum and maximum accuracy differences between MAIM with islands and and MAIM-AIS. The minimum column is the accuracy difference between the worst performing island number and MAIM-AIS, while the maximum column is the difference between best performing island number and MAIM-AIS. Here it is shown that MAIM is generally able to improve results on all data sets, with just a few setups being worse than MAIM-AIS. For Sonar the worsening performance is likely a consequence of the unstable results (see test **TL1.4**) and for Iris it is a consequence of too much distribution (see test **TL1.1.3**).

Data set	**Min. Accuracy Diff.**	**Max. Accuracy Diff.**
Iris	-1.4%	1.4%
Wine	0.5%	1.4%
Diabetes	1.7%	3.2%
Heart Statlog	0.1%	1.5%
Breast Cancer Wisconsin	0.5%	0.7%
Glass	0%	2.7%
Ionosphere	1.6%	3.9%
Sonar	-1.1%	5.6%

Table 4.19: Maximum and minimum accuracy differences between MAIM and MAIM-AIS, using local AB_interaction.

To summarise; as expected MAIM performs very well on easily separable data. Further, it is also proving to be good on the more difficult data sets with many samples of each class and smaller feature spaces. This is likely because of the enhanced exploration capabilities of the island model. However, for difficult data sets with many classes and few of each class it is shown to struggle, something which can be counteracted to a certain degree by increasing the population. However, the lack of feature selection is likely also a limitation here. Finally, MAIM is struggling with high dimensional data sets because of insufficient exploration of the search space. The results are generally in line with the hypothesis for these tests (see table 4.7).

It is clear that varying the number of islands changes the results. However, there is no clear trend to whether increasing or decreasing the number of islands is beneficial, aside from avoiding that the IPs become too small. Subsequently, having the right number of islands an IP sizes for the data set is important and results can often by improved by tuning the population sizes. Further, an approach to dynamically tuning island numbers and population sizes is discussed in section 5.3.3. Finally, 4, 6, 7 and 8 islands seem to give the best performance. 6 islands is best for Glass, 5 for Ionosphere, 8 for Wine and finally; 4 for Iris, Diabetes, Heart Statlog, Breast Cancer, and Sonar, making it the best overall. This is, as mentioned, likely due to its compromise between isolated and connected IP evolution.

Accuracy With global AB_interaction

The following tests will be conducted using global information through the global AB_interaction parameter. As explained in section 3.2.7, when this is enabled the interaction of the AGs across all islands is calculated at the master and communicated to the slaves where it is used in the sharing factor calculation (see section 3.2.7). However, in order to reduce the amount of tests conducted only 4, 6 and 8 islands was selected as they all gave good overall performance in the local AB_interaction tests.

The table 4.20 presents the results of **TG1.1** on fairly easily partially separate data. As seen from these results, the tests conducted with local AB_interaction overall performed better. However, on the Iris data set, the results obtained from test **TG1.1** running with six islands are equal to test **TL1.1** running with four islands, which were the best results of the test. However, by comparing the SD of **TG1.1** with that of **TL1.1**, it indicates local AB_interaction is more stable for this data set. Finally, while Wine performs better with local information on all tests, it should be noted that also here 8 islands are the best.

Algorithm	**Wine**	**Iris**
MAIM-4	0.96 (0.011)	0.949 (0.012)
MAIM-6	0.96 (0.009)	**0.965 (0.017)**
MAIM-8	**0.966 (0.015)**	0.941 (0.009)

Table 4.20: TG1.1 Accuracy tests with global AB_interaction on partially separable data.

Table 4.21 presents the results of test **TG1.2** on more difficult, non-separable data. Again, by comparing these results to test **TL1.2**, local AB_interaction overall performs the best. However, for 6 islands there are slight improvements to be achieved over local information on the Diabetes and Heart Statlog data sets. Finally, 4 islands perform the best on all data sets, similarly to test **TL1.2**.

Algorithm	**Pima Diabetes**	**Heart Statlog**	**Wisconsin Breast Cancer**
MAIM-4	**0.744 (0.005)**	**0.806 (0.011)**	**0.966 (0.001)**
MAIM-6	0.743 (0.007)	0804 (0.004)	0.964 (0.002)
MAIM-8	0.743 (0.006)	0.797 (0.007)	0.964 (0.003)

Table 4.21: TG1.2 Accuracy tests with global AB_interaction on non-separable data.

The table 4.22 presents the results of test **TG1.3** on the non-separable data containing many classes. By comparing the results of this test to **TL1.3** once again the test with local AB_interaction is overall performing better. Further, also for this test a higher number of islands is showing the best results. The result with eight islands is close to the results obtained with six islands in **TL1.3**. Subsequently, another test, **TG1.3.1**, have been conducted with increased population size to see if the results are improved. As a population of 1070 proved the best in test **TL1.3.1**, this was also be selected for test **TG1.3.1**. The results are presented in table 4.23. Further, the results are almost identical to the results obtained in test **TL1.3.1**, with a slightly smaller SD.

Algorithm	**Glass**
MAIM-4	0.621 (0.021)
MAIM-6	0.639 (0.008)
MAIM-8	**0.640 (0.016)**

Table 4.22: TG1.3 Accuracy tests with global AB_interaction on non-separable data with many classes.

Population Size	**Result**
1070	0.674 (0.0018)

Table 4.23: TG1.3.1 Accuracy test on the Glass data set, with global AB_interaction, eight islands and large population.

The last test with global AB_interaction, **TG1.4**, is presented in table 4.24. Yet again, the global AB_interaction overall does not perform as well as the corresponding local tests of **TL1.4**. However, 6 and 8 islands with global information on Sonar perform better than their local counterparts. Additionally, for these tests SDs are generally lower with global information than with local.

Algorithm	**Sonar**	**Ionosphere**
MAIM-4	0.631 (0.011)	**0.641 (0.0001)**
MAIM-6	**0.646 (0.029)**	**0.641 (0.0001)**
MAIM-8	0.629 (0.018)	**0.641 (0.0001)**

Table 4.24: TG1.4 Results from accuracy tests with global AB_interaction on data with large feature spaces.

Overall, it can be seen that employing an island structure is beneficial also here, when compared to the MAIM-AIS results of **TL1.1-1.4**. However, perhaps surprisingly, global information has a limited positive impact on performance, as seen from comparisons between the best results achieved with the different approaches in figure 4.17. For 4 islands the performance is worse than with local information on all tests, making local information the overall best performer and indicating that island independence is important. This is contrary to the hypothesis for these tests in table 4.3.1. However, with 6 islands, Iris, Diabetes, Heart Statlog and Sonar all show a slight improvement over the results with local information and specifically for Sonar, the performance on 8 islands is also improved. Furthermore, for Glass and Wine, 8 islands overall performed the best, 6 for Iris and Sonar, 4 for Diabetes, Heart Statlog and Breast Cancer and Ionosphere performs the same on all tests. This indicates that the optimal number of islands fluctuates more with global information, as 4 islands is overall the best with local information. Further, the SDs of Ionosphere, Glass and considerably also for Sonar are reduced with global information, suggesting the reliance on good initialisation might be reduced as opposed to with local information. This is likely due to the search being directed by the master island when using global information, forcing each island to explore separate parts of the search space.

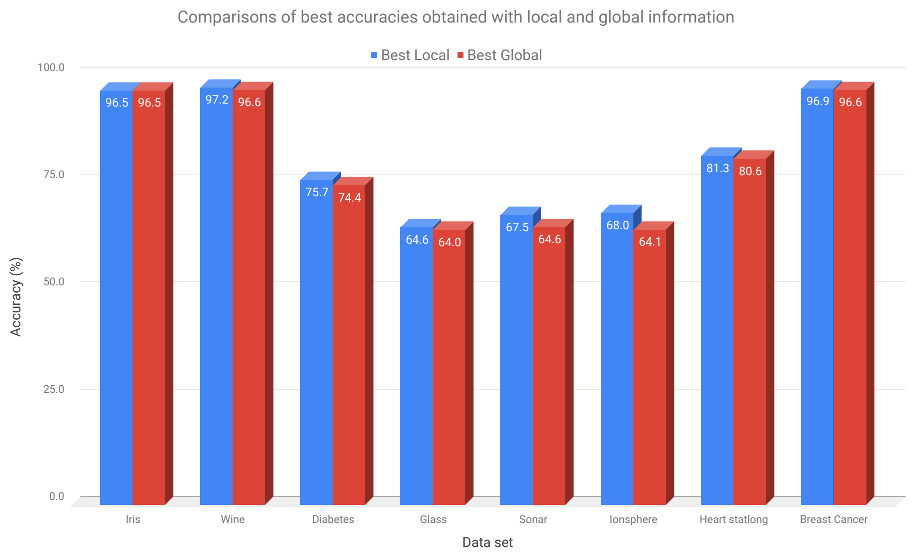


Figure 4.17: Comparisons between best accuracies achieved with local and global information.

MAIM-4 in Comparison with Non-AIS Classifiers

After tests conducted on both local and global AB_interaction it can concluded that overall, MAIM-4 using local AB_interaction achieves the best results. Therefore, the results of MAIM-4 with parameters as in table 4.8 are compared to the results of kNN, *support vector machine (SVM)* and *random forest (RF)* gathered from the work of AISLFS [8]. The comparisons can be seen in table 4.25.

Data Set	MAIM-4	kNN	SVM	RF
Ionosphere	0.671 (0.021)	0.862 (0.009)	**0.952 (0.004)**	0.935 (0.004)
Glass	0.640 (0.021)	0.696 (0.014)	0.682 (0.02)	**0.788 (0.014)**
Breast Cancer	0.969 (0.001)	0.967 (0.003)	**0.976 (0.003)**	0.962 (0.003)
Iris	**0.965 (0.006)**	0.954 (0.008)	0.951 (0.005)	0.951 (0.007)
Wine	0.967 (0.003)	0.964 (0.006)	**0.985 (0.005)**	0.981 (0.003)
Dabetes	0.757 (0.006)	0.752 (0.006)	**0.767 (0.005)**	**0.767 (0.005)**
Heart Statlog	0.813 (0.008)	0.83 (0.013)	**0.843 (0.007)**	0.827 (0.009)
Sonar	0.675 (0.085)	**0.86 (0.015)**	0.844 (0.012)	0.0834 (0.012)

Table 4.25: MAIM-4 comparisons with kNN, SVM and RF.

Not surprisingly, MAIM-4 is challenged on the Ionosphere, Sonar and Glass data sets as a consequence of a lack of exploration in larger feature spaces and poor performance with many classes, as highlighted in tests **TL1.3** and **TL1.4**. However, on the other data sets MAIM-4 arguably achieves results on par with the other algorithms while achieving the overall best results on Iris. The results are generally promising and indicates that the algorithm posses a good generalisation ability which can likely be further improved through improving AB exploration and tuning parameters.

Accuracy Tests With New Initial Radius

The results of test **TL1.4** showed that further work was needed on large feature spaces. Therefore, a new initial radius for the ABs, similarly to what VALIS is doing, was implemented in MAIM. Subsequently, new radiuses are initialised as the distance between the AB and a random AG of its class. This approach was investigated in test **TL1.5.1** with standard parameters as in table 4.8 and extensive accuracy results in appendix .1.1. Incidentally, the new tests were conducted on all data sets with up to 12 islands showing some interesting results. For instance, for 9 islands Wine achieved an accuracy of **0.9755**, for 10 islands Heart Statlog achieved an accuracy of **0.8303** and for 11 islands Diabetes achieved an accuracy of **0.7581**. These are the highest accuracies achieved for these sets, indicating their preference for high genetic isolation and showing there is a point to testing large island numbers, in terms of accuracy.

Through the new accuracies achieved it was clear the change also gave better results on the Sonar and Ionosphere data sets. This is visualised in figure 4.18, where the best results with new initial radius are compared against the best results obtained from local AB_interaction in **TL1.4**. Furthermore, it should be noted that even with the change 4 islands remain the best for the Sonar data set and 5 for Ionosphere. Furthermore, in **TL1.3.1** it was shown that a larger population improved the accuracy on the Glass data set. Consequently, new tests, **TL1.5.2** and **TL1.5.3**, were conducted on both the Sonar and Ionosphere data sets with 4 and 5 islands, respectively. Additionally, since the feature space of these data sets were very large, it was also assumed that more training time (iterations) could enhance the results and give the algorithm sufficient time to explore the search space.

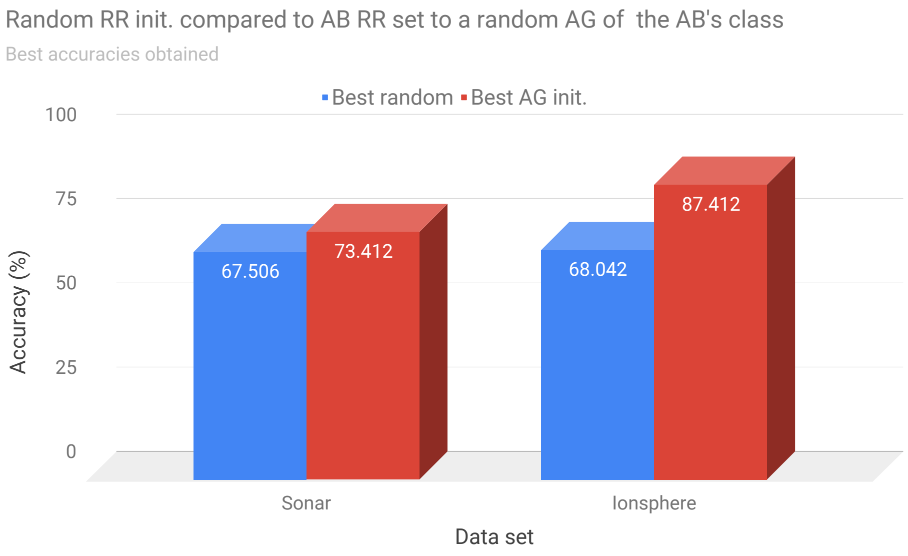
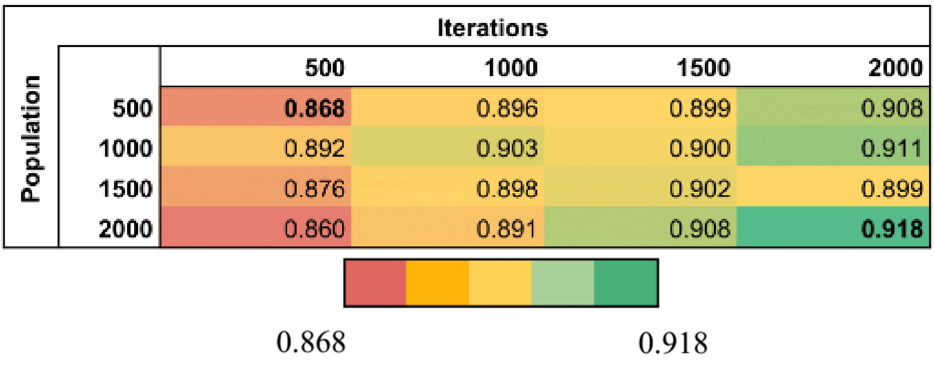


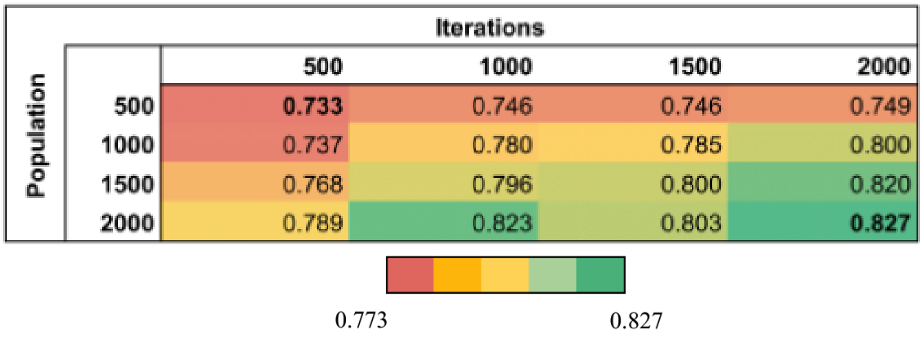
Figure 4.18: TL1.5.1 Comparison of accuracies between random RR initialisation and AB RRs initialised to a random AG of the AB's class.

The new tests were conducted by parameter sweeping the population size and iterations of the algorithm. The results are presented in heat maps in see figure 4.19 and 4.20. Here, more red indicate worse results, while more green indicate better. These tests shows that MAIM is performing better than VALIS on the Sonar data set and AIRS on Ionosphere. That such a small change of the algorithm, along with increases in iterations and population sizes, is providing such improvements in the results is promising. Further, it goes to show just how important AB initialisation and sufficient exploration is for the search. Moreover, a substantial amount of new radiuses created are larger than the old, as there is no longer any limit to their initialised range (see section 3.2.3). Subsequently, with the change the RR cover larger parts of the search space and include more AGs in the function approximation. Consequently, it can be concluded that new initialisation and larger radiuses, along with more exploration through iterations and population sizes, are thus positive.



		Iterations			
		500	1000	1500	2000
Population	500	**0.868**	0.896	0.899	0.908
	1000	0.892	0.903	0.900	0.911
	1500	0.876	0.898	0.902	0.899
	2000	0.860	0.891	0.908	**0.918**

Figure 4.19: TL1.5.2 Accuracy heatmap of the Ionosphere data set.



		Iterations			
		500	1000	1500	2000
Population	500	**0.733**	0.746	0.746	0.749
	1000	0.737	0.780	0.785	0.800
	1500	0.768	0.796	0.800	0.820
	2000	0.789	0.823	0.803	**0.827**

Figure 4.20: TL1.5.3 Accuracy heatmap of the Sonar data set.

Additionally, the results indicate that performance can be improved from increasing just the iterations and population sizes by themselves. Further, when increased in conjunction they complement each other and even further improve the results. Through this one can conclude that these parameters are very important for the search and subsequently the result, indicating that when one is increased the other may reasonably be increased as well. So far increasing the population has generally been shown to have positive affect on both accuracies and stability (SDs), as seen in **TL1.1.2**, **TL1.1.3 TL1.3.1**. Consequently, it would be interesting to see just how far the results can improve by continuing to increase these parameters. Unfortunately, this is outside the scope of this work, but could be investigated through a dynamic island approach, as discussed in 5.3.3.

It should be noted that such large population sizes an iterations are not required to achieve similar results with VALIS and AIRS, indicating that their exploratory ability is not as dependent on these parameters. Subsequently, it is reasonable to assume that MAIM could benefit from forcing a wider exploration of the search space, either through increasing the punishment of AG sharing or the integration of a feature selection approach (see section 5.3.1).

Accuracy Tests With Class Distributions

In the VALIS algorithm ABs employ class distributions instead static classes as in this work. This means that the ABs receive several different accuracies depending on the distribution of AGs they connect to. An example of this is an AB covering 3 AGs in total; 1 AG of class a and 2 AGs of class b. This gives the AB a class distribution of $\frac{1}{3}$ for a and $\frac{2}{3}$ for b, which are subsequently their accuracies for the respective classes. Further, these distributions are used in both the AB fitness and voting function, making each AB able to recognise more than one class of AG at once, something not possible with the static class approach of MAIM. By using this approach in MAIM, voting and fitness is calculated identically to VALIS.

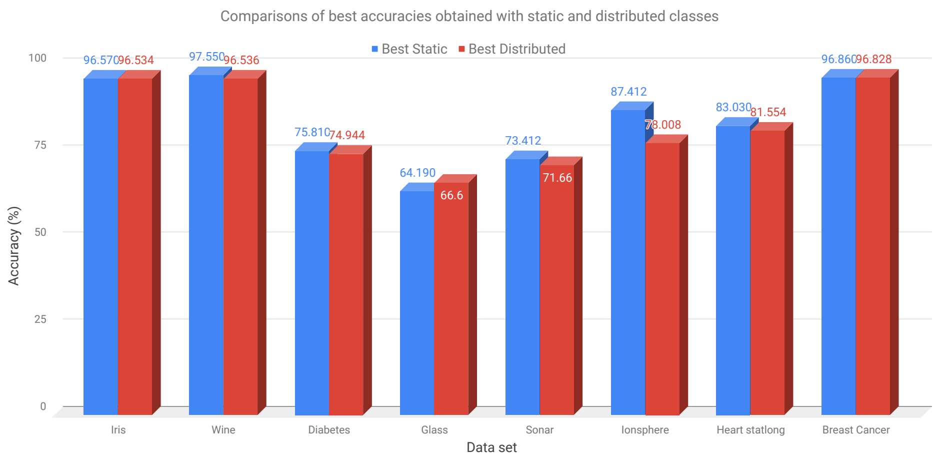


Figure 4.21: Comparisons between static and distributed classes.

An additional test was conducted to compare static and distributed classes, **TL1.6**, using local AB_interaction, class distributions and the initialisation method on **TL1.5**. Extensive accuracy results from testing the class distribution technique can bee seen in appendix .1.2. Based on the results, a comparison between the results from the static class approach of **TL1.5** in appendix .1.1 was compared to the distributed class approach in figure 4.21. Note that the results were selected from the best-performing island numbers of each approach. Further, it is shown that the static class approach in test **TL1.5**, generally performs better than the class distribution approach on most of the data sets tested. However, some benefit can gained from using class distributions, specifically on the Glass data set where it is shown to perform better than all other approaches tested. While MAIM with static classes is able to achieve an accuracy of 64.19%, class distributions achieve 66.6%. Furthermore, while the differences are arguably not large, it indicates that class distributions are better on data sets with many classes that are hard to separate. This is perhaps not surprising as class distributions are able to classify AGs of several different classes at once. Subsequently, class distribution ABs are likely better at classifying AGs in areas containing many overlapping and hard to separate classes, making it difficult for ABs to only interact with AGs the same class. However, static classes still seem to be the best-performing approach overall. This is in line with the hypothesis of table 4.3.1, indicating that the static classes makes the ABs more specialised and subsequently better at its assigned task, which is to correctly classify only one class of AGs.

4.5.2 Results Phase Two - Efficiency Testing

This section will present and elaborate on the results of the efficiency tests, and are separated into six different parts, **T2.1 - T2.6**, as described in table 4.3.1 with setups as described in section 4.4.2.

Efficiency Tests with local AB_interaction

The first test, **TL2.1**, with a population size of 500 is presented in table 4.26 and visualised in figure 4.22. As the results show, MAIM is performing better than AIS only on all island numbers for Sonar and Wine. Further, on Iris it performs better with 4 and 8 islands and finally, on Diabetes, it only performs better with 4.

This indicates that the overhead of migrating between twelve islands on 20% of the iterations, gives a substantial amount of overhead when the population size is small. This is especially true for the Diabetes data set which is substantially larger than the others.

data set / Islands	**AIS Only**	**4 Islands**	**8 Islands**	**12 Islands**
Iris	16.2 (0.25)	**14.64 (0.19)**	15.56 (0.19)	17.06 (0.24)
Wine	28.06 (0.13)	**22.76 (0.27)**	23.89 (0.12)	26.66 (0.34)
Diabetes	81.29 (0.69)	**79.78 (1.33)**	91.31 (1.17)	104.99 (0.63)
Sonar	85.97 (1.61)	**71.07 (0.55)**	76.34 (1.30)	83.04 (1.02)

Table 4.26: TL2.1 Efficiency tests with population size 500 and local AB_interaction.

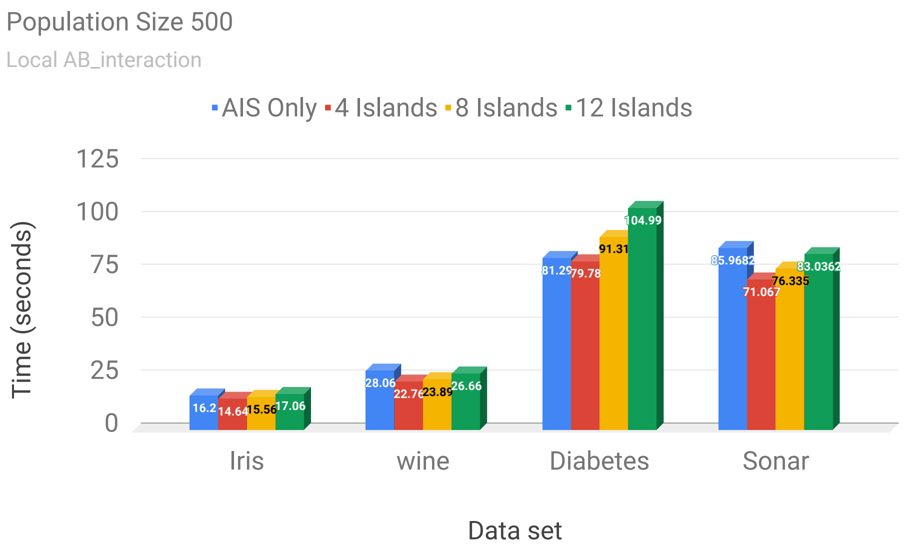


Figure 4.22: TL2.1 Efficiency tests comparisons with population size 500 and local AB_interaction.

The test **TL2.2** with a population size of 1000 is presented in table 4.27 and figure 4.23. The results indicate that the island setups perform increasingly better than AIS only as the population increase. Further, in this test all the different island setups perform better than AIS only. 12 islands is still not as favourable on the Diabetes data set, but the overhead of migration has proportionally become smaller.

data set / Islands	AIS Only	4 Islands	8 Islands	12 Islands
Iris	46.5 (0.27)	31.56 (0.38)	**29.47 (0.27)**	29.99 (0.17)
Wine	78.51 (0.55)	48.21 (0.34)	**47.30 (0.34)**	48.26 (0.25)
Diabetes	193.33 (1.73)	**152.65 (1.43)**	168.24 (2.08)	179.06 (0.67)
Sonar	197.47 (4.47)	**142.60 (1.21)**	143.81 (0.82)	149.94 (1.35)

Table 4.27: TL2.2 Efficiency tests with population size 1000 and local AB_interaction.

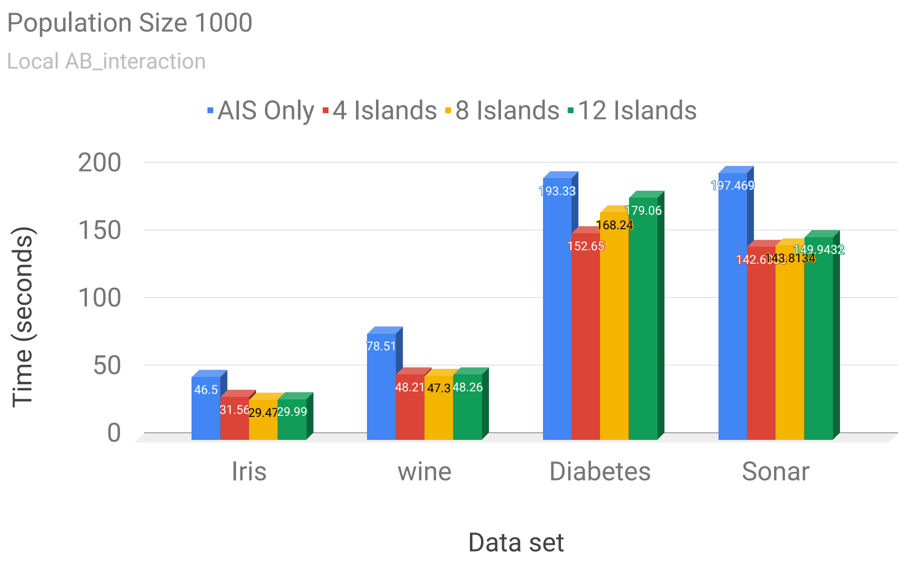


Figure 4.23: TL2.2 Efficiency comparisons with population size 1000 and local AB_interaction.

The test **TL2.3** for a population size of 1500 is presented in table 4.28 and figure 4.24. Further, the results are consistent with assumptions of test **TL2.2**, showing clear indications that MAIM perform increasingly better than AIS only for larger populations. Further, the higher island numbers seem to do increasingly well in relation to other numbers as population increase. Training time of MAIM here is reduced by 59% on Iris running twelve islands compared to AIS only.

data set / Islands	**AIS Only**	**4 Islands**	**8 Islands**	**12 Islands**
Iris	110.53 (1.28)	52.9 (0.7)	48.99 (1.18)	**45.03 (0.32)**
Wine	168.38 (2.81)	78.48 (0.86)	77.73 (1.04)	**74.98 (0.88)**
Diabetes	376.67 (6.24)	**231.28 (1.12)**	251.24 (3.42)	265.23 (1.03)
Sonar	352.25 (6.54)	222.62 (1.1)	**214.42 (2.42)**	218.24 (2.07)

Table 4.28: TL2.3 Efficiency tests with population size 1500 and local AB_interaction.

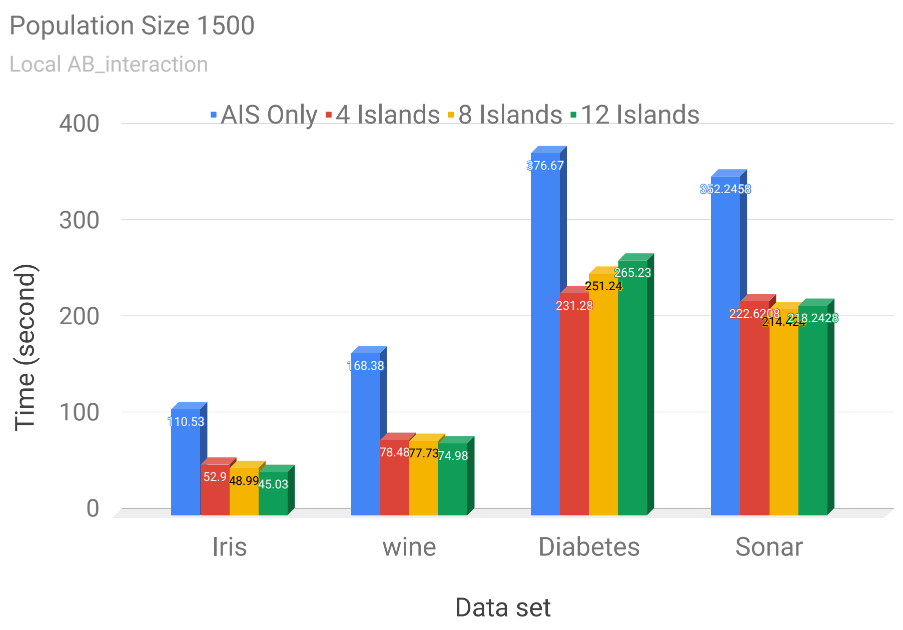


Figure 4.24: TL2.3 Efficiency test comparisons with population size 1500 and local AB_interaction.

The test **TL2.4** for population size of 2000 is presented in table 4.29 and figure 4.25. The results further indicate that that MAIM become increasingly efficient with larger populations simultaneously as increased distribution through employing more islands become increasingly effective. This is in line with the hypothesis of table 4.7.

data set / Islands	**AIS Only**	**4 Islands**	**8 Islands**	**12 Islands**
Iris	215.5 (9.68)	72.49 (5.38)	68.12 (1.3)	**64.07 (1.89)**
Wine	301.81 (9.86)	113.26 (1.9)	105.7 (1.27)	**101.6 (0.84)**
Diabetes	604.20 (1.59)	**316.25 (1.23)**	334.38 (3.72)	343.43 (1.4)
Sonar	556.69 (8.91)	301.95 (2.52)	288.02 (1.44)	**287.18 (5.07)**

Table 4.29: TL2.4 Efficiency tests with population size 2000 and local AB_interaction.

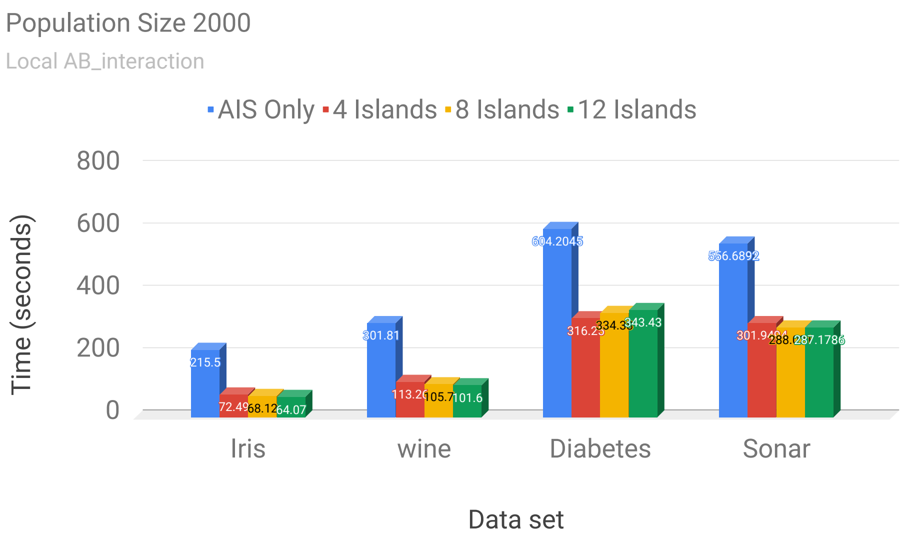


Figure 4.25: TL2.4 Efficiency test comparisons with population size 2000 and local AB_interaction.

As seen for a population of 2000, 12 islands is best-performing on Iris and Wine and second best on Diabetes. It is likely safe to assume that for Sonar 12 islands will perform the best by simply increasing the population a little more. However, for Diabetes 12 islands still perform worse than both 4 and 8. This might be a consequence of the size and AG layout in the Diabetes data set. With many AGs in a normalised space it is likely to contain areas of higher AG density than the other sets. This in turn leads to a greater risk of AB congestion for higher populations (see section 4.2.4). However, this can be counteracted by increasing the distribution. Further, following this assumption the difference between the island setups will continue to grow as populations increase, eventually making 12 islands better. To test this assumption, a new test, **TL2.5**, was conducted with population sizes of 4000 and 6000 on the Diabetes data set with results in table 4.30 and figure 4.26. The results indicate that the assumption is true, as the difference between the setups increase in relation to the distribution. Consequently, for a population size of 6000, 12 islands performs the best overall, while the difference between 4 and 8 has also increased in favour of 4. Finally, it can be seen that training times for AIS only has almost doubled from 4000 to 6000.

data set / Islands	AIS Only	4 Island	8 Island	12 Island
Diabetes (4000)	1615.902 (4.1)	761.321 (3.44)	**651.743 (4.44)**	653.163 (5.31)
Diabetes (6000)	3225.211 (6.21)	1302.02 (5.224)	1064.125 (5.952)	**980.128 (7.44)**

Table 4.30: TL2.5 Efficiency tests with population sizes of 4000, 6000 and local AB_interaction on the Diabetes data set.

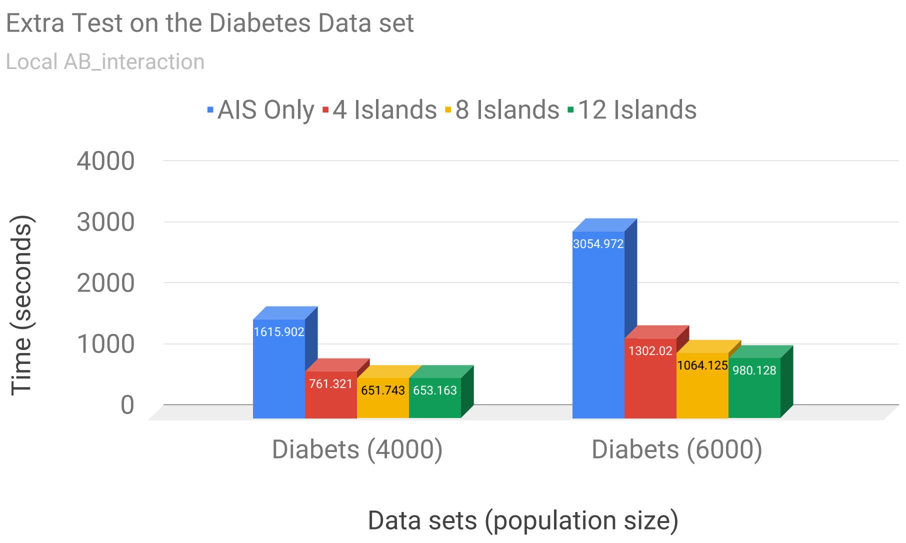


Figure 4.26: TL2.5 Efficiency test comparisons with population sizes of 4000, 6000 and local AB_interaction on the Diabetes data set.

It should be noted that the increase in population for these particular data sets does not necessarily improve the accuracy of the classification as optimal population sizes vary between data sets (detailed accuracy results can be found in appendix .2). However, it does prove that if high population sizes are necessary, like for instance as in section **TL1.5**, an increased distribution through deploying more islands is beneficial.

A population size of 500 was determined to be a small population in the tests conducted, but this is only relative to the other population sizes tested. 500 is not necessarily a small population size for data sets that are already small and easily separable. For instance, Iris only contain 150 cases and 4 features while also being relatively easily separable. Consequently, a population size of 500 might be an unnecessarily high for this data set. Subsequently, it would be interesting to look at the run times for Iris with different island setups using a population size of 150. According to the hypothesis of 4.7 the overhead of using islands is likely to become too big for smaller populations.

An additional test, **TL2.6**, has been conducted for all the data sets using population sizes of #AGs. The results are presented in table 4.31 and figure 4.27 for three and four islands. Three islands have here been added to this experiment to see if it is more efficient than four islands for these populations sizes. The Results indicate that the hypothesis of overhead does in fact become an increasing problem for smaller population sizes. With a population of 150, Iris is more efficiently ran on a AIS only than on both 3 and 4 islands. However, the differences in run times are almost negligible. Additionally, three islands is slightly faster than four, as a result of having slightly more overhead. In all likelihood two islands would further decrease the run times for MAIM on Iris, but at that point the properties of the island model will likely diminish and negatively impact accuracies (see section 4.5.1). Finally, for all data sets tested except Iris, three islands is, to varying degrees, more efficient when using #AGs ABs.

data set (population size)	**AIS Only**	**3 Islands**	**4 Islands**
Iris (150)	**4.79 (0.075)**	5.01 (0.0094)	5.41 (0.0037)
Wine (178)	9.50 (0.1312)	**9.28 (0.0026)**	9.95 (0.0048)
Diabets (768)	150.017 (1.37)	**124.85 (0.0043)**	127.19 (0.931)
Sonar (208)	34.18 (1.12)	**30.67 (0.497)**	33.203 (0.293)

Table 4.31: TL2.6 Efficiency tests with population size #AGs and local AB_interaction.

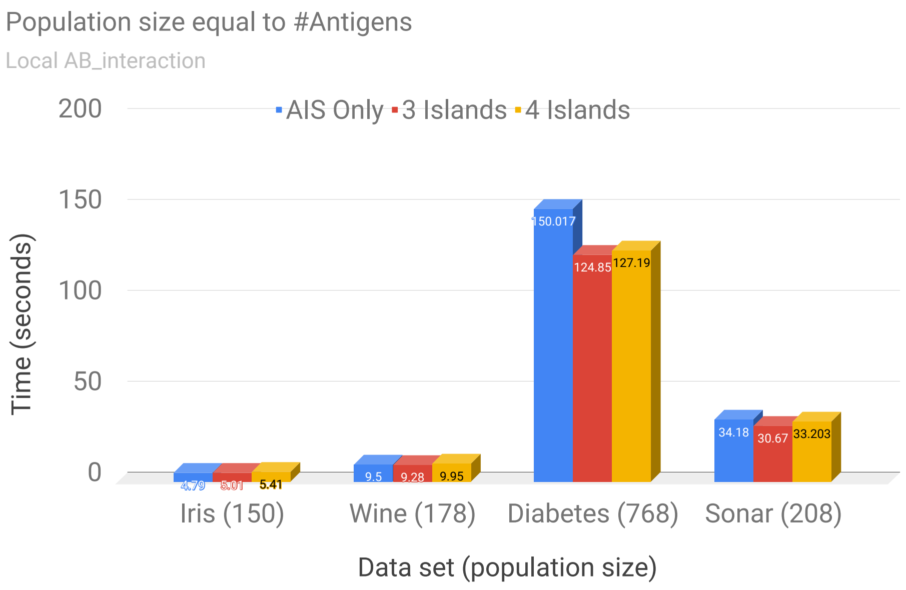


Figure 4.27: TL2.6 Efficiency test comparisons with population size #AGs and local AB_interaction.

Run Time Graphs - Figure 4.28, 4.29, 4.30 and 4.31 illustrates the run time graphs for different datasets and islands setups, gathered from data in **TL2.1-2.5**. As the figures shows, the different island setups have closer to linear increases, while AIS only have much more exponential increases. However, much like the diagrams in the preceding tests, the graphs vary depending on the data set. For instance, AIS only on the Sonar data set have a more drastic increase early, making an island structure beneficial right from the start at 500 ABs. On the other hand, for Diabetes, islands are not beneficial before employing around 1000 ABs. Further, for sonar, 12 islands become better than 4 at population sizes of around 1500, while for Diabetes 12 islands is still the worst-performing island setup at 2000.

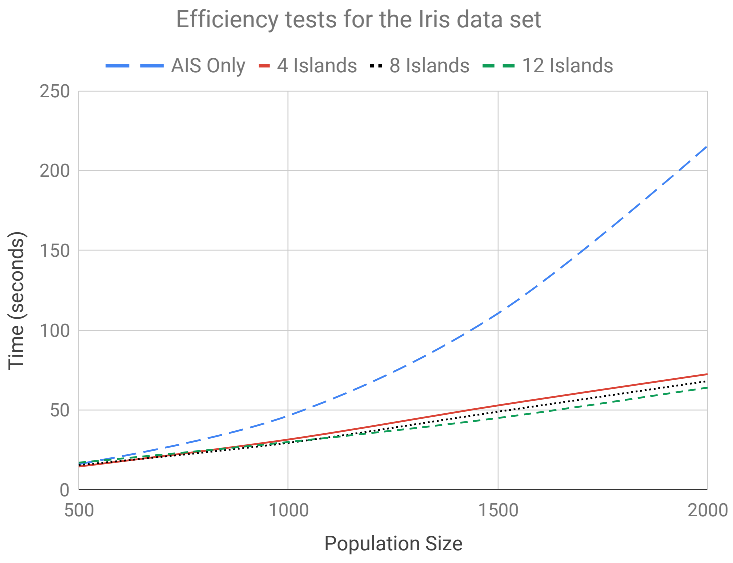


Figure 4.28: MAIM run time graph for the Iris data set.

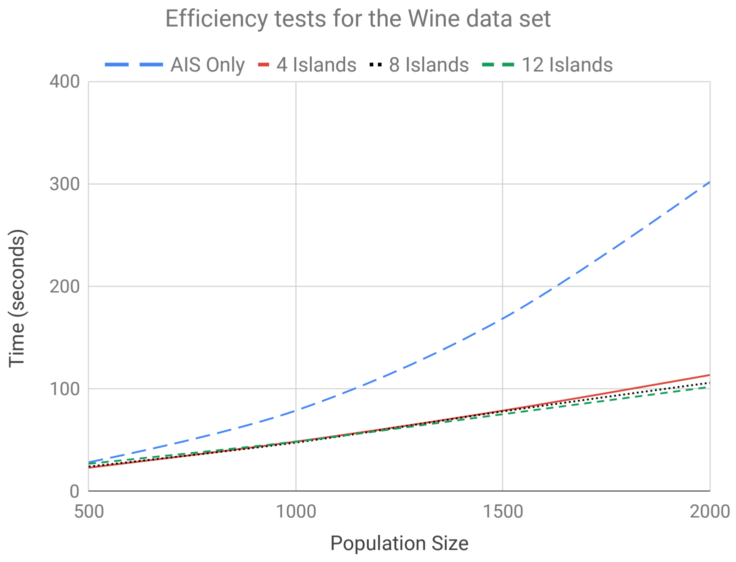


Figure 4.29: MAIM run time graph for the Wine data set.

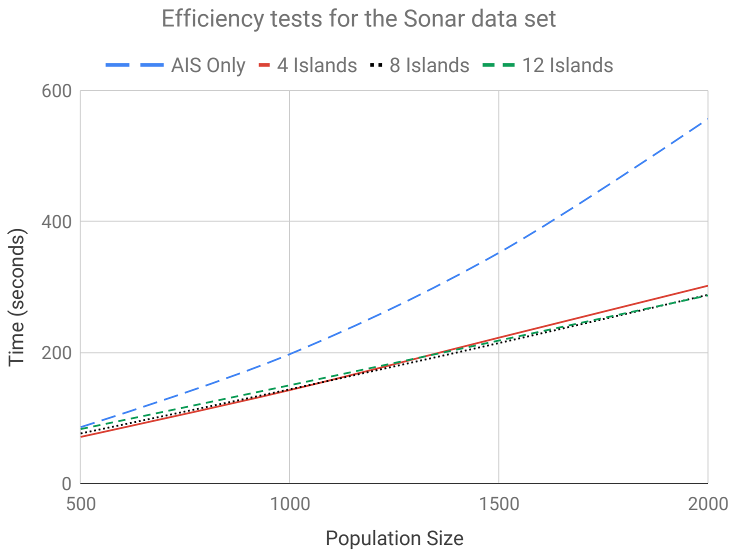


Figure 4.30: MAIM run time graph for the Sonar data set.

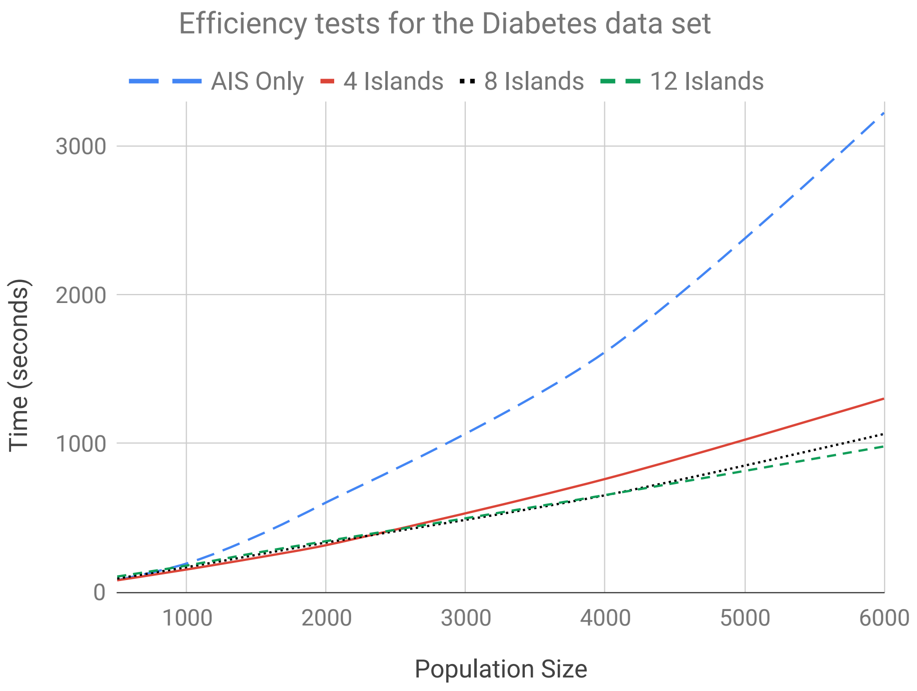


Figure 4.31: MAIM run time graph for the Diabetes data set.

In addition to in tests **TL2.1-2.4**, Diabetes was, as the only set, further tested in test **TL2.5**. Therefore, figure 4.31 presents the combined data from all efficiency tests on Diabetes. Here it can be seen that linear efficiency is obtained with 4 islands until the population reaches 4000. At this point, the islands become congested as there are too many concurrent AB-AG connections, exponentially increasing the computational burden of the fitness calculation. However, when this happens, further spreading the population over 8 islands reduces the AB congestion on each island and subsequently the computational burden. As a consequence, for the case of population sizes of 6000 or above, employing 12 islands is the most efficient.

The exponential increases in run times are a result of too many ABs interacting with too many of the same AGs, i.e. too much AG sharing between ABs. This is a result of the AB to AG ratio becoming too large. Subsequently, increasing the population sizes increases the amount of ABs having to reside in areas of high AG density, resulting in the total number of interactions exponentially increasing as increasing amounts of ABs interact with the same AGs. Further, this heavily impacts the number of calculations having to be done for all affinities and sharing factors to be determined at each iteration. In absolute worst case n^2 affinities and n^2 interaction shares have to be calculated at each iteration if every AB is connected to every AG (see section 3.2.7).

For smaller IPs the sharing factor cause the ABs to sufficiently spread out and connect to fewer AGs on average, alleviating the computational burden. However, for larger IPs this is not possible. Subsequently, at this point and for larger populations, the increase in the number of islands counteracts the inevitable exponential increase in run times by increasing the distribution of the ABs and avoiding congestion.

Efficiency Tests With global AB_interaction

The efficiency tests conducted with global AB_interaction are only conducted for population sizes 500 in **TG2.1** and 2000 in **TG2.4**, as indicated by table 4.7 with setups as described in section 4.4.2. This is because the tests are conducted in order to compare efficiency with local AB_interaction. Additionally, the Sonar data sets is not included for these tests, as it seemed unnecessary since same trend could be seen in all datasets of **TL2.1-2.4**.

Results of **TG2.1** are shown table 4.32 and figure 4.32. Similarly, **TG2.4** is shown in table 4.33 and figure 4.33. Both tests perform worse than their local AB_interaction counterparts, in terms of efficiency, with corresponding accuracy results in appendix .2.

Data set	**AIS Only**	**4 Islands**	**8 Islands**	**12 Islands**
Iris	**16.2 (0.25)**	18.54 (0.26)	19.7 (0.3)	21.04 (0.22)
wine	**28.06 (0.13)**	29.49 (0.34)	32.08 (0.23)	34.41 (34)
Diabetes	**81.29 (0.69)**	96.20 (0.9)	106.58 (0.14)	119.65 (1.47)

Table 4.32: TG2.1 Efficiency tests with population size 500 and global AB_interaction.

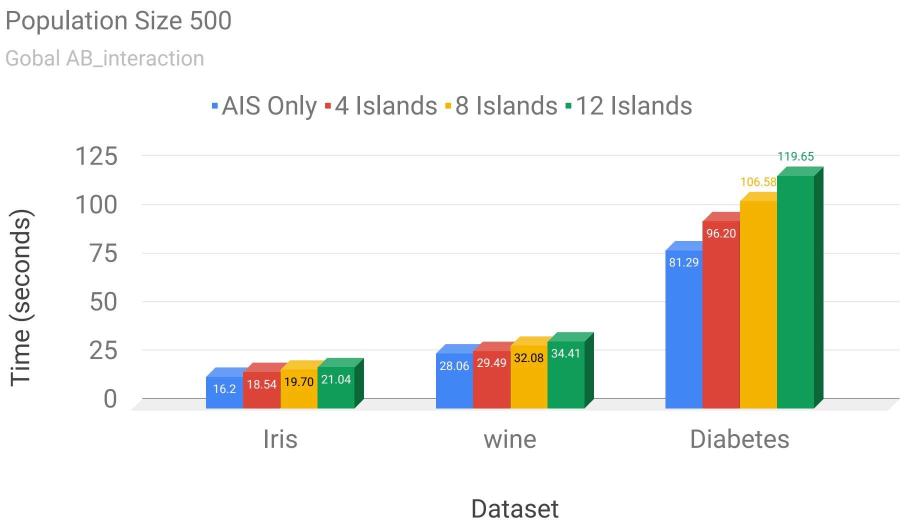


Figure 4.32: TG2.1 Efficiency test comparisons with population size 500 and global AB_interaction.

Data set	**AIS Only**	**4 Islands**	**8 Islands**	**12 Islands**
Iris	215.5 (9.68)	90.48 (1.73)	87.23 (1.31)	**83.43 (2.25)**
wine	301.81 (9.86)	141.51 (1.19)	**132.76 (3.37)**	141.85 (0.44)
Diabetes	604.20	**408.04 (6.14)**	460.15 (8.77)	482.25 (2.64)

Table 4.33: TG2.4 Efficiency tests with population size 2000 and global AB_interaction.

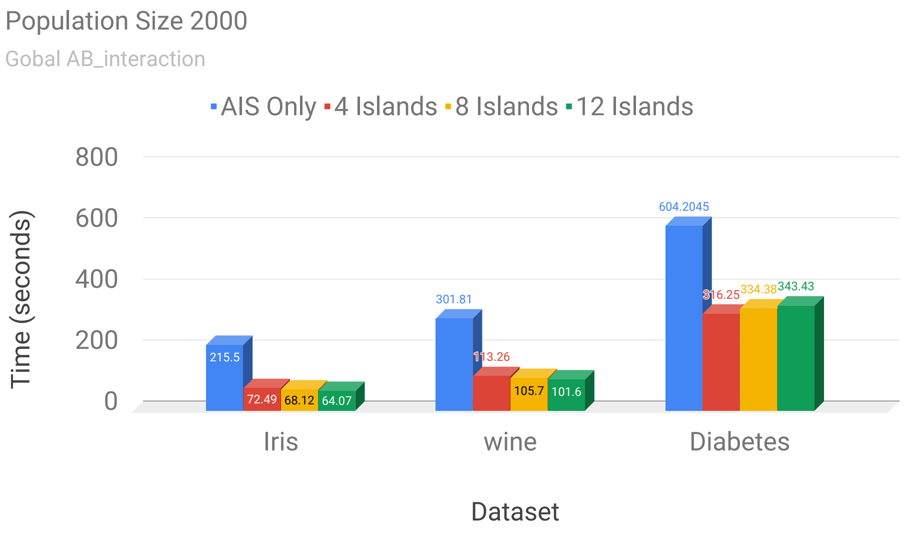


Figure 4.33: TG2.4 Efficiency test comparisons with population size 2000 and global AB_interaction.

Local AB_interaction perform better than global for both smaller and larger population sizes, as indicated by figure 4.34 where the best-performing island setups of 500 and 2000 ABs using local and global information are compared. Here, local information is around 20% faster on all setups shown. Further, it is particularly clear that local information performs better when looking at the smaller population sizes in comparison with AIS only. In test **TG2.1**, AIS only is performing better than all island setups on all data sets, where in **TL2.1** 4 islands is the best. Further, the increased run times from employing global AB_interaction is likely a consequence two things: Firstly, the increased computation needed for calculating the parameter on the master. Secondly, when the master directs the search it forces each island to explore different and more condensed parts of the search space. Subsequently, AB congestion increase with global information as ABs are forced into smaller areas on the islands. On the other hand, with only local information the search is not directed and ABs are free to spread over the whole search space, decreasing AB congestion and consequently run times.

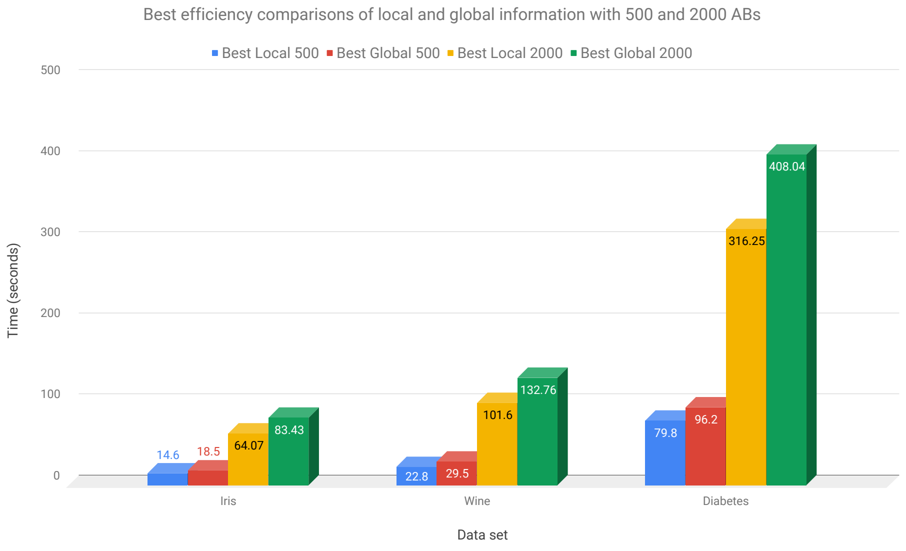


Figure 4.34: Efficiency test comparisons between local and global AB_interaction.

The efficiency results achieved align with the hypothesis of table 4.7 where it was indicated that global AB_interaction would be less efficient. Further, results from all tests conducted so far indicates that overall local AB_interaction performs the best, both in terms of accuracy and efficiency. As a result, there may not be a reason for employing global AB_interaction in the search, except for a few special cases of data set and island number combinations, as discussed in section 4.5.1. Regardless, it is promising that independence between islands is as beneficial as the tests indicate, performing the overall best both in terms of accuracy and efficiency.

4.5.3 Results Phase Three - Effects of Migration Frequency, Rate and Policy

This section will present the results obtained by looking at the effects of MF, MR and policy through parameter sweeping tests. Further, the tests have been conducted over three different data sets, with parameters and setup as explained in section 4.4.3. Every figure in the following chapter displays an accuracy heatmap where more green indicate higher accuracies and more white indicating lower accuracies, as shown by the gradient bar below. Green and white were here selected as the accuracy differences achieved from the parameters are not as great as in test **TL1.5**. Remember, that in the following tests MR refers to what percentage of the population is sent during migration and MF refers how often migration is conducted (see table 3.1).

IGA Effects with local AB_interaction

The following tests present migration effects using local AB_interaction and no-deletion migration.

The figure 4.35 presents the results as a heatmap for test **TL3.1** on the Wine data set. Here, the runs using no migration results in an accuracy of **0.925**. As the figure indicate, although there is no particularly strong trend, the higher accuracies seem to mostly be achieved for lower values of MFs and higher values of MRs. However, migration seem to overall have a substantial positive impact on the result, as a MR of 0.1 and a MF of 0.5 achieves an accuracy of **0.965**, improving accuracies by a whole 4% over no migration.

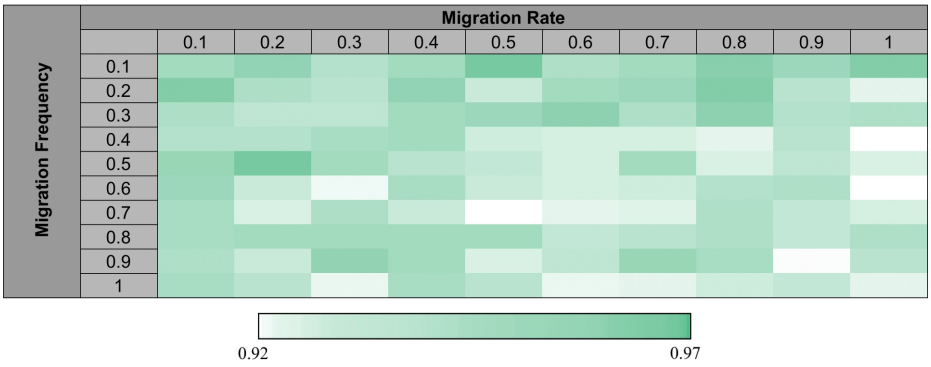


Figure 4.35: TL3.1 Heatmap for the Wine data set using no-deletion migration and local AB_interaction.

The figure 4.36 presents the results as a heatmap for test **TL3.2** on the Diabetes data set. The accuracy obtained from no migration was **0.734**. This test presents a clearer trend than **TL3.1**, that higher accuracies are obtained with lower values for both parameters. Additionally, the Diabetes data set is considered harder than Wine, indicating that for more difficult data, the migration parameters used may have a larger impact on the result. This might not be as surprising as the harder data sets require more exploration to find better solutions, something which is helped by migration. Further, a MR of 0.1 and a MF of 0.5 obtains and accuracy of **0.758**, also here improving on the results without migration.

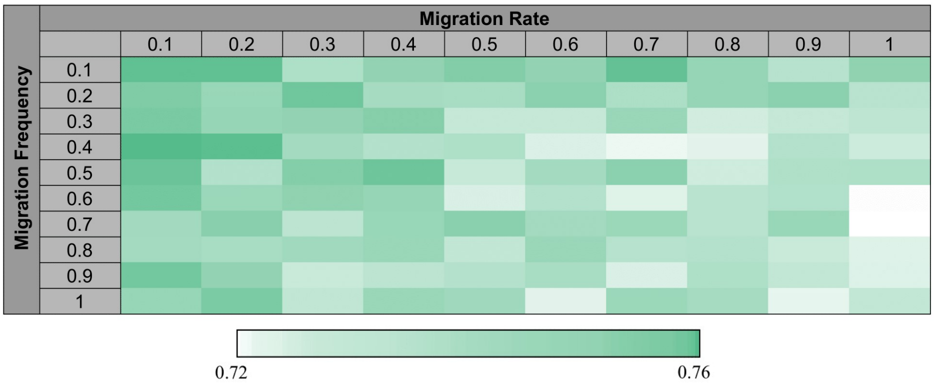


Figure 4.36: TL3.2 Heatmap for the Diabetes data set using no-deletion migration and local AB_interaction.

The figure 4.37 presents the results as a heatmap for test **TL3.3** on the Heart Statlog data set. The accuracy obtained with no migration was **0.821**. This is one of the highest accuracy results achieved by MAIM on this data set, beating AISLFS with its score of **0.818**. Further, this indicates that Hart Statlog might not be as dependent on migration at all, as islands seem to be better of evolving in complete isolation. However, for when migration is employed, the trend is similar to Diabetes in **TL3.2**, but with larger values for MF and MR here being more beneficial. Further, as long as one of the parameters have a low value, the other one can reasonably be high. However, for very high values of both, the results seem to somewhat degrade. Finally, with a MR set to 0.1 and MF to 0.6, the best accuracy with migration was achieved as **0.818**.

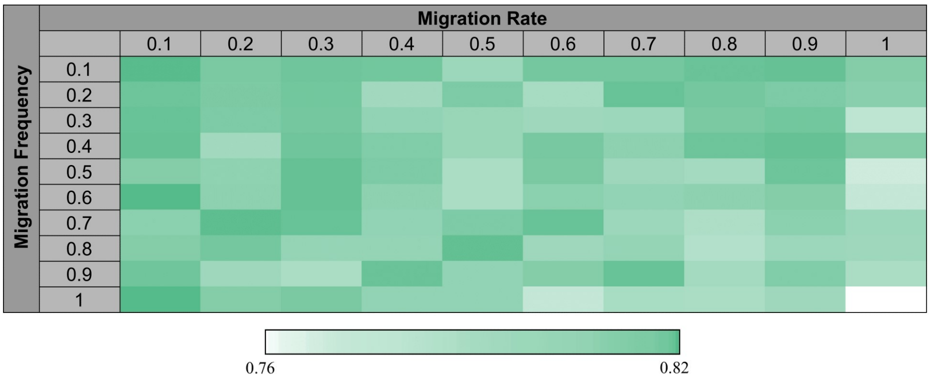


Figure 4.37: TL3.3 Heatmap for the Heart Statlog data set using no-deletion migration and local AB_interaction.

Overall, the best results with migration was achieved with MR of 0.1 and MF of 0.4 – 0.6. This indicates that MR is best kept low while MF can be higher.

The hypothesis, as outlined in table 4.7 indicated that the results would be better with migration than without. However, this was only true for two of the three data sets tested. Further, it indicated that lower migration values would perform better than higher, which partially came true as a combination of one parameter having a low value while another was high also gave good results.

It is interesting that the Heart Statlog dataset perform better without migration. This might be a consequence of the set's properties, preferring high amounts of genetic isolation, as seen in **TL1.5**, were Heart Statlog achieved an accuracy of **0.830** with 10 islands (see appendix .1.1).

Global AB_interaction

The following tests present the migration effects using global AB_interaction and no-deletion migration.

The figure 4.38 presents the results as a heatmap for test **TG3.1** on the Wine data set. The accuracy obtained by running the algorithm with no migration was **0.941**. By comparing these results with the ones obtained in **TL3.1**, one can see that they are quite similar which the exception of global information favouring somewhat larger MFs. However, the results are better for no migration with global information than for no migration with local information. This is perhaps not so surprising, since the search is directed by the master when global information is enabled, likely reducing the need for diversity in the IPs, as islands are forced into exploring different parts of the search space regardless. Consequently, results was improved with a maximum of 1.5% from no-migration to migration, when using global information, and 4% with local information. However, migration still seems to overall be positive, with the best results achieved as **0.956** with MRs of 0.3 and MFs of 0.9.

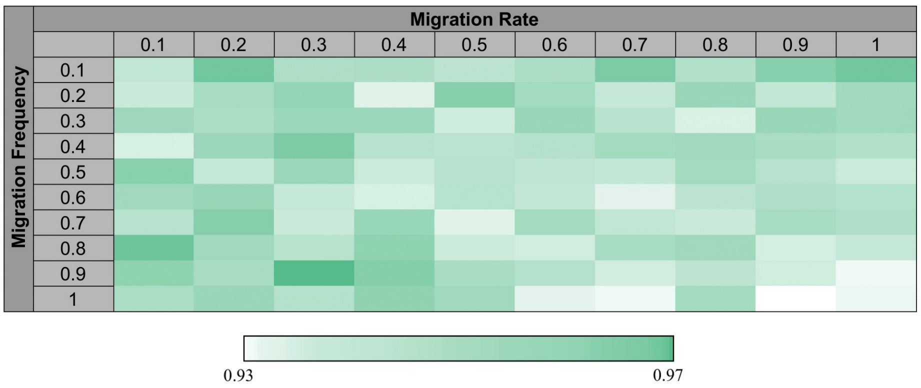


Figure 4.38: TG3.1 Heatmap the for Wine data set, using no-deletion migration and global AB_interaction.

The figure 4.39 presents the results as a heatmap for test **TG3.2** on the Diabetes data set. The accuracy obtained by running the algorithm with no migration was **0.727**. By comparing the results with the results obtained of **TL3.2**, one can see that also here the trends are similar, as lower values of either MR or MF is favoured. However, when using global information it seems that higher MFs are preferred, similar to what was seen in **TG3.1**. Finally, migration was shown to be positive, achieving an accuracy of **0.742** at MRs of 0.2 and MFs of 0.7, indicating that, while positive, the introduction of migration has a smaller impact when employing global information. This assumption is based on the fact that accuracies was improved by a maximum of 1.5% from no-migration to migration, when using global information, as opposed to 2.4% with local information.

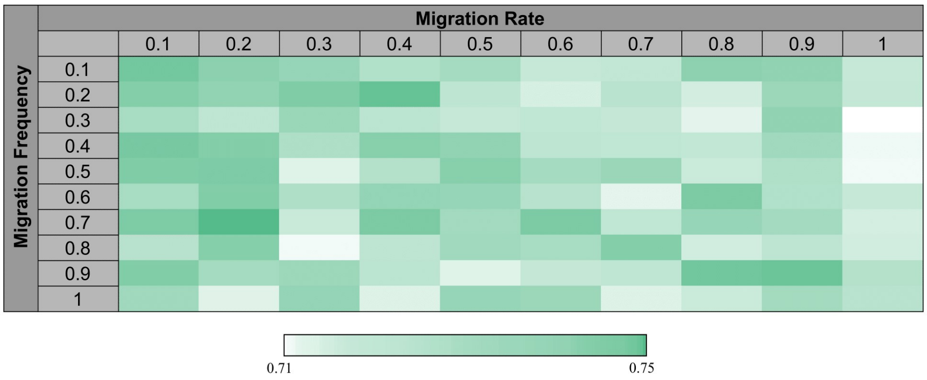


Figure 4.39: TG3.2 Heatmap for the Diabetes data set, using no-deletion migration and global AB_interaction.

The figure 4.40 presents the results as a heatmap for **TG3.3** on the Heart Statlog data set. The accuracy obtained by running the algorithm with no migration was **0.796**. When comparing the results to **TL3.3** there seems to also here be a stronger preference for higher MFs. However, the trend is comparatively not as strong with global information, but it indicates a preference for values in the lower or middle of the map. However, as opposed to **TL3.3**, the introduction of migration seems to improve the results, rather than degrade them, with a maximum accuracy improvement of 1.7%. Subsequently, migration is positive on the Heart Statlog data set with global AB_interaction, achieving an accuracy of **0.813** on MRs of 0.1 and MFs of 0.4.

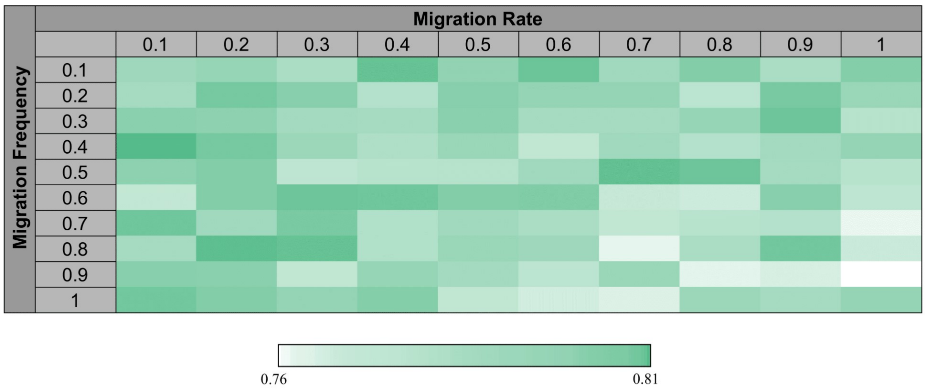


Figure 4.40: TG3.3 Heatmap for the Heart Statlog data set, using no-deletion migration and global AB_interaction.

Overall, with global AB_interaction the best results with migration was achieved with MRs of 0.1 − 0.3 and MFs of 0.4 − 0.9. The trend is similar, although with greater variance, to the results when using local AB_interaction, indicating a preference for lower MR and higher MF.

Generally, migration with global information is positive, while not being as impactful as with local information, which is likely a consequence of the master-directed search not being as dependent on diversity. Further, this is consistent with the test hypothesis of table 4.7. Additionally, while the trends are similar to that of local information, they seem to be less clear and vary more in optimal values. This is likely also consequence of the IPs exploring different parts of the search space, resulting in migration between islands whose ABs are meant to solve completely different sub-problems. Subsequently, this reduces the impact of migration.

IGA Effects with Local AB_interaction and Deletion Migration

While the previous section looked at a combination of MR and MF using non-deletion migration, this section will evaluate tests for migration with deletion (see section 3.1.3). As such it will suffice to look at overall accuracies and patterns for the following heatmaps in comparisons with tests **TL3.1-3.3**. Here only local AB_interaction is included as they were shown to be the most stable in tests **T3.1-3.3**.

The results of **TL3.4** is presented in figure 4.41, 4.42 and 4.43. Interestingly, the Wine data set shows different patterns than what was seen in **TL3.1**, with stronger accuracies towards the middle of the map, indicating that too high and too low MRs might not be as favourable. However a MR and MF of 0.9 achieves the best accuracy overall of **0.972**

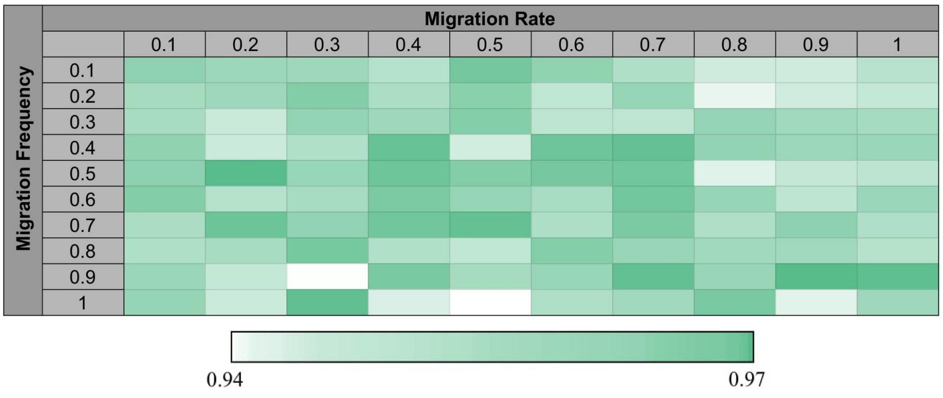


Figure 4.41: Heatmap for the Wine data set with deletion migration and local AB_interaction.

The Diabetes data set shows strong patterns, clearly favouring lower MRs. However, MF can be higher as long as the MRs are lower, achieving an accuracy of **0.752** at MRs of 0.1 and MFs of 0.9.

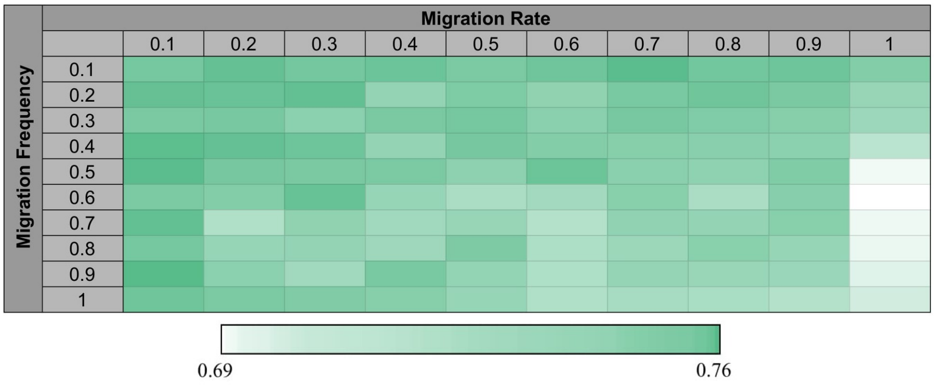


Figure 4.42: Heatmap for the Diabetes data set with deletion migration and local AB_interaction.

Heart Statlog show no clear trend, but seem to also favour lower MR. However, the best result was **0.817** achieved at both MR and MF of 0.5.

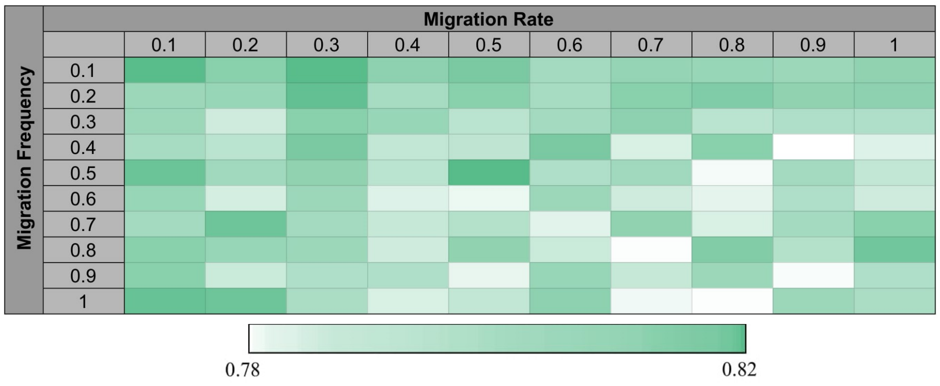


Figure 4.43: Heatmap for the Heart Statlog data set with deletion migration and local AB_interaction.

Overall, stronger results was achieved without deletion for Diabetes with a total average of the no-deletion accuracies as **0.741** and a higher max accuracy, while deletion-migration had an average of **0.735**. Further, with deletion the Wine data set performed better with a total average of **0.959** and a higher max than no-deletion which averaged **0.953**. Finally, for Heart Statlog both the max and overall were about the same with an average of **0.803** for no-deletion and **0.804** with deletion. However, while it could be argued that the differences are not great, it does show that there is some merit to selecting the right policy for the task. Migration with deletion further enhances diversity by deleting random individuals at the risk of destroying sub-solutions. Further, Wine was the data sets to best exploit this extra diversity property without being too punished for it, possibly because of good sub-solutions not being hard to find for an easy data set. Subsequently, a harder data set such as Diabetes might be more heavily affected as good solutions are harder to find and ABs should preferably not be deleted. Finally, Heart Statlog seem to not be as positively affected by migration by either policies, indicating that it may be better off without it, further showing that the optimal policy is very problem dependent.

The hypothesis of table 4.7 indicated that migration without deletion would perform better because deleting ABs was assumed to have negative effects. However, as the tests show what policy perform better is dependent on the properties of the data set, where some data sets are able to benefit from deletion of random ABs.

Chapter 5

Conclusions and Future Work

The following chapter presents an overall discussion and goal evaluation in section 5.1, followed by the contributions in section 5.2. Finally, future work is presented in section 5.3.

5.1 Discussion and Goal Evaluation

The overall objective or goal for this thesis was to *investigate how artificial immune systems could be combined with the island model to create an accurate and efficient novel hybrid classification algorithm.* The goal was subsequently divided into three research questions, as will be discussed below:

Research question 1 *How can the island model combined with artificial immune systems be used to increase the accuracy of the classification?*

In terms of accuracy, MAIM has showed clear indications that the island model positively effect the developed AIS. Consequently, results show that higher classification accuracies are obtained by combining IGA and AIS compared just the AIS model by itself.

The intention was to exploit properties of IGA that help convergence and diversity through sub-populations, migration and genetic isolation. Results conclude that these properties are in fact positive for the model, as several migrating smaller populations together have a stronger generalisation ability than one large population. Further, the level of achieved benefit does vary, as AIS only was able to perform better than a few island setups on some data sets. However, with some tuning of island numbers, MAIM was shown to always perform better than its single AIS counterpart, with 4 islands overall achieving the best results with local AB_interaction, providing a reasonable compromise between genetic isolation and interconnection.

The use of global and local information was tested and compared in terms of accuracies. Further, while one would assume that a more directed search through global information would be beneficial, this was not shown to be the case. The combined generalisation ability of islands evolving mostly separately over the whole space was shown to be better than islands specialised on different parts. Subsequently, some level of independence between islands is assumed to be beneficial in improving accuracies.

MAIM was able to perform on par with other established algorithms, both on separable and non-separable data sets of smaller feature spaces. Particularly on some of the more difficult data sets MAIM showed enhanced exploration leading to better results than other established AIS algorithms, with and without feature selection schemes. However, on data sets with many classes or large feature spaces MAIM was shown to struggle as a result of insufficient exploration in large spaces and static AB classes not being efficient for many classes. However, the worsening performance was shown to be avoided by employing a smarter AB RR initialisation and increasing the population, simultaneously achieving higher accuracies and making the results more stable.

Research question 2 *How can the island model combined with artificial immune systems be used to increase the efficiency of the classification process?*

In terms of efficiency, MAIM has shown that the combination of an AIS and IGA is generally more efficient than AIS only, because of its distributed properties. Furthermore, because of local information causing each island to cover larger parts of the search space, it is more efficient than its global information counterpart where IPs are more condensed in smaller areas resulting in higher levels of AB congestion.

While MAIM is generally more efficient than it's single AIS counterpart, there are cases where efficiency worsens rather than improves. This is the case for very small populations distributed over many islands, where the overhead of migration becomes proportionally high in comparison to the rest of the computation. Simultaneously, AB congestion is not a problem for smaller populations resulting in the properties of IGA negatively affecting efficiency for such setups.

In most of the cases tested a large enough population was employed where MAIM's efficiency properties was shown to be positive. Further, MAIM has an increasingly positive effect as populations increase in conjunction with islands, as the increasing level on distribution provided by employing more islands is able to keep runtimes linearly increasing longer. Subsequently, this counteracts the inevitable exponential increases in runtimes as islands become congested, making the proposed algorithm very efficient for larger populations. Furthermore, while optimal population sizes and island numbers vary between data sets, several tests indicate that higher populations and higher island numbers are beneficial for enhanced exploration. Subsequently, there is a point to using larger populations and island numbers and further exploit MAIM's properties of enhanced efficiency.

Research question 3 *How does the migration configuration used impact the accuracy results of the algorithm?*

In terms of migration, it was shown to overall have positive impact on accuracies. However, its level of impact is dependent on the properties of each data set, with one data set tested preferring no migration at all and subsequently total genetic isolation. Further, local information was shown to benefit more from migration than global information, while also having less variance in optimal parameters for each set. Using lower values of either MF or MR while the other being higher generally achieved good results. However, too high values of both worsened the results. Consequently, while migration should not simultaneously be performed often and for large parts of the IPs, one at the time may be beneficial. Additionally, it was concluded that the optimal migration policy is also something that is dependent on the data set being solved. Furthermore, the higher diversity achieved from deletion migration was more easily exploitable by the easily separable data set Wine. Diabetes, being a substantially harder data set, was likely punished to a greater degree by having important ABs deleted, subsequently favouring the use of migration without AB deletion.

Through the results achieved it can be concluded, when the AB population is set to the number of AGs and a small and easily separable dataset (e.g. Iris, Wine) is used, the algorithm might compromise between accuracy and efficiency. However, if the number of islands are kept low the decrease in efficiency is almost negligible while accuracy is still improved. However, if larger populations are used for these data sets MAIM clearly improves efficiency also here. Further, for larger and more complex datasets (e.g. Diabetes, Sonar) it becomes apparent that no compromise is necessary and both enhancements can be achieved concurrently even when the population was no larger than the number of AGs. Therefore, the proposed algorithm can be said to achieve the thesis' goal of simultaneously being both more accurate and more efficient than its non-distributed counterpart.

The accuracy tests have shown that the AIS implemented for the MAIM have some limitations. This becomes evident when comparing the results of AIS only with other established AIS algorithms (see section 4.5.1). However, the AIS model is only tested with one set of parameters, none which are optimised. Therefore, it is likely that some increase in accuracy could be expected by fine-tuning these parameters. However, it is uncertain whether the accuracy of the MAIM would further improve if the accuracy of AIS only had been better. Consequently, there is a chance that the addition of an IGA technique simply makes up for the shortcomings of searching in the implemented AIS model, and will not improve an AIS algorithm already proficient at searching the whole feature space by itself. Additionally, large population sizes and iterations are required to achieve good results for some datasets where other algorithms perform well with smaller parameter values. This indicates that their exploratory ability is not as dependent on these parameters and sufficient exploration is achieved regardless. Fortunately, increasing the number of islands counteracts much of the negative effects of increasing the population size, making large populations feasible.

Overall, the MAIM algorithm shows promising results for classification, performing on par with many state of the art algorithms while providing a solution for exponentially increasing runtimes. Subsequently, it can be said that the proposed algorithm sufficiently exploits the inherently distributed nature of the IS, arguably moving AIS algorithms further towards a more defined niche.

5.2 Contributions

The contribution of this work is the successful combination of the AIS and IGA techniques, resulting in the proposed algorithm, MAIM, for supervised classification. Through this work, several discoveries have been made.

The proposed hybrid algorithm generally achieves improved accuracy and efficiency compared to the AIS alone and thus shows that such a hybrid does provide the advantages sought and AIS is able to effectively exploit the properties of IGA. The enhanced exploration and convergence provided by the combination of IGA and AIS is therefore positive.

A novel topology of combining a directed ring model with a master slave approach is presented. The results show promising results for such a topology as the algorithm is able to exploit the properties of both. Enhanced genetic isolation and diversity is achieved through the ring model and solution recombination and search control through the master-slave approach.

Varying degrees of genetic isolation provided by the number of islands employed, has been shown to be favourable for AIS classification, depending on the dataset used. However, it is clear that separately evolving and occasionally combining chromosomes from different sub-populations enhances the search process over time.

Independence between islands is shown to be important. When global AB_interaction is employed (less independent island evolution) the results are worse than when only local AB_interaction is used, both in terms of accuracy and efficiency. Through this it was concluded that a combination of sub-populations evolving in almost complete isolation is better at generalising than a combination sub-populations evolving dependent on each other, at least in the case of global and local AB_interaction.

Migration is shown to positively affect the accuracies achieved. However, the optimal migration parameters and policy varies depending on the properties of each dataset, but generally migration has a positive impact. This shows that while some level of independence is important, the sub-populations still benefit from exchanging genetic material.

5.3 Future Work

The following section will present potential extensions to this work which could be used for further research. Further, figure 5.1 displays a brief overview of the project's future work and process.

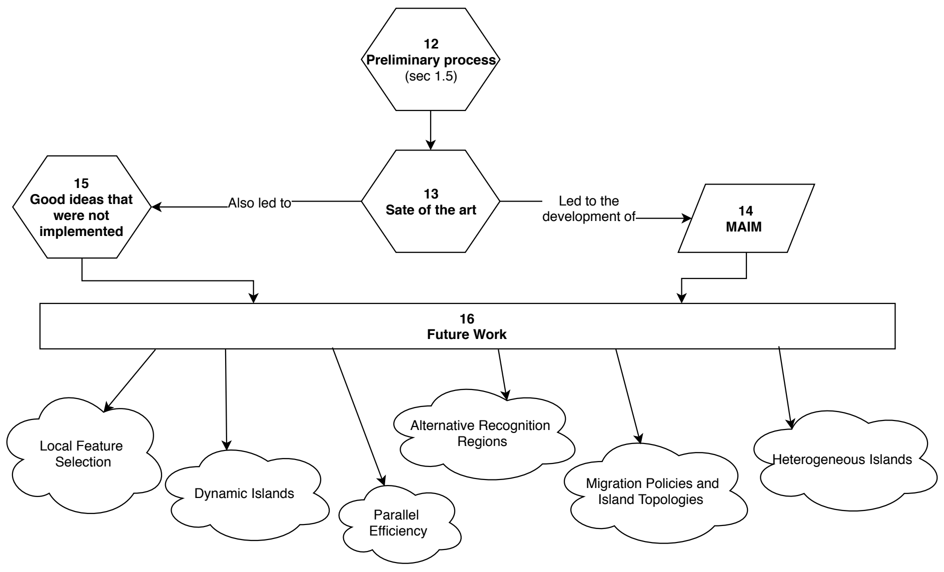


Figure 5.1: Extended process map for future work.

5.3.1 Integrated Feature Selection

A feature which is not included in the proposed algorithm, but is generally an important component of machine learning algorithms is the feature selection process. The AISLFS algorithm has successfully integrated a feature selection process into an AIS (see section 2.2.2) and achieved enhanced results because of it. Further, the results indicate that feature selection is important for achieving the best results on difficult sets like Glass and Sonar [8]. Consequently, it was originally intended for MAIM to employ a LFS approach, similar to the one in AISLFS. However, when attempting to integrate LFS into MAIM it became clear that it would severely affect the efficiency of the algorithm. Feature subset selection methods, which LFS is a part of, are NP-hard problems that with a complexity of $O(2^m)$ where m is the total number of features [34]. This makes exhaustive feature subset searches impractical for a large m. Further, LFS is a subset selection method that attempts to find subsets for n different parts of the search space which increases the complexity even further to $O(n2^m)$. When implementing LFS into MAIM, it was realised that LFS had detrimental effects on the efficiency of the algorithm. Because of this, it was concluded that LFS should not be implemented into the base algorithm in this thesis, but rather suggested as future work where it could possibly be implemented in a more efficient way.

For LFS to not degrade the efficiency of the algorithm it needs to be implemented in such a way that it exploits the inherent distributed and parallel nature of the model. Therefore, future work should be conducted on a *feature selection island* running asynchronously in parallel to the rest of the algorithm. Such an extension could be as presented in figure 5.2 where an additional third layer is introduced, running in parallel to the other islands, in order to handle the feature selection process. Subsequently, its is possible to achieve the same performance as in the original MAIM, with the addition of occasional feature subset input from the feature selection island. In such an approach MAIM would have to run a separate algorithm for finding different feature subsets. The feature selection island could be ran with an AIS approach or with a completely different optimisation method, as it its task is only to communicate what feature subsets are preferred for what parts of the search space.

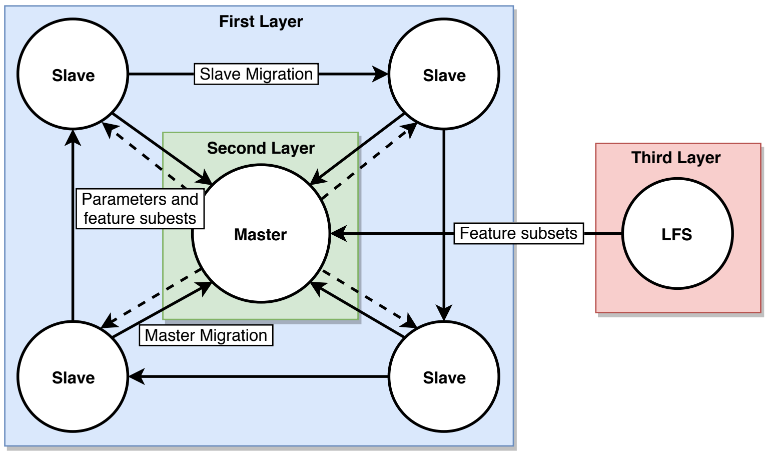


Figure 5.2: Proposed model extension incorporating feature selection.

5.3.2 Recognition Region Shapes

Alternative recognition regions, here meaning regions with geometric shapes that differ from a hypersphere, is something that have shown promising results under certain conditions (see section 2.2.2). Due to the efficiency and simplicity benefits of hyperspheres, alternative recognition regions were not implemented into MAIM for this work, but it is something that may enhance the classification process, due to the algorithm's distributed nature. More specifically, it would be interesting to see if each island could be further specialised by employing and evolving ABs with different recognition region shapes. Future work should therefore be conducted to see if a combination of different AB shapes could potentially be used to more accurately classify difficult and non-separable regions of a dataset, especially in high AG density regions of many classes where hyperspheres makes separation difficult (see **TL1.6** in section 4.4.1).

Future work should see what properties different shapes have for AIS classification specifically, both in terms of convergence and accuracy. However, it should be noted that that while hyperspheres have their own problems in higher dimensions (see section 2.2.2) so does other shapes. For instance, for hypercubes, it would require 2^n points to define a hypercube in n dimensions, as opposed to to a hypersphere that always only require an RR radius regardless of n. Therefore alternative RRs might not be very practical with datasets with large feature spaces. However, if local feature selection is employed (see section 5.3.1) this problem could be handled as in the work of AISLFS, where hypercubes are only available to ABs with a certain dimensionality. Subsequently, future work on alternative recognition regions should ideally be conducted together with the LFS technique.

5.3.3 Dynamic Islands

Both the results from the accuracy tests (see section 4.5.1) and the efficiency tests (see section 4.5.2) have shown that the number of islands and populations sizes have an impact on the performance of the proposed algorithm. The accuracy tests shows that different data sets benefit from the different number of islands. Additionally, datasets like Glass, Sonar and Ionosphere achieve higher accuracies as the population sizes increase over #AGs.

Future work should be conducted on dynamic islands in order to optimise both efficiency and accuracy of the proposed algorithm. This can be implemented in multiple ways. One approach would be to keep the same total population size as the number of islands dynamically change and redistribute the population over a new set of islands. This way, accuracy and efficiency could be incrementally optimised during the course of the algorithm in relation to island numbers.

Another approach that should be investigated is to increase the population size as new islands are initialised, similar to the work of Huang [17]. This approach would likely not optimise much in terms of efficiency, IP populations would remain the same. However, it could potentially help finding the optimal popualtion size in relation to accuracy. Results indicate that accuracies increase and results stabilise as populations increase up to a certain point (see test **TL1.3** in section 4.4.1), indicating that a dynamic island approach could be used to tune population sizes.

5.3.4 Migration Policies and Island Topologies

The migration policies defined for the salve islands in this work, as discussed in section 3.1.3, are based on random selection of migrants. Further, results of section 4.5.3 indicate that the different policies are more easily exploitable by different data sets. Subsequently, other policies, like the what is discussed in section 2.2.1, should also be implemented to evaluate their properties and effect on the result. The most common approach to migration is an elitist selection scheme, indicating that future work should be conducted to evaluate the effect of migrating sets of the fittest individuals.

The chosen topology for MAIM in this work is a directed ring topology. However, as discussed in section 2.2.1 several other topologies can be implemented. The directed ring topology is, of the ones discussed, probably the one were the islands are the most isolated from each other. This is favourable in terms of keeping diversity and avoiding premature convergence, at the cost of possible slow convergence. Therefore, future work should study the effects of different topologies, such as the bidirectional or fully connected topology, in terms of accuracy, efficiency and convergence.

5.3.5 Heterogeneous Islands

When comparing different AIS algorithms in section 4.5.1, it is clear that the algorithms possess different strengths and weaknesses in terms of performance on the data sets. As MAIM employs multiple AISs, future work should be conducted on implementing truly heterogeneous islands through completely different AIS algorithms on the slaves. This way, much like ensemble methods (see section 2.1.3) the different AISs could be proficient at different types of AGs and data sets. Consequently, a better generalisation ability could be achieved through having the different AISs vote on classifications, much like in the original MAIM. However, it is important to note that the ABs on the different islands in such an algorithm would need to use the same, or at least sufficiently similar, structure and data types. ABs, which are shared between the slaves, needs to be usable on all the different AIS algorithms employed. The model extension proposed in section 5.3.1 is an example of such an algorithm, where one island (the feature selection island) employ a different algorithm than the slaves.

5.3.6 Parallel Efficiency

The proposed algorithm employs a distributed architecture and therefore also possess an inherent parallel nature that allows islands to iterate independently of each other. Subsequently, future work should be conducted on using these properties for creating a parallel version of the MAIM algorithm. Further, when only local AB_interaction is employed islands and migration may run asynchronously as no islands would be required to wait for communication from the master. While such an implementation may not achieve better accuracies it could potentially achieve promising levels of parallel efficiency.

Bibliography

[1] U. Aickelin and D. Dasgupta. *Artificial Immune Systems*, pages 375–399. Springer US, Boston, MA, 2005.

[2] E. Alpaydin. *Introduction to machine learning.* MIT press, 2009.

[3] L. Araujo, J. J. Merelo, A. Mora, and C. Cotta. Genotypic differences and migration policies in an island model. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, GECCO '09, pages 1331–1338, New York, NY, USA, 2009. ACM.

[4] E. Cantú-Paz. Migration policies, selection pressure, and parallel evolutionary algorithms. *Journal of Heuristics*, 7(4):311–334, Jul 2001.

[5] A. L. Corcoran and R. L. Wainwright. A parallel island model genetic algorithm for the multiprocessor scheduling problem. In *Proceedings of the 1994 ACM Symposium on Applied Computing*, SAC '94, pages 483–487, New York, NY, USA, 1994. ACM.

[6] L. N. de Castro and F. J. V. Zuben. Learning and optimization using the clonal selection principle. *IEEE Transactions on Evolutionary Computation*, 6(3):239–251, June 2002.

[7] D. Dua and C. Graff. UCI machine learning repository, 2017.

[8] G. Dudek. An artificial immune system for classification with local feature selection. *IEEE Transactions on Evolutionary Computation*, 16(6):847–860, Dec 2012.

[9] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing.* Springer Publishing Company, Incorporated, 2nd edition, 2015.

[10] D. Floreano and C. Mattiussi. *Bio-inspired artificial intelligence: theories, methods, and technologies.* MIT press, 2008.

[11] Y. Gong and A. Fukunaga. Distributed island-model genetic algorithms using heterogeneous parameter settings. In *2011 IEEE Congress of Evolutionary Computation (CEC)*, pages 820–827, June 2011.

[12] Y.-J. Gong, W.-N. Chen, Z.-H. Zhan, J. Zhang, Y. Li, Q. Zhang, and J.-J. Li. Distributed evolutionary algorithms and their models: A survey of the state-of-the-art. *Applied Soft Computing*, 34:286 – 300, 2015.

[13] A. A. Gozali and S. Fujimura. *Localization Strategy for Island Model Genetic Algorithm to Preserve Population Diversity*, pages 149–161. Springer International Publishing, Cham, 2018.

[14] E. Hart. Not all balls are round: An investigation of alternative recognition-region shapes. In C. Jacob, M. L. Pilat, P. J. Bentley, and J. I. Timmis, editors, *Artificial Immune Systems*, pages 29–42, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[15] E. Hart and J. Timmis. Application areas of ais: The past, the present and the future. *Applied Soft Computing*, 8(1):191 – 201, 2008.

[16] H. Homayounfar, S. Areibi, and F. Wang. An advanced island based ga for optimization problems. In *Proceedings of the international DCDIS conference on engineering applications and computations*, pages 46–51. Citeseer, 2003.

[17] H. S. Huang. Distributed genetic algorithm for optimization of wind farm annual profits. In *2007 International Conference on Intelligent Systems Applications to Power Systems*, pages 1–6, Nov 2007.

[18] C. Janeway. *Immunobiology Five*. Immunobiology 5 : the Immune System in Health and Disease. Garland Pub., 2001.

[19] P. Karpov, G. Squillero, and A. Tonda. Valis: an evolutionary classification algorithm. *Genetic Programming and Evolvable Machines*, 19(3):453–471, Sep 2018.

[20] R. Kicinger, T. Arciszewski, and K. De Jong. Distributed evolutionary design: Island-model based optimization of steel skeleton structures in tall buildings. 11 2004.

[21] A. Leitão, F. B. Pereira, and P. Machado. Island models for cluster geometry optimization: how design options impact effectiveness and diversity. *Journal of Global Optimization*, 63(4):677–707, Dec 2015.

[22] R. Maclin and D. W. Opitz. Popular ensemble methods: An empirical study. *CoRR*, abs/1106.0257, 2011.

[23] C. McEwan and E. Hart. Representation in the (artificial) immune system. *Journal of Mathematical Modelling and Algorithms*, 8(2):125–149, Jun 2009.

[24] L. Meng, P. van der Putten, and H. Wang. A comprehensive benchmark of the artificial immune recognition system (airs). In X. Li, S. Wang, and Z. Y. Dong, editors, *Advanced Data Mining and Applications*, pages 575–582, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[25] Q. Meng, J. Wu, J. Ellis, and P. J. Kennedy. Dynamic island model based on spectral clustering in genetic algorithm. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 1724–1731, May 2017.

[26] T. M. Mitchell et al. Machine learning. wcb, 1997.

[27] N. Neves, A. . Nguyen, and E. L. Torres. A study of a non-linear optimization problem using a distributed genetic algorithm. In *Proceedings of the 1996 ICPP Workshop on Challenges for Parallel Processing*, volume 2, pages 29–36 vol.2, Aug 1996.

[28] S. Ozsen and C. Yucelbas. On the evolution of ellipsoidal recognition regions in artificial immune systems. *Applied Soft Computing*, 31:210 – 222, 2015.

[29] B. J. Park, H. R. Choi, and H. S. Kim. A hybrid genetic algorithm for the job shop scheduling problems. *Computers & Industrial Engineering*, 45(4):597 – 613, 2003.

[30] M. M. Rahman, D. Śl̨ezak, and J. Wróblewski. Parallel island model for attribute reduction. In S. K. Pal, S. Bandyopadhyay, and S. Biswas, editors, *Pattern Recognition and Machine Intelligence*, pages 714–719, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[31] I. Roitt. *Essential Immunology.* Blackwell Scientific Publications, 1994.

[32] A. Sharma and D. Sharma. Clonal selection algorithm for classification. In P. Liò, G. Nicosia, and T. Stibor, editors, *Artificial Immune Systems*, pages 361–370, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[33] C. Simões, R. Neves, and N. Horta. Using sentiment from twitter optimized by genetic algorithms to predict the stock market. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, pages 1303–1310, June 2017.

[34] J. Tang, S. Alelyani, and H. Liu. Feature selection for classification: A review. In *Data Classification: Algorithms and Applications*, 2014.

[35] J. Timmis, T. Knight, L. N. de Castro, and E. Hart. *An Overview of Artificial Immune Systems*, pages 51–91. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[36] A. Watkins. Exploiting immunological metaphors in the development of serial, parallel, and distributed learning algorithms. 01 2005.

[37] A. Watkins and J. Timmis. Exploiting parallelism inherent in airs, an artificial immune classifier. In G. Nicosia, V. Cutello, P. J. Bentley, and J. Timmis, editors, *Artificial Immune Systems*, pages 427–438, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[38] A. Watkins, J. Timmis, and L. Boggess. Artificial immune recognition system (airs): An immune-inspired supervised learning algorithm. *Genetic Programming and Evolvable Machines*, 5(3):291–317, Sep 2004.

[39] A. B. Watkins and L. C. Boggess. A resource limited artificial immune classifier. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*, volume 1, pages 926–931 vol.1, May 2002.

[40] D. Whitley, S. Rana, and R. Heckendorn. The island model genetic algorithm: On separability, population size and convergence. *Journal of Computing and Information Technology*, 7, 12 1998.

[41] S. Wright. The roles of mutation, inbreeding, crossbreeding and selection in evolution. *Proceedings of the Sixth International Congress of Genetics*, 1(1):356–366, 1931.

Appendices

.1 Results Accuracy Tests

Additional accuracies are here presented with SDs in parenthesis.

.1.1 Tests Conducted with New Initial Radius

Table 1 presents the results obtained from the tests conducted with a new radius initialisation.

Data sets — Islands	**1**	**3**	**4**	**5**	**6**	**7**	**8**
Iris	0.9613 (0.0056)	0.9600 (0.0094)	**0.9657 (0.037)**	0.9533 (0.0067)	0.9547 (0.0056)	0.9533 (0.0115)	0.9573 (0.0060)
Wine	0.9663 (0.0045)	0.9597 (0.0026)	0.9651 (0.0048)	0.9698 (0.0059)	0.9637 (0.0053)	0.9712 (0.0024)	0.9703 (0.0052)
Diabetes	0.7304 (0.7304)	0.7429 (0.0043)	0.7509 (0.0046)	0.7509 (0.0032)	0.7506 (0.0072)	0.7525 (0.0016)	0.7500 (0.0032)
Glass	0.6301 (0.0215)	0.6094 (0.0134)	0.5917 (0.0076)	0.5750 (0.0098)	0.5933 (0.0152)	0.5974 (0.0103)	0.6007 (0.010)
Sonar	**0.7364 (0.0066)**	0.7341 (0.0154)	0.7107 (0.071)	0.6992 (0.0232)	0.7078 (0.0158)	0.7042 (0.0195)	0.6980 (0.0168)
Ionsphere	0.8610 (0.0117)	0.8713 (0.0210)	0.8074 (0.0189)	**0.8741 (0.0159)**	0.7903 (0.0163)	0.8479 (0.0112)	0.7768 (0.053)
Heart statlong	0.7919 (0.0134)	0.8163 (0.0042)	0.8266 (0.0084)	0.8163 (0.0096)	0.8237 (0.0056)	0.8178 (0.0195)	0.8266 (0.0092)
Breast Cancer Wisconsin	0.9625 (0.0012)	0.9660 (0.0012)	0.9653 (0.0021)	0.9665 (0.0023)	0.9651 (0.0035)	**0.9686 (0.0015)**	0.9645 (0.0006)

Table 1: Tests on all data sets with RRs set to a random AG of the AB's class (up to eight islands).

Data set / Islands	**9**	**10**	**11**	**12**
Iris	0.9520 (0.0087)	0.9547 (0.0073)	0.9547 (0.0073)	0.9587 (0.0056)
Wine	**0.9755 (0.0028)**	0.9715 (0.0046)	0.9722 (0.0040)	0.9730 (0.0050)
Diabetes	0.7516 (0.0032)	0.7500 (0.0025)	**0.7581 (0.0087)**	0.7462 (0.0031)
Glass	0.6146 (0.0098)	0.6306 (0.0223)	**0.6419 (0.0190)**	0.6224 (0.0142)
Sonar	0.6971 (0.0171)	0.6841 (0.0039)	0.6842 (0.0062)	0.6644 (0.0051)
Ionsphere	0.7930 (0.0054)	0.7818 (0.0081)	0.7824 (0.0112)	0.7737 (0.0048)
Heart statlong	0.8074 (0.0037)	**0.8303 (0.0071)**	0.8266 (0.0061)	0.8281 (0.0042)
Breast	0.9663 (0.0049)	0.9651 (0.0055)	0.9634 (0.0050)	0.9620 (0.0008)

Table 2: Tests on all data sets with RRs set to a random AG of the AB's class (from nine to twelve islands).

Population Size / Iterations	**500**	**1000**	**1500**	**2000**
500	0.733 (0.014)	0.746 (0.014)	0.746 (0.015)	0.749 (0.017)
1000	0.737 (0.006)	0.780 (0.010)	0.785 (0.009)	0.800 (0.016)
1500	0.768 (0.006)	0.796 (0.012)	0.800(0.011)	0.820 (0.006)
2000	0.789 (0.07)	0.823 (0.018)	0.803 (0.010)	**0.827 (0.012)**

Table 3: Accuracies on the Sonar data set on four islands with iterations and population sizes spanning 500 to 2000.

Population Size / Iterations	**500**	**1000**	**1500**	**2000**
500	0.868 (0.008)	0.896 (0.003)	0.899 (0.007)	0.908 (0.007)
1000	0.892 (0.011)	0.903 (0.009)	0.900 (0.011)	0.911 (0.006)
1500	0.876 (0.014)	0.898 (0.009)	0.902 (0.012)	0.899 (0.007)
2000	0.860 (0.011)	0.891 (0.010)	0.908 (0.006)	**0.918 (0.007)**

Table 4: Accuracies on the Ionosphere data set on five islands with iterations and population sizes spanning 500 to 2000.

.1.2 Tests Conducted with Class Distribution

Table 5 presents the results obtained from the tests conducted with a class distributions.

	1	**3**	**4**	**5**	**6**	**7**	**8**
Iris	0.9560 (0.0037)	**0.9653 (0.0056)**	0.9627 (0.0076)	0.9600 (0)	0.9613 (0.0073)	0.9640 (0.0037)	0.9640 (0.0089)
Wine	0.9524 (0.0030)	0.9600 (0.9600)	0.9540 (0.0073)	0.9548 (0.0012)	0.9563 (0.0075)	**0.9654 (0.0079)**	0.9628 (0.0105)
Diabetes	0.7189 (0.0060)	0.7466 (0.0067)	0.7395 (0.0017)	0.7440 (0.0060)	**0.7494 (0.0046)**	0.7449 (0.0012)	0.7473 (0.0050)
Glass	0.**6728 (0.0078)**	0.6660 (0.0073)	0.6572 (0.0160)	0.6606 (0.0091)	0.6652 (0.0185)	0.6582 (0.0048)	0.6335 (0.0074)
Sonar	**0.7602 (0.0248)**	0.7166 (0.0108)	0.7043 (0.0326)	0.7110 (0.0167)	0.6876 (0.0100)	0.6794 (0.0277)	0.6801 (0.0098)
Ionsphere	**0.8336 (0.0083)**	0.7801 (0.0076)	0.7198 (0.0032)	0.7345 (0.0033)	0.7073 (0.0086)	0.7033 (0.0078)	0.6885 (0.0040)
Heart statlong	0.7748 (0.0151)	0.8052 (0.0132)	0.8096 (0.0056)	0.8141 (0.0061)	0.8074 (0.0069)	**0.8192 (0.0041)**	0.8155 (0.0112)
Breast Cancer Wisconsin	0.9631 (0.0027)	0.9654 (0.0021)	0.9649 (0.0032)	0.9663 (0.0022)	0.9677 (0.0026)	**0.9683 (0.0011)**	0.9671 (0.0023)

Table 5: Accuracies using class distributions.

.2 Results Efficiency Tests

The following sections will present more details about the efficiency tests conducted in section 4.5.2. Table 6 to 9 are presenting all results, including the accuracies obtained during the efficiency tests conducted with local AB_interaction. Table 10 and 11 are presenting all results including accuracies, obtained during the efficiency tests conducted with global AB_interaction. All values in the time columns are representing seconds, with standard deviation (also in seconds) in the following parenthesis. The values in the accuracy columns are representing percentage, with standard deviation in the parenthesis. As mentioned in section 4.5.2, all tests are conducted 5 times as ten-fold cross validation and averages are presented.

.2.1 Tests Conducted with local AB_interaction

Data set / Islands	**AIS Only**		**4 Islands**		**8 Islands**		**12 Islands**	
	Time	*Accuracy*	*Time*	*Accuracy*	*Time*	*Accuracy*	*Time*	*Accuracy*
Iris	16.2 (0.25)	95.05 (0.74)	14.64 (0.19)	94.93 (0.76)	15.56 (0.19)	95.73 (0.76)	17.06 (0.24)	95.87 (1.1)
Wine	28.06 (0.13)	95.44 (0.9)	22.76 (0.27)	96.08 (0.87)	23.89 (0.12)	94.91 (1.33)	26.66 (0.34)	95.03 (1.39)
Diabetes	81.29 (0.69)	71.43 (2.19)	79.78 (1.33)	74.09 (0.54)	91.31 (1.17)	74.61 (0.86)	104.99 (0.63)	74.64 (1.3)
Sonar	85.97 (1.61)	76.78 (1.51)	**71.07 (0.55)**	71.88 (3.13)	76.34 (1.30)	67.43.61 (4.84)	83.04 (1.02)	66.94 (1.35)

Table 6: TL2.1 Accuracies on efficiency tests with population 500 and local AB_interaction.

Data set / Islands	**AIS Only**		**4 Islands**		**8 Islands**		**12 Islands**	
	Time	*Accuracy*	*Time*	*Accuracy*	*Time*	*Accuracy*	*Time*	*Accuracy*
Iris	46.5 (0.27)	95.6 (0.89)	31.56 (0.38)	94.3 (0.76)	29.47 (0.27)	95.07 (0.76)	29.99 (0.17)	95.2 (1.1)
Wine	78.51 (0.55)	96.84 (1.51)	48.21 (0.34)	95.42 (0.99)	47.30 (0.34)	95.76 (0.98)	48.26 (0.25)	96.67 (0.82)
Diabetes	193.33 (1.73)	71.16 (1.31)	152.65 (1.43)	75.12 (0.44)	168.24 (2.08)	75.38 (0.61)	179.06 (0.67)	74.96 (0.49)
Sonar	197.47 (4.47)	79.39 (2.76)	**142.60 (1.21)**	76.38 (2.70)	143.81 (0.82)	73.53 (2.17)	149.94 (1.35)	72.33 (2.10)

Table 7: TL2.2 Accuracies on efficiency tests with population 1000 and local AB_interaction.

Data set / Islands	**AIS Only**		**4 Islands**		**8 Islands**		**12 Islands**	
	Time	*Accuracy*	*Time*	*Accuracy*	*Time*	*Accuracy*	*Time*	*Accuracy*
Iris	110.53 (1.28)	95.33 (0.82)	52.9 (0.7)	94.93 (0.76)	48.99 (1.18)	94.80 (0.87)	45.03 (0.32)	95.87 (1.19)
Wine	168.38 (2.81)	96.78 (1.14)	78.48 (0.86)	96.70 (1.13)	77.73 (1.04)	96.39 (1.03)	74.98 (0.88)	95.29 (1.64)
Diabetes	376.67 (6.24)	71.33 (1.44)	231.28 (1.12)	73.20 (0.5)	251.24 (3.42)	75.03 (0.63)	265.23 (1.03)	74.8 (0.7)
Sonar	352.25 (6.54)	7.07 (2.76)	222.62 (1.1)	78.33 (2.8)	**214.42 (2.42)**	75.7 (3.13)	218.24 (2.07)	71.06 (1.74)

Table 8: TL2.3 Accuracies on efficiency tests with population 1500 and local AB_interaction.

Data set / Islands	**AIS Only**		**4 Islands**		**8 Islands**		**12 Islands**	
	Time	*Accuracy*	*Time*	*Accuracy*	*Time*	*Accuracy*	*Time*	*Accuracy*
Iris	215.5 (9.68)	94.67 (1.94)	72.49 (5.38)	95.2 (0.73)	68.12 (1.3)	95.47 (0.87)	64.07 (1.89)	95.20 (0.56)
Wine	301.81 (9.86)	96.21 (0.67)	113.26 (1.9)	97.10 (0.85)	105.7 (1.27)	96.36 (1.19)	101.6 (0.84)	96.52 (0.77)
Diabetes	604.20 (1.59)	70.76 (1.44)	316.25 (1.23)	73.2 (1.38)	334.38 (3.72)	74.52 (0.66)	343.43 (1.4)	74.70 (0.92)
Sonar	556.69 (8.91)	80.74 (1.0)	301.95 (2.52)	79.5 (2.1)	288.02 (1.44)	76.3 (1.3)	**287.18 (5.07)**	74.0 (3.0)

Table 9: TL2.4 Accuracies on efficiency tests with population 2000 and local AB_interaction.

.2.2 Tests Conducted with global AB_interaction

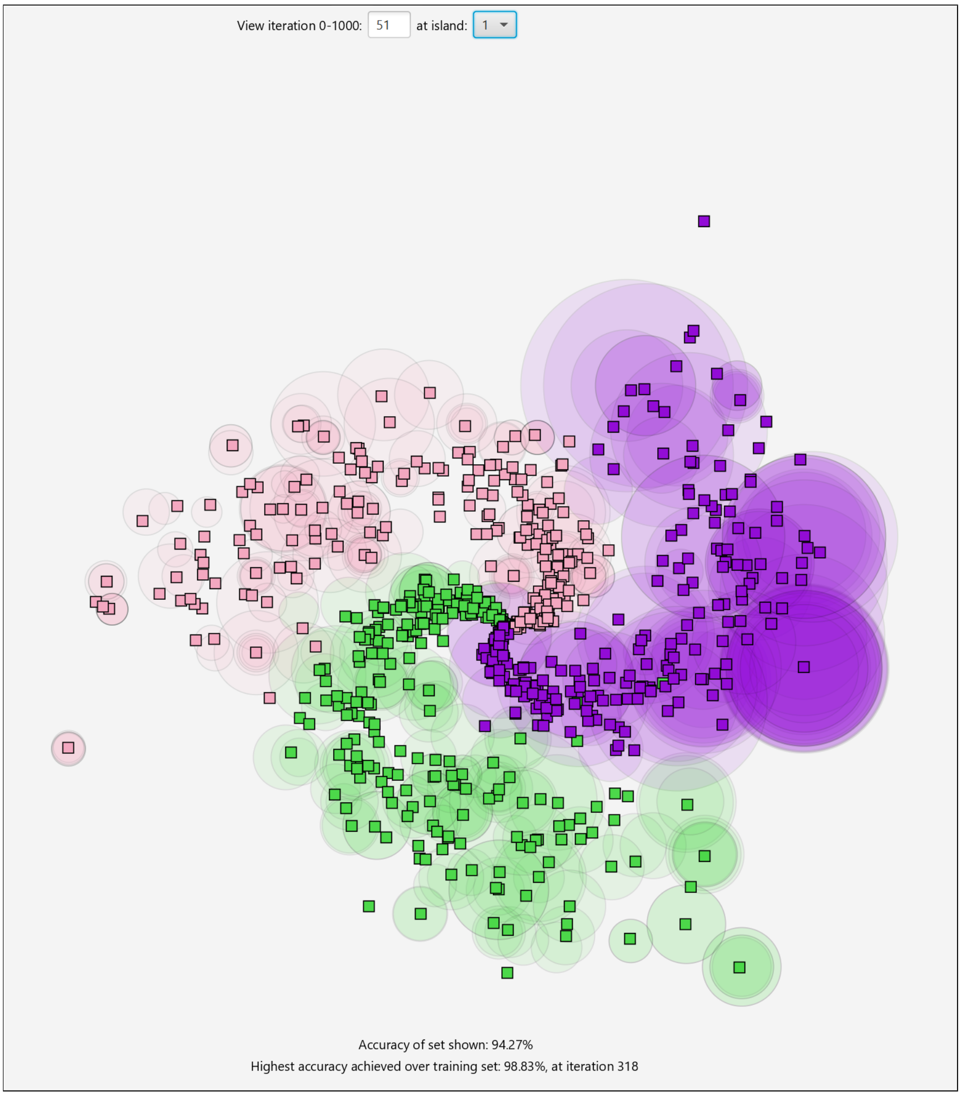
Data set	**AIS Only**		**4 Islands**		**8 Islands**		**12 Islands**	
	Time (seconds)	*Accuracy*	*Time (seconds)*	*Acuaracy*	*Time (seconds)*	*Accuracy*	*Time (seconds)*	*Accuracy*
Iris	**16.2 (0.25)**	95.05 (0.74)	18.54 (0.26)	94.8 (0.26)	19.7 (0.3)	95.40 (1.01)	21.04 (0.22)	94.27 (1.92)
wine	**28.06 (0.13)**	95.44 (0.9)	29.49 (0.34)	95.48 (0.33)	32.08 (0.23)	95.58 (0.45)	34.41 (34)	95.62 (0.76)
Diabetes	**81.29 (0.69)**	71.43 (2.19)	96.20 (0.9)	74.27 (0.59)	106.58 (0.14)	72.93 (0.75)	119.65 (1.47)	73.42 (1.76)

Table 10: TG2.1 Accuracies on efficiency tests with population 500 and global AB_interaction.

Data set	**AIS Only**		**4 Islands**		**8 Islands**		**12 Islands**	
	Time (seconds)	*Accuarcy*	*Time (seconds)*	*Accuarcy*	*Time (seconds)*	*Accuarcy*	*Time (seconds)*	*Accuarcy*
Iris	215.5 (9.68)	94.67 (1.94)	90.48 (1.73)	94.53 (0.73)	87.23 (1.31)	93.86 (1.66)	**83.43 (2.25)**	94.93 (1.74)
wine	301.81 (9.86)	96.21 (0.67)	141.51 (1.19)	96.87 (0.65)	**132.76 (3.37)**	97.06 (0.29)	141.85 (0.44)	96.4 (0.95)
Diabetes	604.20 (1.59)	70.76 (1.44)	**408.04 (6.14)**	72.80 (0.73)	460.15 (8.77)	73.66 (1.25)	482.25 (2.64)	73.76 (1.77)

Table 11: TG2.4 Accuracies on efficiency tests with population 2000 and global AB_interaction.

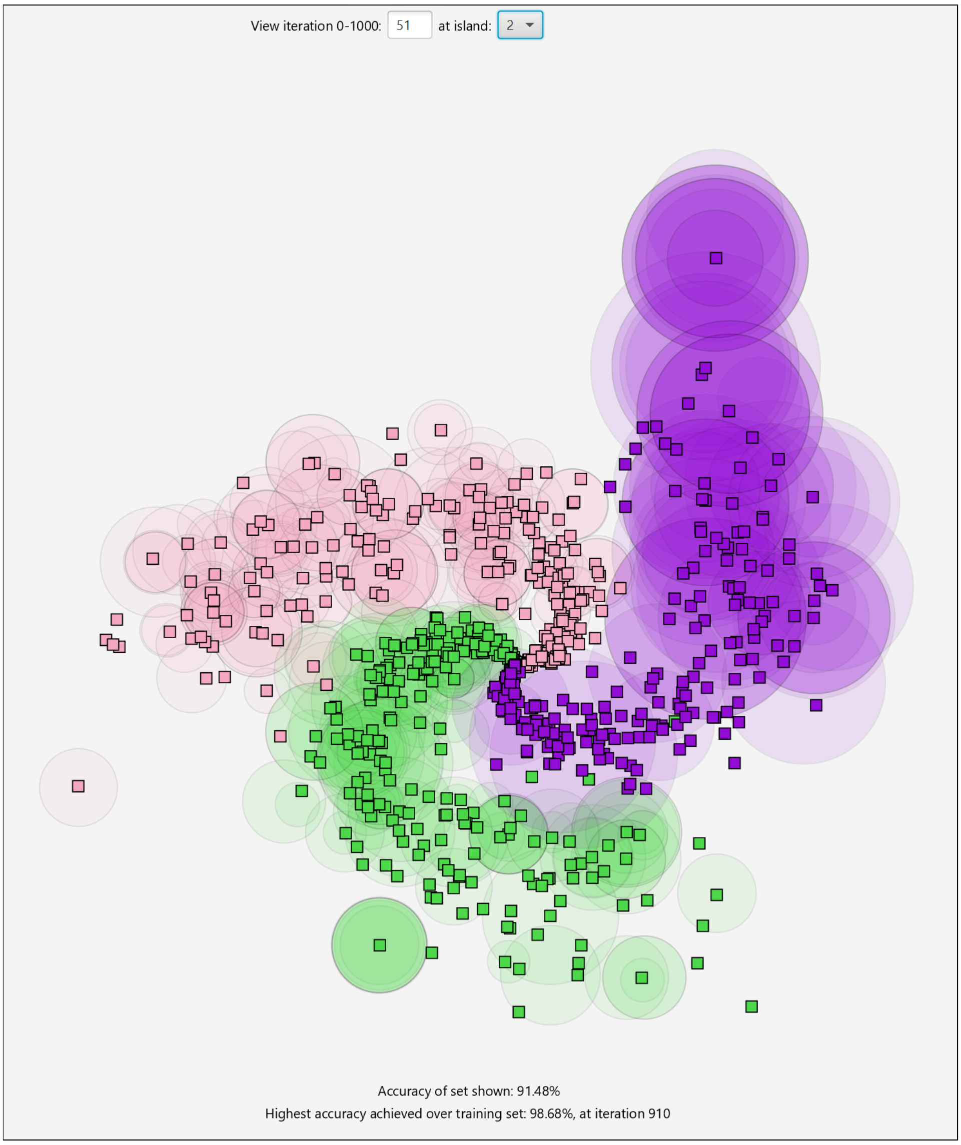
.3 Slave Islands Population Visualisation



Accuracy of set shown: 94.27%

Highest accuracy achieved over training set: 98.83%, at iteration 318

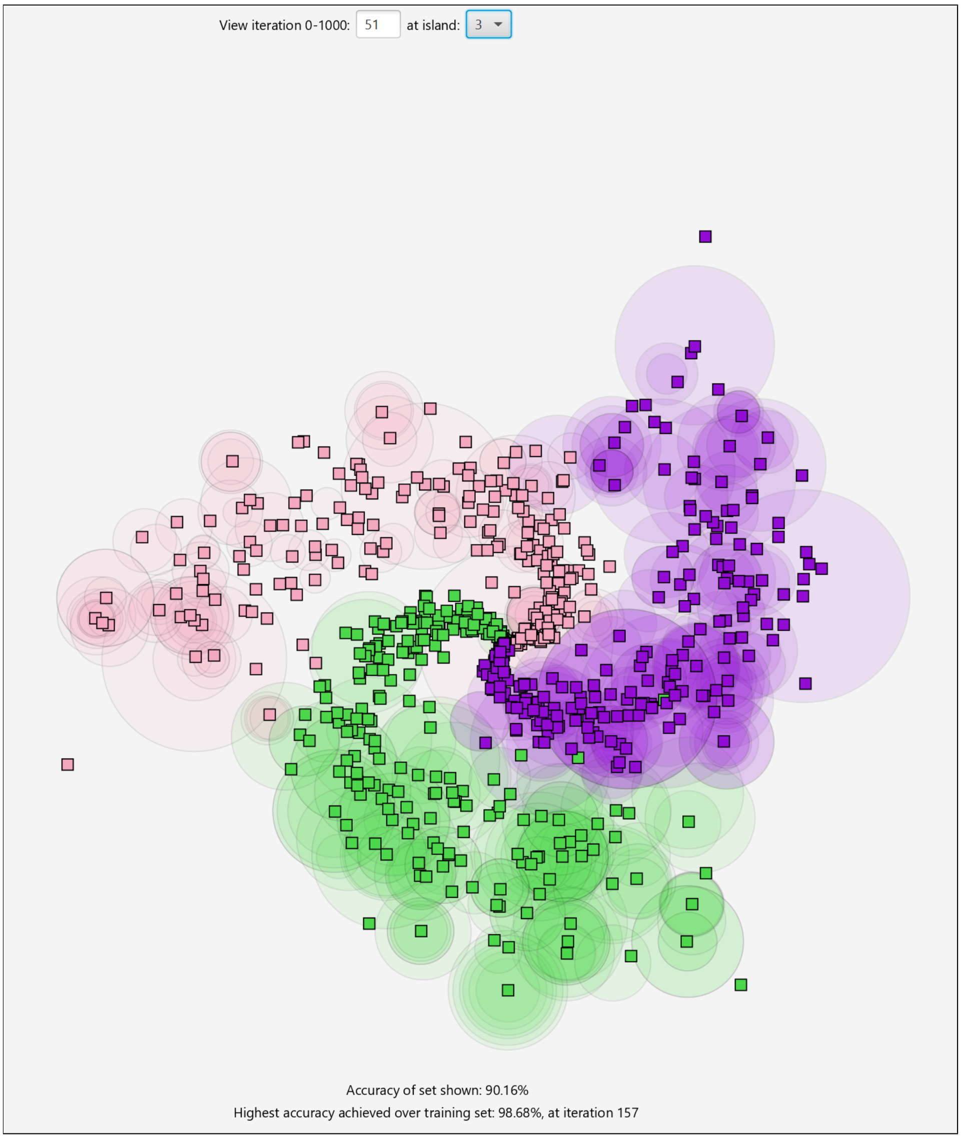
Figure 3: Population of slave island 1.



Accuracy of set shown: 91.48%

Highest accuracy achieved over training set: 98.68%, at iteration 910

Figure 4: Population of slave island 2.



Accuracy of set shown: 90.16%

Highest accuracy achieved over training set: 98.68%, at iteration 157

Figure 5: Population of slave island 3.