A Hybrid Artificial Immune System and Particle Swarm Optimisation Based Algorithm for Mathematical Optimization

Ivan D. Lyubenov

51227356

ivan.lyubenov.12@aberdeen.ac.uk

A dissertation submitted in partial fulfilment

of the requirements for the degree of

Bachelor of Science

of the

University of Aberdeen.

Department of Computing Science

Declaration

No portion of the work contained in this document has been submitted in support of an application for a degree or qualification of this or any other university or other institution of learning. All verbatim extracts have been distinguished by quotation marks, and all sources of information have been specifically acknowledged.

Signed: Ivan D. Lyubenov

Date: 29/04/2016

Abstract

This work examines the work done in the area of Hybrid algorithms and proposes a novel approach to optimization problems. The concept is designed, developed, tested and evaluated on the basis of multiple criteria, against its component algorithms in order to investigate if a hybrid algorithm will improve the overall performance of optimization and under what conditions.

Acknowledgements

I would like to thank all my family and friends for the moral support provided during the years of my degree culminating with this work. I would also like to thank to my supervisors for the support and help throughout this project.

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1 Overview

"*We can see only a short distance ahead but, but we can see plenty ahead that needs to be done.*"[1] - Alan Turing

In evolutionary computing the recent growing area of Hybrid algorithms (Memetic Algorithms) has received a lot of attention from researchers. The reason for that is the fact that the area is concerned with the combination of two or more different algorithms into a single hybrid algorithm which could potentially outperform any of its components without bringing any disadvantages. The aim of a hybrid algorithm is to combine and synergize the strengths of the individual algorithms or mitigate their weaknesses in order to provide advantages such as producing better solution or producing solutions in less time. Any area in which problems are solved by algorithms could benefit from advancement of research on hybrid algorithms. One such area is Multi-modal optimization. Multi-modal optimization has application in areas like mechanics, economics, electrical engineering, molecular modelling and many more. The problems in these areas are incredibly complex and classic methods fail, which is why techniques such as heuristics are used. Particle Swarm Optimisation (PSO) and Artificial Immune Systems (AIS) are meta-heuristic approaches that have considerable success on their own but no sufficient research has been done in combining them.
The aim of this project is to fill the gap in the existing research and propose a novel hybrid algorithm based on Particle Swarm Optimisation and Artificial Immune Systems algorithms.

1.2 Motivation

In computing science mathematical optimization is the process of selection of a best element (with regard to some criteria) out of a set of available alternatives[2]. In the simplest case an optimization problem consists of maximizing or minimising a real mathematical function. The generalization of optimization theory and techniques comprises a sizeable area of applied mathematics. The applications of optimization techniques in the fields of modern physics, biology and economics are very important and every improvement of these techniques propagates into these areas.
The concept of hybrid algorithms is relatively new and warrants more research. There has been one attempt to use combination of PSO and AIS which is restricted to only one type of problems which is not nearly enough to explore the potential in combining the two algorithms. Conducting a research on combining these two algorithms for solving a problem with broad application could not only provide valuable insight on hybrid algorithms but also influence different areas of science.

1.3 Objectives

The aim of this research is to investigate if, combining Particle Swarm Optimization and Artificial Immune Systems to solve optimization problems is possible and if so propose a design that shall be developed and tested against the component algorithms in order to evaluate the benefits of this method and possibly gain valuable insights in the process.

1.4 Primary Goals

- Investigate the research done so far in the field and come up with a design for a hybrid algorithm based on PSO and AIS
- Implement the proposed design into a working tool
- Ensure that the hybrid algorithm is working correctly by continuous testing while developing
- Test the performance of the hybrid algorithm against the performance of the component algorithms
- Evaluate the results of the research
- Gain insight on hybrid algorithms
- Lay foundations for future research in the area.

1.5 Secondary Goals

- Use open source tools and follow well-established standards to ensure compatibility and quality of software
- Develop the software in modular fashion in order to allow improvements and modifications for future research

1.6 Structure

1. Introduction – Introduces the topic, argues the need for conducting and the goals of the research.

2. Background and Related work - Introduces the background of the research, introduces important concepts and inspiration for the approaches used in the area, reviews related research and describes different techniques used and discusses the relevant algorithms and tools.

3. Requirements - Describes the functional and non-functional requirements of the project

4. Methodologies - Discusses the methodologies and plan of the project

5. Design - Discusses potential issues with the design process, discusses main features of PSO and opt-AiNET and how they affect design choices, examines the initial design and the reasons for rejecting it and elaborates on the new, reviewed design and the thought process behind it.

6. Implementation - Discusses the development process of the software

7. Evaluation – Shows the performance tests of the hybrid algorithm based on different criteria and evaluates the results.

8. Conclusion – The report finishes with the summary of the research and discussion of the results then followed by the bibliography and appendices.

2. Background and Related Work

This chapter will introduce the principles of evolution and how they inspire and can be transformed into useful techniques in evolutionary computing. The chapter will also cover how evolutionary algorithms work and relevant research. Towards the end of the chapter, Swarm Intelligence and representing the field algorithms – Particle Swarm Optimization and Opt-AiNet will be discussed and review work done on hybrid algorithms which will give insight and serve as a guide for designing such hybrid algorithm.

2.1 Evolution

Evolution is the change of heritable traits of biological populations over successive generations[3]. Evolutionary mechanisms such as natural selection, mutation and genetic drift are the means through which the process of evolution changes populations over time. In the 19th century, Charles Darwin formulated his famous scientific theory of evolution by natural selection in his book "On the Origin of Species" (1859). Evolution through natural selection is embodied in three principles[4] :

1. Different individuals in a population have different morphologies, physiologies, and behaviours (phenotypic variation).
2. Different phenotypes have different rates of survival and reproduction in different environments (differential fitness).
3. There is a correlation between parents and offspring in the contribution of each to future generations (fitness is heritable).

Based on that we can observe that natural selection preserves traits that are beneficial for the individual and over successive generations the members of a population are replaced by "offspring" that is better suited for survive and therefore reproduce (Figure 1).

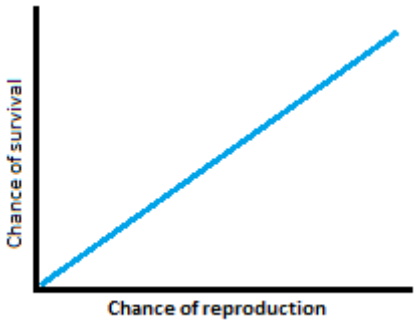


Figure 1 – Principle of Natural Selection

The other two mechanisms – mutation and genetic drift are non-adaptive or also called microevolution and both of them are non-hereditary. In fact genetic drift is caused by the failure of gene being passed on to the next generation as genes are subject to sampling error[5]. When the pressure from natural selection is absent or weak, the genetic drift becomes more prominent and plays an important role in the process of a certain gene disappearing completely or replacing other genes. The size of the population also affects genetic drift as the larger the population the harder it is for a single gene to gain prevalence over others or disappear completely (Figure 2).

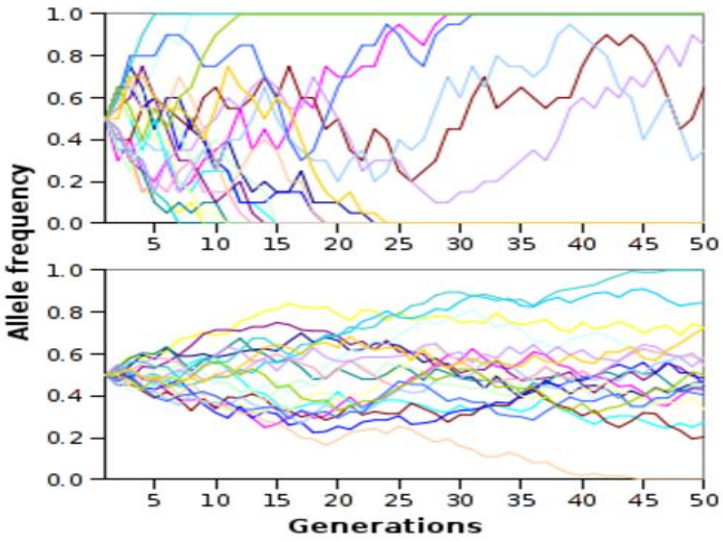


Figure 2 – Genetic Drift

Mutation is the source of variation in evolution, it is the change of DNA sequence of a gene and when introduced it may benefit, damage (loss of function) or have no effect over a gene. In the majority of cases – 70% mutations are harmful and the remaining are neutral or weakly beneficial[6]. In natural selection the loss of function is usually harmful but sometimes if a function is no longer needed and also comes at a certain cost (e.g. pigments are no longer useful to animals living in caves) it can also be seen as beneficial mutation. Similar to genetic drift the size of the population influences the rate at which a loss of function mutation can propagate but in this case it is interesting that if selection pressure is absent or weak it depends more on the mutation rate rather than population size.

2.2 Evolutionary computation

In computing science, evolutionary computation is a subfield of artificial intelligence[7] which originated in the 1950s. Later on different interpretations started to emerge simultaneously. In the US Lawrence J. Fogel[8] came up with "Evolutionary programming". Also in the US John Henry Holland proposed a method called "Genetic algorithm" and in Germany Ingo Rechenberg and Hans-Paul Schwefel called their method evolution strategies. For some years all of these methods developed separately but later around the 1990s they were unified as important sub-fields of evolutionary computation as nature-inspired algorithms started become an influential paradigm in computing.

The area of evolutionary computation is interested in the type of algorithms that adopt Darwinian principles to solve certain problems and from this derive the names evolutionary algorithms and evolutionary computation. The type of problems that the field is concerned with solving is usually one where the derivatives are unknown or also known as "Black box" problems. Usually the algorithms that are able to solve that type of problems are of stochastic and heuristic nature. Stochastic algorithms make use of random variables when there is unknown information or even when there are precise measurements in order to speed up the process. Heuristic are used when there is incomplete or missing information or the set of solutions is too large. The drawback of heuristics is that there is no guarantee that the global best solution can be found but in the most cases it is a trade-off between a good solution and feasibility. This also means that in most of the cases genetic algorithms are computationally expensive since they employ an iterative process through which they grow and develop population, select and guide random search following the aforementioned principles of evolution. As the academic interest and computational capabilities grew in the 1990s evolutionary computation found practical applications and nowadays is more effective in solving multi-dimensional problems than any other approach.

2.3 Evolutionary algorithms

The generality of the principles of natural selection means that any entities in nature that have variation, reproduction, and heritability may evolve. This is the basis for genetic algorithms. When optimising a problem with a evolutionary algorithm a population is constituted of candidate solutions which "evolve" toward better solutions. Typically an individual or candidate solution has a set of properties representing its chromosomes genotype which are altered through the set of genetic operators used in the algorithm (e.g. mutation). After a generation of the population undergoes the process natural selection and then passes on its genes to the next generation that has replaced the old one is typically a better set of solutions to the problem.

2.3.1 Implementation of biological processes into algorithm:

1. *Randomly generate a population of individuals. - (Initialisation)*
2. *Evaluate the fitness of all individuals of the population - (Initialisation)*
3. *Repeat the following processes until – time limit / sufficient fitness reached - (Termination)*
    a. *Select the best individuals to be parents and reproduce – (Selection)*
    b. *Generate offspring through – cloning, crossover, mutation, etc. – (Genetic operators)*
    c. *Evaluate the fitness of all individuals of the population – (Selection)*
    d. *Replace the least fit individuals with new ones – (Selection)*

Initialization:

The initial size of the population depends on the nature of the problem that needs solving or the type of algorithm used. It can vary from few individuals to thousands. Some algorithms use a constant size of population while others allow the population to grow in order reach a solution faster. Usually for the most types of problems it is desired to have a sufficient amount of individuals to cover a wide range of possible solutions and even sometimes individuals can be "planted" in places where it is expected for an optimal solution to be found.

Selection:

With each generation the best individuals of it are selected to be parents, reproduce and pass on their "genes" to the next generation. Individual solutions are selected based on their fitness, measured by a fitness function. There are different selection methods. Typical ones are selecting all the fittest individuals to procreate, select the fittest and part of the unfit ones as it is not uncommon that after the process of mutation an unfit solution shoots right to the top in terms of fitness. The process of assessing the fitness of huge number of individuals is quite computationally expensive, so it also common practice to only assess a part of the population to save time. The fitness function measures how good an individual of the population is depending on the genetic representation used and problem. For example an individual can be represented by its genotype by the means of bits array, where each bit takes value of 0 or 1. In some problems it is hard or impossible to use genotype representation that can be evaluated by the fitness function. In these cases a phenotype representation might be used to evaluate the fitness of the solution. A phenotype might be location (coordinates) of a particle, shape (measuring aerodynamics) or similar characteristics.

Genetic operators:

In order to create the next generation of solutions from the initial or previous generation, means of procreation take form of a combination of techniques such as genetic operators, crossover (recombination) and mutation. As in the real world, for a new individual to be

created, a pair of parent individuals is selected to pass on their genes through the aforementioned techniques which results in a new individual which usually resembles the characteristics of its parents. Although slightly removed from the real world, some research[9, 10] suggests that selecting more than two parents results in higher quality new solutions. Repeating this process to generate a sufficient amount of child solutions will eventually lead to new generation of population. Typically the average fitness of the population will increase as it is common to choose all the best individuals to be parents and only a small number of individuals of lower fitness in order to provide genetic diversity for the next generations. Crossover and mutation are main genetic operators and it is still debated and researched in what cases one is better over the other. Mutation and crossover rates should be tuned according to the problem as too high or too low mutation rates may lead to respectively genetic drift or loss of good solutions and too low or too high crossover rates to slow development or too early convergence of the population. There are also other not so popular genetic operators[11] such as migration, colonization-extinction, regrouping that could be used according to purpose. Example transitional process between consecutive generations depicted (Figure 3).

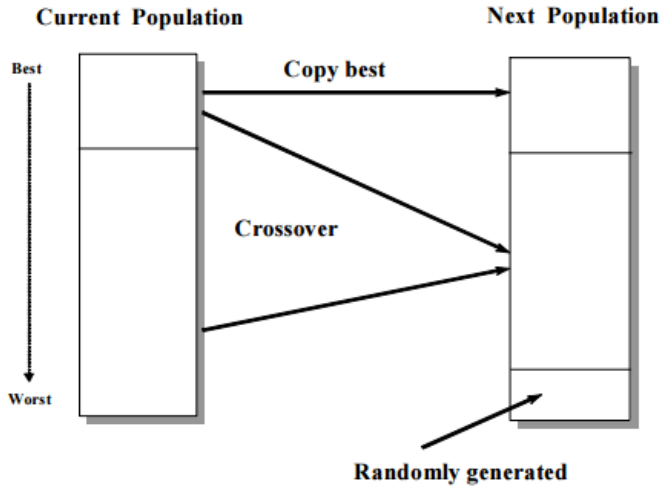


Figure 3 – Crossover Genetic Operator

Termination:

The process of creating new, better generations of solutions continues until a certain criteria have been met. Typical termination criteria are:

- A solution that satisfies minimum criteria is found
- A set number of generations reached
- Allocated budget (computation time/money) reached
- The global best solution's fitness is reaching or has reached a plateau such that successive iterations no longer produce better results
- Manual inspection
- Combinations of the above

2.4 Swarm intelligence

Swarm intelligence another subfield of artificial intelligence. The expression was introduced by Gerardo Beni and Jing Wang in 1989 in their work on cellular robotic systems[12]. Swarm intelligence refers to the collective, decentralized behavior of self-organized systems of natural or artificial character. Swarm intelligence systems are comprised of agents who follow a simple set of rules and interact only locally with other agents and environment. There is no centralized control mechanism and even though there is also a degree of randomness the local interactions lead to the emergence of an intelligent global behavior. The inspirations for such approach comes from systems existing in the natural world such as bird flocking, fish schools, animal herding and bee hives. Example algorithms in the field are:

- Particle swarm optimization
- Artificial immune systems
- Ant colony optimization
- Artificial bee colony algorithm
- Glowworm swarm optimization
- Gravitational search algorithm

Swarm intelligence algorithms find application in swarm robotics and optimization problems. The collection of agents represents population and each agent can be represented by its genotype or phenotype therefore it allows the formulation of algorithms that follow the principles of evolution and natural selection.

2.4.1 Artificial immune systems

"Artificial Immune Systems (AIS)[13] are adaptive systems, inspired by theoretical immunology and observed immune functions, principles and models, which are applied to problem solving"[14]. Artificial immune systems are part of swarm intelligence and are inspired by the immune system in vertebrates. The biological immune system is complex, robust and adaptive system defending an organism from foreign pathogens. These traits are highly desirable for techniques such as genetic algorithms and neural networks. All AIS algorithms mimic the behaviour and characteristics of immunological cells such as B-cells, T-cells and dendritic cells hence comes the names of such algorithms like B-cell algorithm[15] and dendritic cell algorithm[16, 17]. Of course as these cells serve different purpose and have different characteristics so do the algorithms inspired by them. All research in the field of AIS stems from foundations in theoretical immunology. The area of immunology itself is in the process of developing and there have been major discoveries and paradigm shifts in the field (Figure 4).

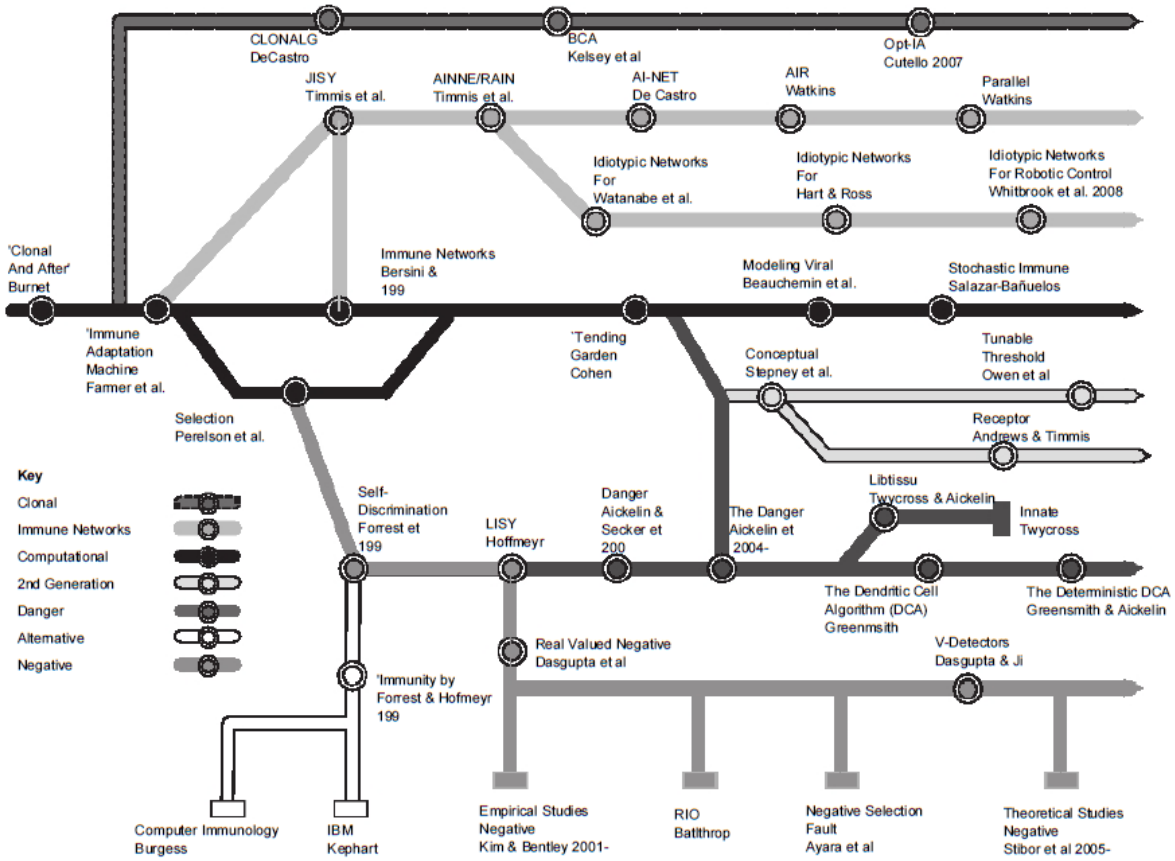


Figure 4 – Artificial Immune System Development

Immunologists like to depict the immune systems as comprised of two parts: the innate immune system and the adaptive immune system. It was believed that these two systems are distinct sub-systems with little interaction between them with the innate system reacting to previously encountered threats and the adaptive immune system responding to unknown threats. However with the advancements in the area and new findings emerging suggesting that it is the interaction between the two systems that ensures the high level of protection required. Such findings of course influenced the approaches used in computing which is expressed in the difference between two generations of algorithms namely first and second-generation algorithms. The first generation of algorithms use very simplistic models of immunology as inspiration for example negative selection[18] and clonal selection[19]. The second generation of algorithms are built on the foundations of interdisciplinary research that allows finer understanding of the foundations in immunology and sometimes are based on first generation algorithms. The second generation algorithms have emerged relatively recently which means there is much more theoretical study required but they show great potential.

2.4.2 Artificial Immune Systems in Optimisation

There is a natural parallel between the immune system and optimisation. The immune system is not specifically an optimiser but the process of generating antibodies in response to antigens and constantly improving is evolutionary in nature and therefore allows the process to inspire the implementation of number of algorithms. An algorithm that has received much

attention is aiNET. The algorithm was developed for data compression and clustering[20] but later improved[21] to perform multimodal search to create the opt-aiNet algorithm. Opt-AiNET evolves a population of antibodies representing candidate solutions connected in a network which interact with each other and undergo the process of clonal expansion, mutation, evaluation and selection. As in the natural immune system Opt-AiNET creates a set of "memory set of individuals" which embody the best solutions found. The main principles representing Opt-AiNET are:

- Dynamically adjustable population size
- Both exploration and exploitation features are present
- It determines the location of multiple optima
- It has the capability to maintain multiple optima (does not converge to a single point)
- It has defined stopping criteria

2.4.3 Particle swarm optimization

Particle swarm optimization[22, 23] (PSO) is computational method use to optimize a problem by iteratively looking through a search space by constantly improving the candidate solution represented by particles which have the attributes of position and velocity. As the name might suggest Particle swarm optimization method is part of swarm intelligence field and it follows the same general principles. PSO is slightly different than standard evolutionary algorithms in the aspect that individual of the population do not parent new individuals but change their behavior and phenotype based on set of rules. The particles interact with each other and "cooperate" by adjusting their behavior according to personal best solution found by a particle and the global best of the whole population.

The algorithm was proposed and developed by Kennedy, Eberhart and Shi[24] in the late 1990s and PSO was firstly intended to simulate the social behavior[25] of the movement of organisms in bird flock or fish school. Later it was simplified and observed to perform optimization. In their book "Swarm Intelligence"[26], Kennedy and Eberhard further elaborate on possible uses of PSO. A decade later Poli[27, 28] conducts a thorough research on possible application for PSO and just recently Bonyadi and Michalewicz published a comprehensive review on theoretical and experimental works on PSO[29].

In PSO the choice of parameters such as swarm size, velocity and affinities towards personal and global best known solutions have a great impact on the performance of the algorithm therefore warrants a huge amount of research[30, 31, 32, 33, 34]. The parameters should also be tuned according to the problem to balance out between exploration and exploitation to avert premature convergence to a local optimum and failing to find the global one (Figure 5). In regards to PSO convergence refers to either the convergence of all particles to a single point in the search space which may local or global optimum or convergence to an optimum where the swarm's global best solution approaches the optimum of the problem with no concern of the behavior of the swarm. The first definition refers to a case in which all particles are

"stuck" on local optima which in most cases is undesirable and aimed to be avoided. The second definition is the case in which the best solution is found and the behavior of the particles is not of importance anymore. There are studies[35, 36] on convergence which aim to improve the performance of PSO by avoiding premature convergence, improving convergence time, solution quality and robustness of the algorithm but they have been criticized[37] for methodological flaws or lack of theoretical evidences.

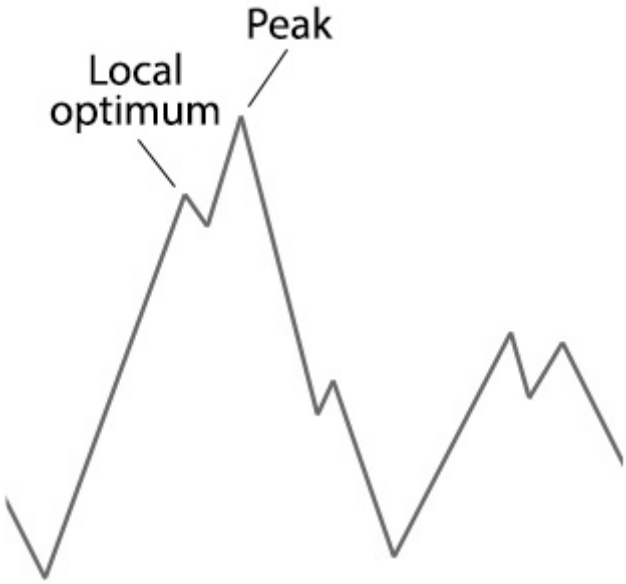


Figure 5 – Local and Global Optima

2.5 Combining two algorithms

Hybrid genetic algorithms[38] like many hybrid systems aim to combine[39] two different methods, complementing each other in order to mitigate a weakness or improve strengths of one or both algorithms. In hybrid genetic systems[40] algorithms incorporate one or more optimization methods which can be combined to achieve an optimization goal. Genetic algorithms can be seen as global search method it can be combined with a problem-specific method as a "local search" to improve search efficiency. This could also ensure the production of feasible solution as sometimes genetic algorithms produce invalid solutions which also need to be evaluated and rejected which uses up computational capacity. The recently growing area that is interested in that type of global-local search synergy is known as Memetic algorithms (MA)[41], Lamarckian search[42] or Baldwinian search[43].

There are several aspects which a combination of search techniques can improve the overall efficiency:

- Improving Solution Quality
- Improving Efficiency:
    - Convergence Speed

    - Population Size
- Guarantee Feasible Solutions
- Operation Substitution

There are different approaches on how improving solution quality can be achieved. Holland, cited in[44] suggested that genetic algorithms should be used as pre-processor for performing the initial search and then using a local search method to optimise the final population. Bilchev and Parmee[45] used an ant colony optimization model for continuous search spaces as a local search method to improve the quality of the solutions produced by genetic algorithm solving heavily constrained engineering problems. It is also extremely beneficial that integrating such a local search introduces diversity which helps resisting genetic drift tendencies and enables better representation of different areas which fights premature convergence.

The Improvement of efficiency in a hybrid algorithm is usually based on the fact that genetic algorithms are good at isolating promising areas of the search space and therefore decrease the time needed to find the global best solution or memory needed to process the population. In real world problems the complexity of the problems is overwhelming. Evaluating the fitness of all solution is usually the most computationally consuming process and when hybridisation offers "relaxation" on that process the search speed improves significantly. Another aspect of efficiency improvement is the reduction of population size and therefore memory required and convergence rate. Some algorithms struggle with crossing over extrema (local minima) to get to another optima lying on the other side of it and when combined with a technique which is good at dealing with it, that weakness is deflated and consequently so is the population required to do so.

In highly constrained optimisation problems genetic operations such as crossover and mutation produce a vast amount if illegal solutions and waste computational resources. Implementing a domain-specific knowledge could help prevent the production of such illegal solutions or repair them so that premature convergence is avoided which occurs when all or most of the solutions are illegal[46]. As an example a force feasible operator[47] was used to solve scheduling aircraft landing times problem.

Operation substitution is the incorporation of different techniques which can replace the genetic operators, crossover and mutation. There are also algorithms that do not allow or use the implementation of one of the genetic operators. It is possible that the genetic operator is introduced by using it within other algorithm and then combined to the original one.

In this chapter all the relevant principles of evolution and their application in the field of computing were discussed. Further it covered how evolutionary algorithms based on "swarm" or population techniques work and cover how hybrid algorithms work in order to give insight on how to approach goals of the research.

3. Requirements

As described in chapter 1, the aim of the project is to determine whether combining Opt-AiNet and PSO would result in hybrid system with overall better performance. The list of functional requirements (Table 1) describes a set of functionalities and their priority in respect to the complete final software.

3.1 Functional Requirements

Number	Requirement	Priority
1	Load search problem boundaries from config file	High
2	Opt-AiNet algorithm works corectly	High
3	PSO algorithm works corectly	High
4	Log the fitness of all individuals of PSO	Medium
5	Ensure PSO and Opt-AiNet "communicate" correctly	High
6	Log population size in Opt-AiNet	Low
7	Log generations taken until termination conditions met	Medium
8	Log time taken until termination conditions met	High

Table 1 – Functional Requirements

Functional requirements justification:

1. The config file consists of a set of predefined variables and it is important that the software do not fail to load it.
2. Manually testing all Opt-AiNet functions for correct behaviour.
3. Manually testing all PSO functions for correct behaviour.
4. Logging the fitness value of all individuals created by PSO so they can be tested against the fitness values Opt-AiNet gives the same individuals. This is done in order to ensure requirement No. 5 is met.
5. This is the most critical requirement for the software. Manually testing the interface between Opt-AiNet and PSO to ensure the right data is passed, data is uncorrupted and correctly assimilated by the other algorithm.
6. Log the population size of Opt-AiNet to evaluate if using the Hybrid method will decrease the population needed until termination criteria is met.
7. Log the generations needed until termination criteria is met in order to evaluate if using the Hybrid method will decrease it.

8. This is the most important variable that needs to be evaluated. It is the time taken for the algorithm to complete the problem optimization.

3.2 Non-Functional Requirements

Non-functional requirements are often called “Quality attributes”. In contrast to functional requirement they do not describe how the system should behave but rather desirable properties of the project. Some important non-functional requirements for this project are:

- Recoverability – Being able to recover the software in case of disaster or data loss.
- Testability – It is important to be able to test the behavior of the software. In the domain of optimizing problems it is the case that it is unknown what value to expect or in other words it is a “black box” problem. In order to ensure that everything is working as desired test function need to be implemented into the software so behavior can be monitored.
- Extensibility – It is often the case that after completing the goals of a research, more research questions based on the results emerge. It is desirable that the software design takes into account possible future work and allows the extension of functionalities if needed.

4. Methodology

The project follows the AGILE approach. The project plan and project phases (Figure 6) should be planned out and clear at the start of the project and revised with the completion of each phase if needed. To address the needs of the project a thorough literature review should be conducted in order to get to a sufficient understanding of the problem domain. Previous work done in the area should serve as inspiration and guidance of the research. Appropriate tools existing should be identified and assessed if they would serve a purpose for the project. At that point the designing phase will commence and the main objective will be coming up with a well-thought plan on how to approach the implementation in most optimal way and avoid potential problems in the following phases. In the implementation phase the goal is to produce a working product. In the later phases of implementation comes the testing. Testing will be conducted to insure that the software is working as intended. At the final stages of the project the main focus of the efforts should be focused on debugging and polishing early design choices.

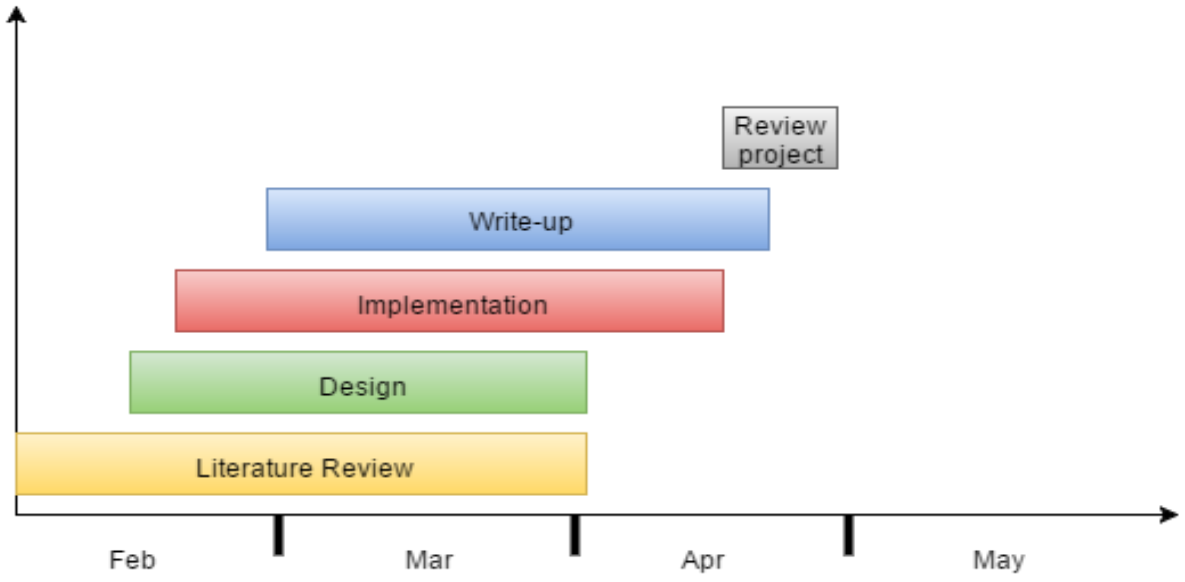


Figure 6 – Project Phases

5. Design

This chapter points typical issues with hybrid algorithms so they can be avoided in the design process. The chapter continues with defining principles, behavior and properties of PSO and Opt-AiNet and elaborates on how they will affect and lead to the initial design proposed then demonstrates the initial design and elaborates on the thought process behind it. The chapter discusses the issues with the initial design and the reasons it had to be rejected and then describes the reviewed design and justify why it is likely to work.

5.1 Hybrid design issues

Combining two algorithms can improve the search performance under the condition that their roles cooperate to achieve the optimization goal. There is the opportunity to get the best of both techniques but this depends on the design choices and details. There are some issues and dangers in designing a hybrid system which should be reviewed and be careful about. Typically when combining two algorithms one is seen as a global search and the other as local search. The global search could be seen as embodiment of exploration and the local as exploitation. The danger here is how to balance between both. The role of the global search is typically to isolate the areas which contain the best solutions and the role of the local search is to find the best solutions in these areas. The ratio between exploration-exploitation is problem dependent but even more so on the algorithms used. If there is not enough exploration there is the risk the algorithm will end up with relatively good solutions from bad

areas which are not very optimal or suffer from premature convergence. On other hand if exploration rate is too high there is a good chance the algorithm will find many good areas of solutions but the local search will have hard time keeping up and end up with relatively bad solutions from the good areas. Also the high exploration rate leads to increasing the search area which will be comprised of range of bad or good areas which will need more and more computational resources as it grows.

5.2 Similarities and differences

In order to carefully and correctly come up with a design which has high as possible chance to work as intended, a careful consideration and analysis of techniques and tools must be taken into account therefore the core similarities and differences between Opt-AiNET and PSO should be reviewed (Table 2).

Properties	Opt-AiNet	PSO
Population size:	Dynamically adjustable	Constant
Population behavior:	Competition	Cooperation
Individual representation:	Phenotype	Phenotype
Parent-Child:	One Parent – Many children	No parent – No children
Genetic operators:	Clonal expansion, Mutation, Clonal selection, Clonal suppression	Individual changes phenotype according set of rules and interaction with other individuals (could be seen as type of guided mutation)
Evaluation type:	Phenotype	Phenotype
Natural Selection:	Yes	No
Determines the location of multiple optima:	Yes	Yes
Has the capability to maintain multiple optima:	Yes	Yes(limited)
Exploration and Exploitation abilities:	Poor exploration, Excellent exploitation	Excellent exploration, Poor exploitation
It has defined stopping criteria:	Yes	Yes

Table 2 – Algorithm Similarities and Differences

What do these properties and characteristics of Opt-AiNet and PSO mean for design choices in regard of combining them?

- Population size in Opt-AiNet is dynamically adjustable which means that it changes its size with each generation, typically growing when clonal expansion is applied and then slightly trimmed down by clonal suppression which "kills" unfit solutions. This process leads to a population size that grows bigger with the scope of the problem to be solved which means that the algorithm has some scalability issues. PSO on other hand has a constant sized population which is predefined for the problem. The issue here is that inadequate size of the population might either end up in no good solution found if too small or waste of computational resources if too large.
- Population behavior in Opt-AiNet can be seen as competition. An analogy can be made with real world organism like plants. If two plants grow too close to each other they will compete for resources like nutrients from the soil and sunlight. Eventually the better individual with grow larger root system, overshadow the other and the less fit individual will die. In Opt-AiNet respectively when two or more individuals are too close (depending on defined thresholds) only the fittest solution will be kept. In PSO on other hand individuals cooperate. When an individual finds a solution which is the best solution found so found, it notifies all other individuals about it which changes the behavior of the whole population. Eventually due to that cooperation all individuals will converge to the single best solution area.
- Both PSO and Opt-AiNet do not have a genetic representation (e.g. bit string). They are represented by their phenotype or their attributes like location, velocity for PSO and affinity for Opt-AiNet which is expressed in real values like coordinates. The fitness of the individuals is evaluated on the basis of these values (coordinates).
- Each individual in Opt-AiNet "parents" many children by cloning itself. The number of times it clones itself is based on how fit the individual is. PSO population does not change in terms of individuals over time but rather each individual changes its phenotypic characteristics based on its interaction with other individuals of the population.
- Clonal selection and Clonal expansion are responsible to determine the affinity and fitness of all individuals in Opt-AiNet and reproduce them proportionally to their affinity. A mutation rate is introduced with every clone, usually the mutation rate is inversely proportional to the fitness of the individual which keeps the good solutions and tries to improve on the bad ones. Clonal suppression is the mechanism responsible for killing the unfit solutions too close to a good one and by doing so keeping a nice spread of the population. Opt-AiNet seems to struggle when crossing over extrema unless the mutation rate is high enough which will increase its ability to of exploration but that also hides the danger to reduce the ability of exploitation. Alternatively PSO does not really make use of a genetic operator but rather change the behavior of each individual in accordance to the interactions of the swarm.
- As aforementioned both PSO and Opt-AiNet individuals' fitness is evaluated by the fitness function based on their phenotype or more concretely their location in the search space.
- Both algorithms are able to identify multiple optima as each individual will try to increase the overall fitness of the population and eventually come across multiple

local optima. Since Opt-AiNet incorporates the mechanism of the immune system to "remember" by keeping "memory cells" and the clonal suppression mechanism keeps all individuals spread it allows for the algorithm to keep multiple optima. PSO individuals can only "remember" their personal best solution and the global best which limits the algorithm ability to maintain multiple optima to the exact number of population size.

- PSO and Opt-AiNet have the same stopping criteria, which is expressed in either reaching the maximum amount of generations or reaching a plateau in regards to finding optima. In the case of Opt-AiNet it is important to note that since the population can grow infinitely, if the problem allows it and termination criteria are permissive, there is a chance that the machine will run out of memory and terminate the process.

5.3 Initial design and design choices

Baring in mind the potential design issues of hybrid systems and the core characteristics, strength and weaknesses of both algorithms I came up with the initial design (Figure 7). The core idea behind the design lies in the difference in principle of population behavior of both algorithms – cooperation and competition and their possible combination and synergy. The reasoning behind this approach is to keep the strengths of techniques while avoiding their weaknesses. While competition ensures a good spread and diversity of solutions it suffers from the lack of exploration, in which cooperation excels. If the design allows for such structure that balances out and alternates between the two behaviors the population will benefit from both and therefore improve the search capability.

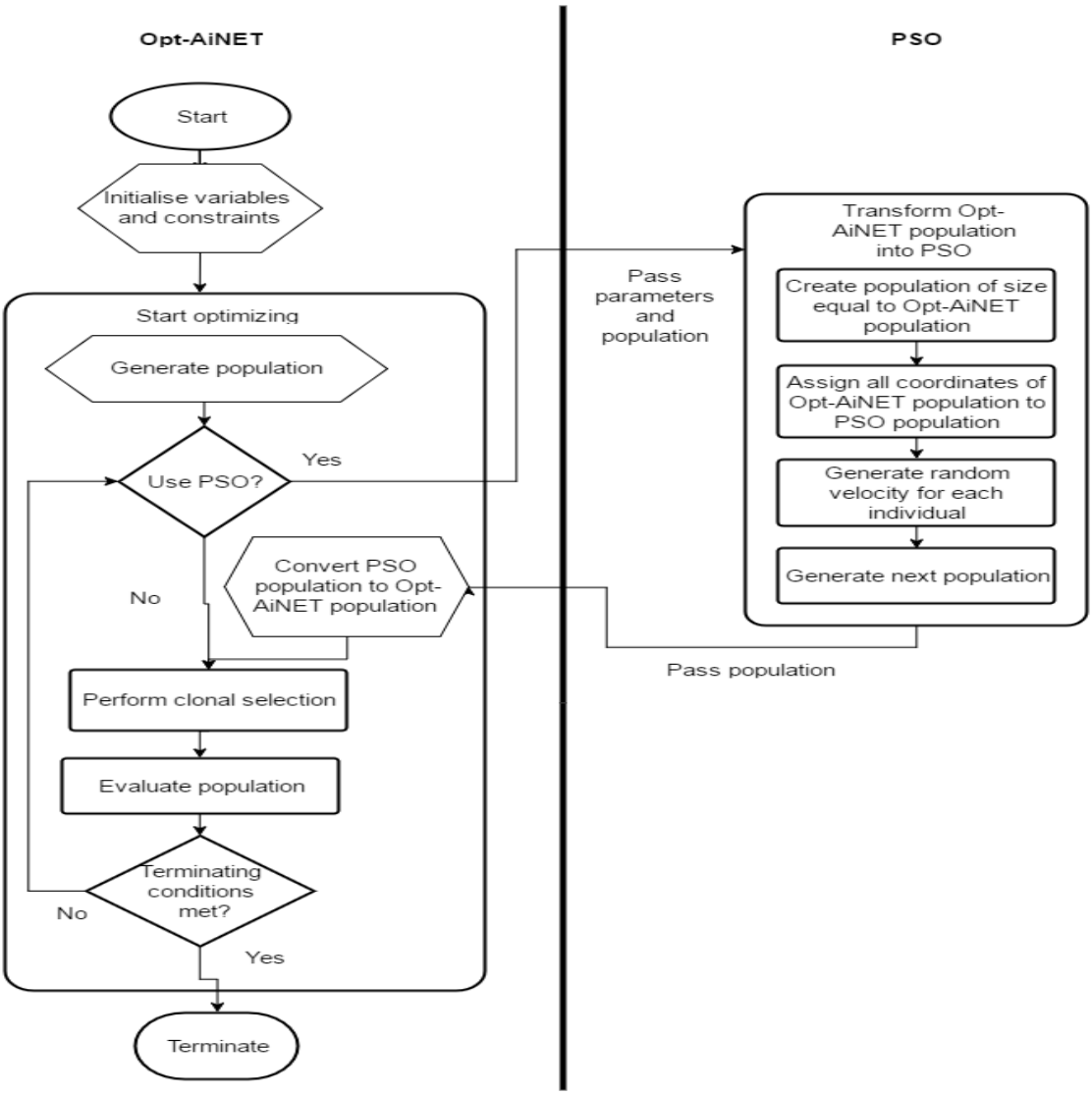


Figure 7 – Initial Design

The initial architecture design was to modify both algorithms and build an interface between them so they could communicate. PSO and Opt-AiNet individuals are evaluated by the same characteristic – their location (coordinates). Also since they solve the same problem their fitness function is the same, meaning that for the same individual or set of coordinates, both Opt-AiNet and PSO will produce the same evaluation. If PSO is modified to be able to have an adjustable size of population rather than constant, then the two algorithms would be able pass the same population between them and "take turns" in optimizing it. In (Figure 8) it is illustrated how at Opt-AiNet generates the initial population, passes it to PSO which updates the population returns it back, Opt-AiNet uses clonal expansion and selection, evaluates the population and if the stopping criteria is not met repeats the whole process from step one.

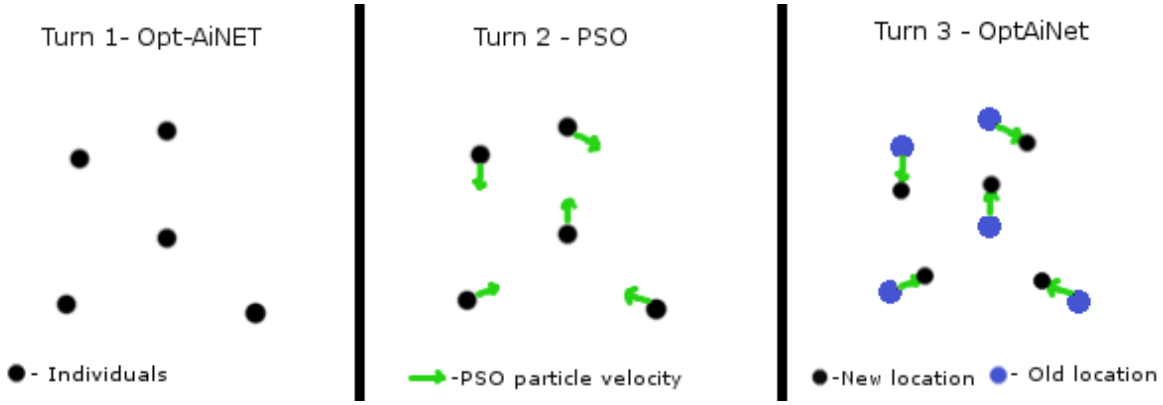


Figure 8 – Hybrid Process

5.4 Design issues

As I was further elaborating on the design some conceptual and computational problems became apparent.

While Opt-AiNet will not have any trouble using the population produced by PSO as it will be treated like a population of new clones, PSO needs all individuals to have velocity in order to move through the search space. That means every time it is PSO's turn it will have to initialize brand new population of Opt-AiNet size and assign the corresponding coordinates to each individual. This increases the computational time and the memory needed as the machine will have to store two populations of equal size in the process of transfer. This alone may outweigh the benefits of combining both algorithms. Also there's another problem with the velocity of particles. The vector of velocity updates according to the interaction between the individuals and as this will be a brand new population generated they will not have any velocity set. If random velocity is assigned to each individual as it is usually done in the first generations of PSO there is a huge risk that the individuals will shift out of good locations. Storing the vectors of the previous PSO population will not help either since the size of the population have most likely increased, which leaves some individuals with random velocities which leads right back to the problem shifting good solutions out of place and also will need more memory. Furthermore the individuals from the previous population might not exist, switched places and there is no way to discriminate which velocity belongs to each individual and will be no better than assigning random velocity values.

A conceptual error of this design is that Opt-AiNet makes use of some individuals to become "memory cells". These individuals serve the purpose to keep the best found solutions in memory by staying at the respective location. These individuals rarely undergo any change as it is desired that they keep multiple optima in the "memory" of the immune system algorithm. The problem comes from the fact that if we pass these individuals to PSO they will shift location which will effectively erase the "memory" of the optima they represent. In order to keep these individuals intact it is possible to only pass to PSO the individuals with lowest fitness but in order to do so the algorithm must keep them in memory which could lead to scalability issues. It is also hard to balance how many and up to what point Opt-AiNet picks individuals to pass to PSO.

Even though this design may provide some ideas on how Opt-AiNet and PSO might be combined to work together, any of the aforementioned issues with the design is sufficient to point out that the initial design will not work in its current form, needs to be conceptually revised and is therefore rejected.

5.5 Revised design

After revising the issues of the initial plan and the concepts and behavior of both algorithms I came up with a new improved design. In most of the works on combining two algorithms reviewed, one of the algorithms assumes the role of global search and the other of local search. In the case of the initial design both algorithms were both global and local search. Also for PSO to be effective it needs to run for at least few generations. In one of the works previously reviewed[34], the particle trajectories and their behavior are studied and three phases are described – Explosion, Stability and Convergence. During the explosion phase the particles expand in the search space very quickly and it is also the phase in which local optima are found very quickly even if there is no guarantee that they are very good solutions. This means that if PSO is allowed to run through its explosion phase and is terminated before the stability phase, the algorithm could potentially seed out good areas of the search space with excellent spread since the phase is mostly exploration for minimal computational resources. Also with this approach PSO will assume the role of global search technique and then Opt-AiNet can take on the local search. Another positive effect of that is that Opt-AiNet will get aid with crossing multiple extrema with which it struggles. Also with this technique PSO and Opt-AiNet do not have to "take turns" but rather have PSO prepare the best possible initial population for Opt-AiNet and let it do the rest of the job. This way PSO keeps the constant population size which avoids the scalability issues and also the compatibility issues of the individuals when converted from Opt-AiNet to PSO. PSO algorithm usually works well with a very small number of particles which means that it will perform well with a swarm size equal to the initial swarm size of Opt-AiNet or even slightly bigger. As the locations of the population of Opt-AiNet is randomly generated and the size of the population grows rapidly this guarantees that in the worst case scenario using this approach will not be slower than not using it therefore it can potentially bring benefits without any risk. This is the "corner stone" of the improved design (Figure 9).

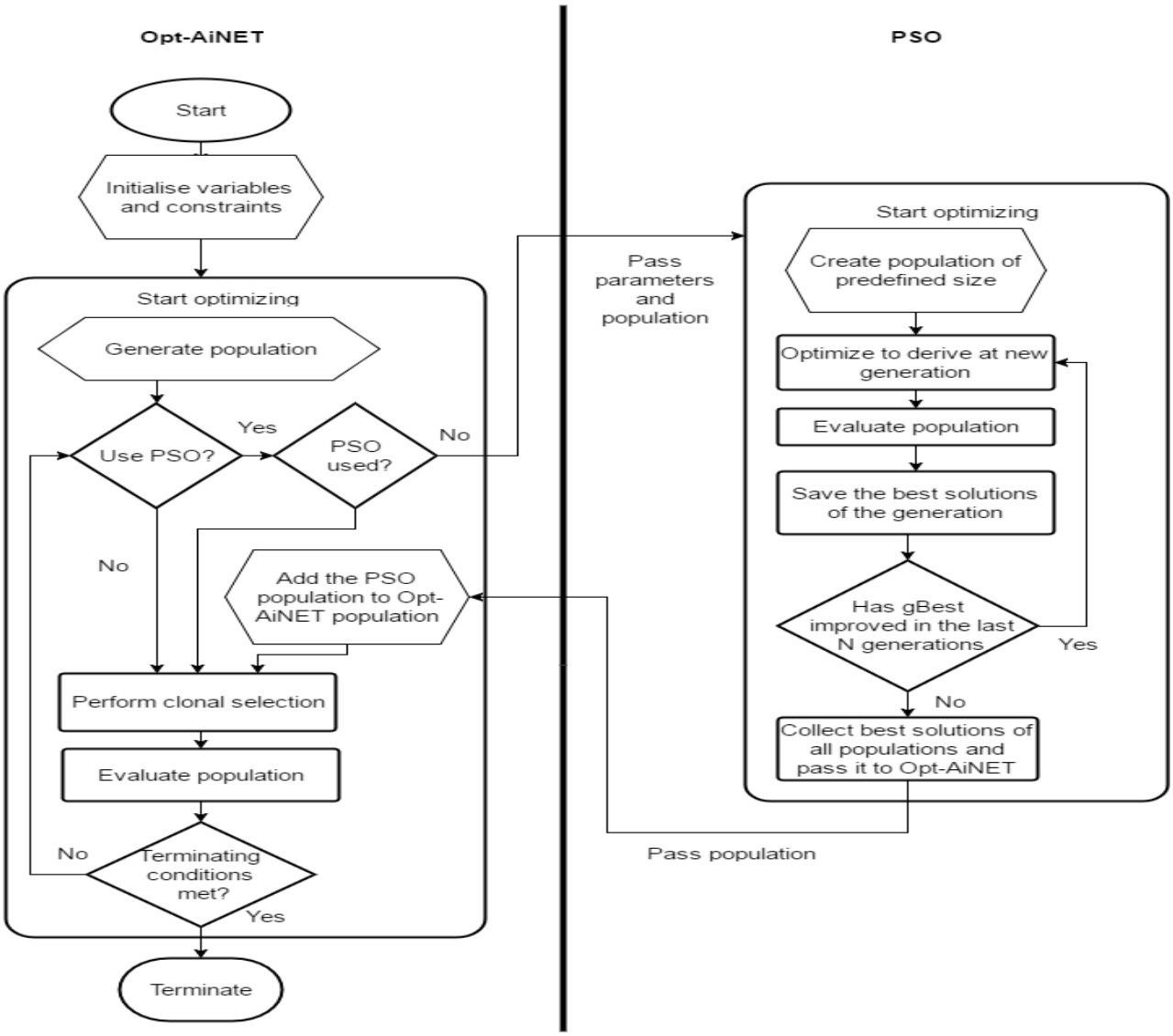


Figure 9 – Reviewed Design

The revised design is somewhat inspired by the initial design but is quite different in its core. At the start a flow control (if statement) allows to decide whether PSO is used or not which is mainly used for evaluation of performance. The second flow control is introduced to ensure that PSO is only used once. Potentially PSO can be used many times but with each use the ratio between the populations of Opt-AiNet and PSO grows and proportionally decreases the usefulness of PSO which may be a waste of computational resources. If under the predefined conditions PSO is to be used, PSO initiates its optimization process. Here it is important to note that choosing an adequate size of population may improve performance but the tradeoff is large size can provide with more and better solutions but it will be more computationally expensive. An important alteration of the PSO algorithm is needed and that is the ability to remember all the best solutions found and their coordinates so it can later pass them Opt-AiNet. The rest of the optimization process is standard for PSO but is of immense importance when to stop it. A flow control assumes the function of termination criteria. Its function is important because as soon as the algorithm exits the explosion phase, it hits a plateau and enters the stability phase. It is desirable at this point for the PSO to stop to avoid resource

waste as it is less likely to find any better solutions. A variable in the flow control counts the generations passed in which no better solution is found. It is worth testing and evaluating weather being restrictive or permissive with this threshold would improve the overall performance of the hybrid algorithm. After PSO has met its stopping criteria it passes all the best solutions found to Opt-AiNet. Opt-AiNet in turn adds the population with good fitness from PSO to its population and proceeds with its standard optimization process. Initially PSO assumes the exploration role and Opt-AiNet the exploitation and by doing this Opt-AiNet receives an immense help with what it struggles to perform which the main reason this design may results in significant increase in performance.

In the design chapter potential issues with hybrid algorithms were pointed out. Then it illustrated the initial design idea, the reasons why it would not work and then describes the final design with which the research will proceed into the implementation phase.

6. Implementation

This chapter will discuss the methods and tools used in the development of the software, the development process behind it, show the architecture of the complete software and the techniques for disaster prevention.

The project was developed in Java. Java is an open source programming language created by Sun Microsoft and overtaken by oracle. Java is platform independent which means it runs on any computer satisfying its basic needs with no hardware or software dependencies. Java latest Just in Time (JIT) compilers approach the speed of C/C++ code and in some memory intensive circumstances exceeds it. Java also have standard API's meaning that any machine with recent enough Java version will be able to run the program. Java's garbage collection also alleviates the need to worry about memory, especially with memory demanding algorithms. Eclipse provides IDEs and platforms for nearly every language and architecture and is famous for their Java IDE. This platform delivers most extensive collection of add-on tools available to ease the process of software development.

6.1 Development process

The development process followed the exploratory programming approach as even though there is some research done in hybrid algorithms no research has been done in combining Artificial Immune Systems with Particle Swarm Optimisation to optimize search space problems. The first step of the development process was to develop both Opt-AiNet and PSO

algorithms independently, tested if they work correctly and then proceed to modifying them. In software development the philosophy "Don't reinvent the wheel" is necessary to avoid wasting time on algorithms and implementations that are readily available. Both algorithms are well known and have reliable free distributions which could be used. Opt-AiNet is freely provided by the International Conferences on Artificial Immune Systems (ICARIS) on their dedicated to AIS algorithms webpage[48] and is the source used for Opt-AiNet. PSO on other hand didn't have any reliable java sources and I developed it based on Gandhi Manalu's PSO tutorial[49]. This adaptation of PSO had several mistakes that surfaced in the development phase and needed correction. A conceptual mistake was the assumption that the lowest evaluation function value is 0 which lead to erroneous calculations. Another error was the lack of search space constraints which would allow the particles to wonder out of the search space and generate infeasible or irrelevant solutions to the problem. Both of these errors along with few small code errors were corrected in the development process resulting in a reliable and correct PSO algorithm. Both algorithms fitness function was manually tested to ensure they provide the same evaluation values for the same individuals.

After both algorithms were available and working the next phase of development was to create and interface and modify both algorithms to follow the hybrid algorithm design. The problems solved by both these algorithms have unknown derivatives and known as "Black box" problems it is hard to follow the Test-driven development process. Instead the approach used was to write test function that log the system behaviour and monitor manually each modified step's behaviour and results to ensure they work correctly.

Following the aforementioned methodologies and design ended up with in complete, working hybrid system that is able to optimize search problems (Figure 10). With the design guarantee that in the worst case scenario the system will not perform slower than the original, the final step left was to test and evaluate the performance of the system and see if this approach is successful and improves the overall performance.

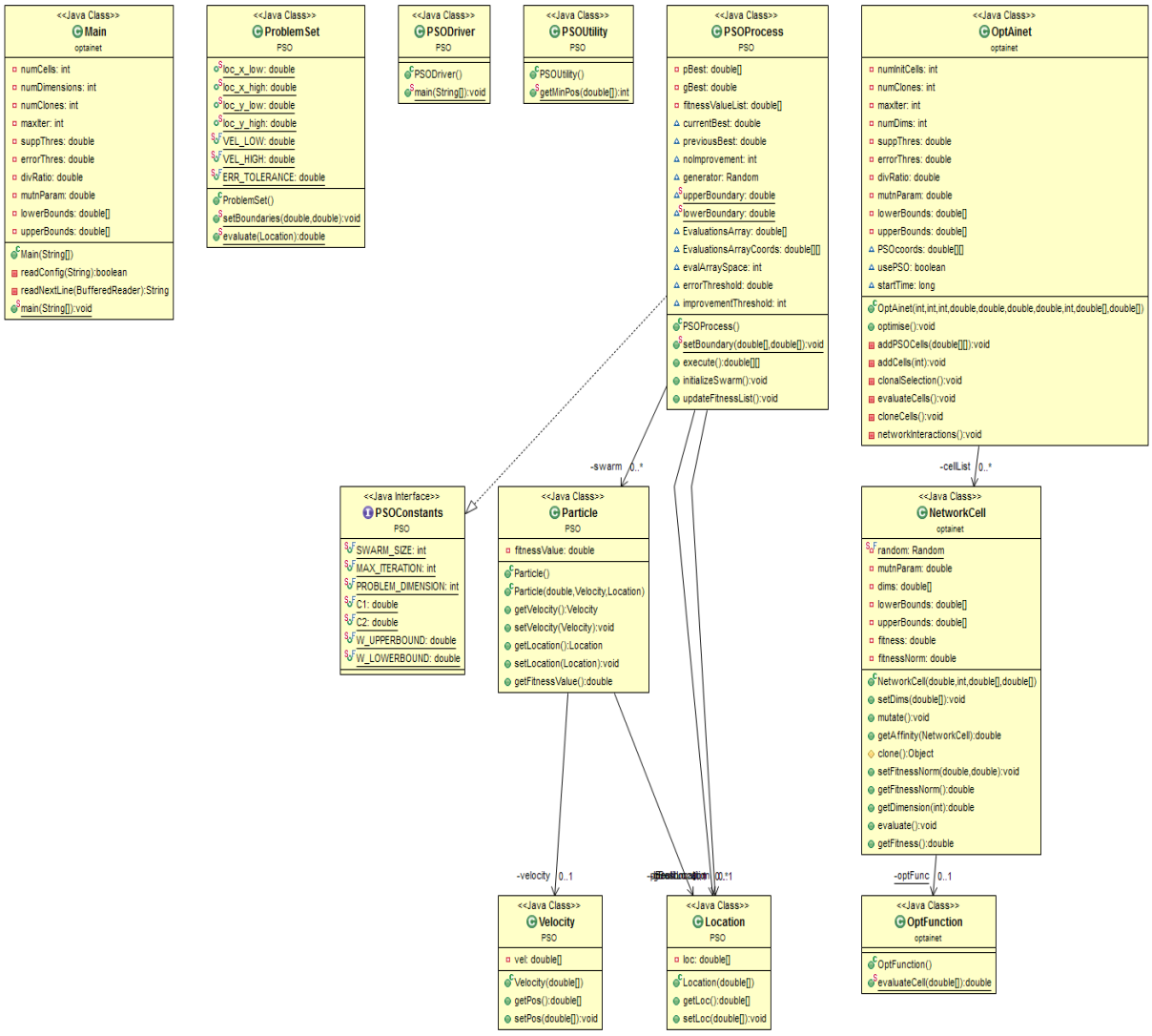


Figure 10 – Class Diagram

6.2 Version Control and Back-up

Version control and backing up data are extremely important when developing software in order to prevent disastrous events which may lead to the loss of the whole project. Version control ensures that any changes made to the software are logged and if needed reversed. In the process of exploratory development it is often the case that the developer needs to take a step back and revise the approach and code. For the goals of this project the version control approach of choice is Git[50]. Git is free open source that can handler small and large projects and is extremely easy to learn and use which makes it perfect for the needs of this project. To ensure the prevention of data loss the project directory was synchronized with a third party, cloud repository – Google Drive[51].

7. Evaluation

This chapter discusses the goals of evaluation, defining the research questions and respectively defining the null and alternative hypotheses. After that the experimental design is discussed and a clarification on how the tests were designed and how to read the data in them. Before each set of test a justification for the choice of test is given and after each section an elaboration on what the results mean is given. At The end of the chapter a review of the results of all tests is given. With that an answer to the research questions is given and a discussion on insights gained in the process of evaluation.

Having complete and finished software will allow testing and evaluating the hybrid algorithm and possibly gain an insight from the results. There is an immense amount of different variables that could influence the performance of the algorithm some of which considerably and others slightly. This evaluation will focus on investigating whether the proposed hybrid algorithm performs better than its component algorithms based on optimizing different functions. Also apart from the problem, the variables that have biggest impact on the performance of the algorithms are constraints (size of the search area) and starting population size both form Opt-AiNet and PSO.

Evaluating the performance of the hybrid algorithm can be measured in 2 aspects – Quality of solutions and the speed of completing the task. Since the PSO algorithm converges to a single solution point and has a constant amount of population, the average population fitness will be extremely close to or equal to the best optima found. Also the fact that PSO is not able to maintain multiple optima will not be a fair testing criterion or bring any valuable insight therefore Opt-AiNet would be the benchmark for both quality and speed.

It is important to formally define the research questions and the experimental design before taking up on the evaluation.

The research questions are as follows:

- Does using the hybrid algorithm lead to improvement in the average quality of solutions compared to Opt-AiNet?
- Does using the hybrid algorithm lead to a speed increase in solving the problem compared to Opt-AiNet?

The null hypotheses are:

- H_0: Using the hybrid algorithm will not improve the quality of average solutions compared to Opt-AiNet
- H_0^I: Using the hybrid algorithm will not improve the speed of solving the problem

Respectively the alternative hypotheses are:

- H_1: Using the hybrid algorithm will improve the quality of average solutions compared to Opt-AiNet
- H_1^I: Using the hybrid algorithm will not improve the speed of solving the problem

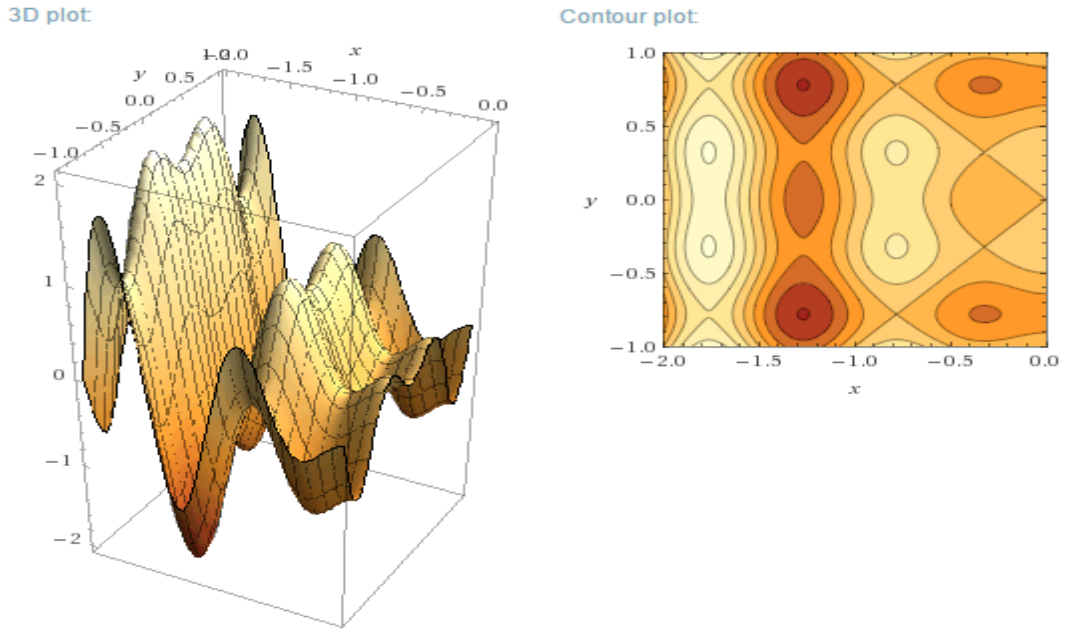
A good practice is to choose a significance level of the data before the evaluation[52]. Usually the level of significance depends on the field of study but it is often set to 5% (α=0.5). This means that the results are significant at if the p value of the test is p< α or p<0.5 in which case concluding that there is strong evidence supporting rejecting the null hypothesis and accepting the alternative hypothesis.

7.1 Experimental design

Appropriate targets for this experiment are multimodal optimization functions which are selected from the UC Irvine Machine Learning Repository[53]. The parameters for evaluating the performance of the algorithms based on these functions are initial population size, where starting population size of Opt-AiNet is equal to the sum of populations in the hybrid algorithm (e.g. Opt-AiNet - 40 individuals / Hybrid - Opt-AiNet -20 individuals + PSO 20 individuals). The second testing parameter is the search constraints which reflect in the search area (e.g. search space from -10 to +10). The data collected from the test will be plotted in a table indicating the parameters, average quality population or time taken for a single test, average values for the whole test set and statistical significance values allowing to reject or accept the null hypotheses based on the data.

Function 1

y*sin(2* π *y)-x*sin(2*(π *x+ π))+sin(2* π)*π



This is a multimodal function, meaning it possesses multiple good solutions. Also it has particular shape characterized by high extrema surrounded by optima which as aforementioned is something that Opt-AiNet struggles which is ideal for the test as it may show how the hybrid algorithm mitigates this weakness.

Average Fitness of population – Quality

The first aspect to be evaluated is the average quality of solutions (Table 3).

Standard Opt-AiNET	Opt-AiNet + PSO
Population average Quality	Population average Quality
10.59195074	10.46195104
10.60531159	10.54528638
10.606142	10.51413583
10.56848088	10.41706268
10.61742497	10.47467969
10.58118041	10.45276309
10.57992947	10.4190394
10.54849065	10.45427878
10.53526914	10.52485194
10.56023537	10.39842087
10.49063239	10.47335555
10.47156641	10.34023032
10.465662	10.339433
10.51804025	10.40206788
10.54572307	10.37851569
Average	Average
10.55240262	10.43973814
F	30.3880144
P-value	6.84664E-06
F crit	4.195971819

Table 3 – Average population fitness comparison

The data suggest that the average population quality between the two algorithms is smaller than the variance between the instances of each algorithm alone therefore this suggest that the hybrid algorithm does not improve the average quality of the population. The *f* and *p* values show that data is statistically significant.

Average Speed and Generations

This section will evaluate the speed performance of both algorithms. The tests are conducted in sets of 2 tables showing the performance of the algorithms with respect to different variables. As mentioned before the starting population size of both should be equal. Since in the initialization phase of the hybrid algorithm the starting population of PSO is added to Opt-AiNet the sum of both should be equal to the population size of the standard Opt-AiNet algorithm.

SET 1 – Opt-AiNet starting population 40 / Hybrid Opt-Ainet -20 + 20 PSO

The differences in performance between the two algorithms based on time taken (measured in nanoseconds) and iterations (generations) can be observed. The population size is also observed since if the hybrid algorithm is faster it may also reduce the population size required to complete the task.

Standard Opt-AiNet - boundaries: -10/+10		
Time	Iterations	Population
6015397272	21	445
7680991132	23	456
8071312065	25	456
8881326652	27	456
6810525459	23	456
6057463820	21	452
10399296406	30	455
6968488046	23	453
7555202512	24	456
5328631309	19	442
6944921615	23	453
9643681093	29	456
9027684179	28	456
8553477038	27	457
7959715006	25	457
Average	Average	Average
7726540907	24.53333333	453.7333333
F	*P-value*	*F crit*
444.2157515	5.57009E-29	3.219942

Table 4 - Standard Opt-AiNet 70 starting population

Opt-AiNet + PSO - boundaries: -10/+10		
Time	Iterations	Population

7131430092	21	461
7471243128	22	461
6191052976	19	460
7280775567	20	456
6704935074	19	461
7746723275	22	460
7214300027	20	457
8200462726	21	456
9011114705	24	461
9220661644	23	457
7639554454	21	462
7085780604	20	460
6390460447	19	457
6155497820	19	459
6776515227	20	461
Average	Average	Average
7348033851	20.66666667	459.2666667
F	*P-value*	*F crit*
959.7373804	8.79018E-36	3.219942293

Table 5 - Opt-AiNet 20 starting population + 20 PSO

The data in test set 1 show that on average the hybrid algorithm performs 4.9% faster. The boundaries for this set of test is -10/+10 is not very large, which could mean that the 4.9% improvement is negligible or if the hybrid algorithm performance scales with complexity it could indicate that even better performance improvement could be expected in more complex instances. The average number of generations the hybrid algorithm needs to complete the task is 15.7% smaller than Opt-AiNet which correlates with the time but is not strong evidence. The population size in both algorithms is very similar which could be attributed to exhaustive search and similar spread affinities of both algorithms. The *f* and *p* values show that the data is statistically significant.

SET 2 - Opt-AiNet starting population 70 / Hybrid Opt-Ainet -20 + 50 PSO

This set is similar to set 1 but allows for larger starting population especially in the PSO part of the hybrid algorithm. The distribution is 70 starting population of Opt-AiNet against 20 of Hybrid Opt-aiNet + 50 PSO. The reason for this distribution is to examine whether allocating more individuals to the initial exploration phase will further improve the overall performance of the hybrid algorithm which could support the limited evidence from set 1.

Standard Opt-AiNet - boundaries: -10/+10		
Time	Iterations	Population
7652243493	27	451
6240416712	24	451
9252755595	30	454
4948026511	21	442
7556953975	27	456
6927551298	26	449
5762731952	23	447
7782584135	27	455
6317530282	24	444
6296953168	24	448
6523244657	25	453
8871384831	30	453
9068455423	31	458
7621571365	27	454
5831512228	23	450
Average	Average	Average
7110261042	25.93333333	451
F	*P-value*	*F crit*
455.9685806	3.29853E-29	3.219942

Table 6 – Opt-AiNet 70 starting population

Opt-AiNet + PSO - boundaries: -10/+10		
Time	Iterations	Population
5697222076	16	456
5678402580	16	457
7257195459	19	461
9184004043	22	461
4992217928	15	448
7245186899	19	461
7840812553	20	458
4519070145	14	449
6223200956	17	457
7895146857	20	460
7100950823	19	458
5193720861	15	457
8195254160	21	459
6260523989	17	459
5829951573	16	456
Average	Average	Average
6607524060	17.73333333	457.1333333

F	*P-value*	*F crit*
367.5531032	2.44426E-27	3.219942293

Table 7 - Opt-AiNet 20 starting population + 50

The results from this set suggest that the allocating more individuals to PSO in the hybrid algorithm at the start will show improvement with 7.1% against the standard Opt-AiNet. It is also worth to compare it with the Standard Opt-AiNet algorithm performance from set 1 where the initial population was 40. The difference in that case is 14.5% faster average time in advantage of the hybrid algorithm. Both of these results and the results from set 1 strongly suggest that the hybrid algorithm offers efficiency improvement if an adequate starting population size is allocated. The *f* and *p* values show that the data is statistically significant.

SET 3 – Increased boundaries to -20/+20 - Opt-AiNet starting population 70 / Hybrid Opt-AiNet -20 + 50 PSO

The coordinates in this software are defined as an object of the primitive type *double.* A type *double* has 15 decimal digits which have 10 (0-9) possibilities for each digit resulting in 10^{15} search points in the span between 0 and 1 of the search space. The boundaries set from -20 to +20 would mean 40 times that and since the problems solved are multidimensional the complexity is multiplicative for every additional dimension. This is to show that the complexity of the problem grows exponentially with respect to the boundaries set and the increase of the search space in set 3 compared to set 1 and 2 brings significant complexity increase which will allow testing if the hybrid algorithm's performance scale with complexity.

Standard Opt-AiNet - boundaries: -20/+20		
Time	Iterations	Population
46907082923	32	1714
40413242448	31	1707
57176895610	40	1717
39525496048	31	1712
41851698836	32	1713
43740443042	33	1713
38201245788	30	1708
35822397239	29	1702
35608364616	29	1702
42144116394	32	1716

39584328348		31	1712
39763957497		31	1713
31980792733		27	1708
48007061244		35	1716
45283940646		34	1714
Average	Average		Average
41734070894		31.8	1711.133333
F	*P-value*		*F crit*
708.01459	4.45805E-33		3.219942

Table 8 - Opt-AiNet 70 starting population with large constraints

Opt-AiNet + PSO - boundaries: -20/+20			
Time	Iterations		Population
37116194219		27	1715
32892208832		25	1722
43304932885		30	1722
29939775969		23	1716
29932660014		23	1711
33693559119		25	1719
39278613767		28	1717
31634598247		24	1712
39358694768		28	1720
37197355778		27	1719
35745617413		26	1717
35204974398		26	1717
38254367542		26	1714
40796288077		29	1720
28554009795		23	1713
Average	Average		Average
35526923388		26	1716.933333
F	*P-value*		*F crit*
994.5202198	4.2281E-36		3.219942293

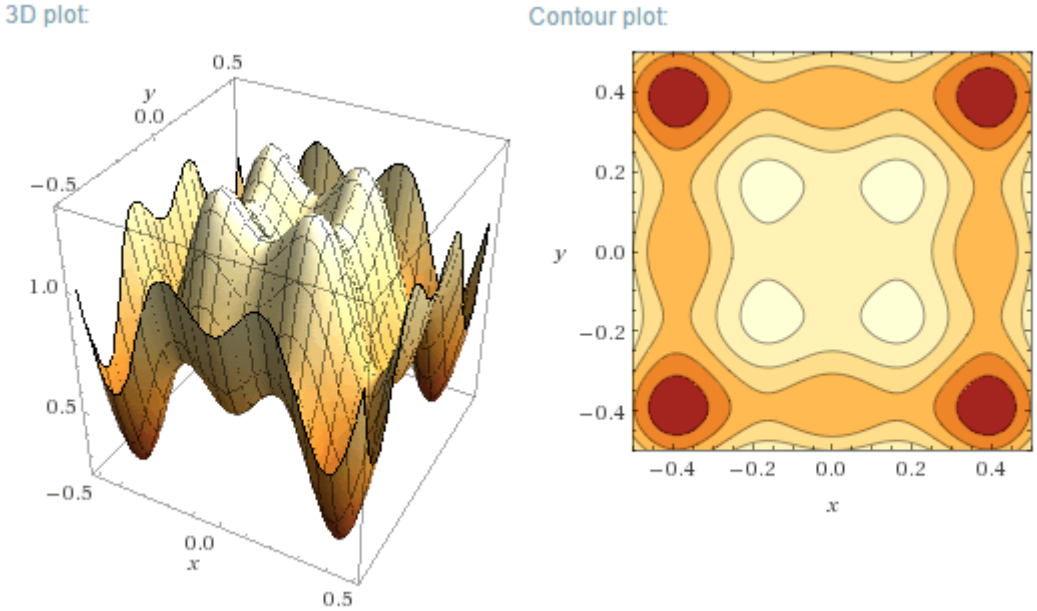
Table 9 - Opt-AiNet 20 starting population + 50 with large constraints

In this set a significant increase in time taken, generations and population size can be observed due to the increase in complexity of the problem. The data shows that the average time taken of the hybrid algorithm to complete is 15% better and the number of iterations 19% better. The size of the starting population in this set was not increased to address the change of complexity and yet the results show that the hybrid algorithm improvement is similar to the one of set 3 even slightly higher, strongly suggesting that the algorithm scales well with complexity. The *f* and *p* values show that the data is statistically significant.

In order to confirm and reproduce the results the test parameters from this set will be applied to a different optimization function.

Function 2

x*sin(4* π *x)+y*sin(4* π *y)+1



This function is also multimodal, characterised by multiple optima and extrema. The complexity of this function is significantly higher than the one in the previous tests. If the results confirm the trend from sets 1, 2 and 3 then it will be reasonable to believe that the hybrid algorithm improves the overall performance and scales with complexity.

Average Fitness of population – Quality

In order to confirm or oppose the results that the hybrid algorithm does not increase the average population quality the same test is performed on this function as well.

Standard Opt-AiNET	Opt-AiNet + PSO
Time	Time
8.234840622	10.51962772
9.24051609	9.882376594
9.613032865	10.29578201
9.489130984	9.674520759
9.511930627	9.996321293
9.46229983	9.953872021
9.783168221	9.776910928
9.674987106	9.715930534
9.761574072	9.892328465

9.790049207	9.56668085
9.646014614	9.413471021
9.61070637	9.662411199
9.674857893	9.731310953
9.492276395	9.742238009
9.549616282	9.47919292
Average	**Average**
9.502333412	9.820198352
F	6.621258
P-value	0.015665102
4.195971819	4.195971819

Table 10 - Average population fitness comparison

The data show that like in the first quality test, there is no significant difference between the average quality of population which gives sufficient evidence to accept the H_0 null hypothesis and reject the H_1 alternative hypothesis.

Average Speed and Generations

This test reproduces the test setup in set 3 on a function with higher complexity. If the trend observed in the previous test is confirmed, there will be improvement in the hybrid algorithm which will provide strong evidence allowing rejecting the H0I null hypothesis and accepting the H_1^I alternative hypothesis.

SET 4 - Increased boundaries to -20/+20 - Opt-AiNet starting population 70 / Hybrid Opt-Ainet -20 + 50 PSO

Standard Opt-AiNet - boundaries: -20/+20		
Time	Iterations	Population
1.61406E+11	43	6709
1.78445E+11	43	6697
1.73692E+11	42	6704
2.29305E+11	50	6713
1.7523E+11	42	6706
2.43669E+11	52	6716
1.67598E+11	41	6700
1.99166E+11	49	6713
2.44328E+11	52	6718
1.74561E+11	42	6706
1.55986E+11	39	6701
1.68721E+11	41	6692
2.0489E+11	47	6711
2.51239E+11	51	6710

2.15483E+11		48	6712
Average	Average		Average
1.96248E+11	45.46666667		6707.2
F	*P-value*		*F crit*
529.2436195	1.6404E-30		3.219942

Table 11 - Opt-AiNet 70 starting population with large constraints

Opt-AiNet + PSO - boundaries: -20/+20		
Time	Iterations	Population
1.53771E+11	36	6717
1.59776E+11	37	6720
1.50568E+11	36	6711
1.3034E+11	33	6714
1.54409E+11	36	6718
1.09566E+11	30	6713
1.63999E+11	38	6714
1.36804E+11	34	6710
1.87362E+11	42	6721
1.67732E+11	39	6721
1.6377E+11	38	6716
1.59295E+11	37	6713
1.52054E+11	36	6713
1.5852E+11	37	6717
1.23425E+11	32	6705
Average	Average	Average
1.51426E+11	36.06666667	6714.866667
F	*P-value*	*F crit*
915.446623	2.31986E-35	3.219942293

Table 12 - Opt-AiNet 20 starting population + 50 with large constraints

As expected the increase in complexity increases significantly the amount of time and generations taken to complete. There is also an increase in the population size which follows the results from the previous tests and does not show much difference between the hybrid algorithm and Opt-AiNet. The average time show that the hybrid algorithm performed 22.8% better with regards to time and 20.6% better in generations needed than Opt-AiNet which means that the evidence from all the tests allow the rejection of the H_0^I null hypothesis and accept the H_1^I alternative hypothesis.

7.2 Result review

The results of the tests reviewed in this chapter supported evidence to accept the H_0 null hypothesis and reject the H_0^I null hypothesis. This leads to the answer of the research questions defined earlier:

- The hybrid algorithm does not improve the average quality of solutions nor makes it any worse.
- The hybrid algorithm improves the time needed to optimize the problem and scales well with increasing complexity.

Valuable insights from the evaluation show that the hybrid approach does not change the population size. Also the efficiency of the hybrid algorithm is greatly influenced by choosing an appropriate size of starting population proportional to the complexity of the problem.

Other insights lead to questions which these tests do not address and are worth investigating in the future. If variables such as starting population size and constrains affect the performance in such a significant way, how would a change in the numerous thresholds and affinities in the algorithm change performance and how impactful would they be.

8. Summery and Conclusion

8.1 Summary

The aim of this research was to investigate if combining Particle Swarm Optimization and Artificial Immune Systems to solve optimization problems is possible and if so to propose a design, implement it into working software and evaluate it against the component algorithms to see if the approach will provide overall increase in the performance and possibly gain a valuable insight of the process.

Looking back at the project's primary goals it is safe to say that all of them have been successfully carried through.

- **Investigate the research done so far in the field and come up with a design for a hybrid algorithm based on PSO and AIS**
  Researching the relevant literature and related work gave valuable knowledge and inspiration allowing the formulation of a design for the hybrid algorithm.
- **Implement the proposed design into a working tool**

  The proposed design was successfully implemented into working software.

- **Ensure that the hybrid algorithm is working correctly by continuous testing while developing**

Manually testing and monitoring the behaviour of the software in the process of development ensured the correct behaviour of the software.

- **Test the performance of the hybrid algorithm against the performance of the component algorithms**
  A set of test carefully picked to collect relevant data was carried out.
- **Evaluate the results of the research**
  An evaluation of the data collected from the tests was carried out which showed that the hybrid algorithm approach offers an overall performance improvement to the optimization process in different levels with respect to choosing adequately the value of relative variables
- **Gain insight on hybrid algorithms**
  The evaluation of the hybrid algorithm showed that the increase in performance levels is heavily impacted by carefully choosing an adequate size for the starting population, allocation of population with respect to Opt-AiNet and PSO and size of the search space (complexity of the problem). This also suggests that other variables like thresholds and affinities may also impact the performance of the algorithm and warrants further research.
- **Lay foundations for future research in the area.**
  The design of the algorithm and working implementation allows the further improvement and additional modifications to the software which will allow carrying out the suggested future research.

8.2 Future work

There is two ways to continue the project further. The first way is to continue improving on the research done in the area by investigating further the circumstances and variables which impact the performance of the hybrid algorithm which may further improve its performance.

The other approach is to try and modify the algorithm into solving different kind of problems like the travelling salesman problem and virus scanning. Another possibility is adding the modification allowing optimizing multi-objective problems which usually adds to the complexity of the problem.

8.3 Conclusion

Being at the boundaries of knowledge in modern days, scientists try to push them as far as possible and any improvement on the tools and methods used to do so, allow them to push them even further. This project tries to identify and fill a small gap in the knowledge in the area of computing and offer some insight and improvement the relatively young field of hybrid algorithms and if possible lay foundations for further research.

In summery this project achieves the primary and secondary goals set, by researching relative work, proposing a viable design and implementing it into working software which allowed gaining knowledge and insight of hybrid algorithms which introduces foundations for future research.

Bibliography

[1] A.Turing, Computing Machinery and Intelligence (1950), p. 460.

[2] "The Nature of Mathematical Programming," Mathematical Programming Glossary, INFORMS Computing Society.

[3] Hall & Hallgrímsson, Strickberger's Evolution 2008, pp. 4–6

[4] Lewontin, R. C. (November 1970). "The Units of Selection" Annual Review of Ecology and Systematics

[5] Masel, Joanna (October 25, 2011). "Genetic drift". Current Biology (Cambridge, MA: Cell Press)

[6] Sawyer, Stanley A.; Parsch, John; Zhang Zhi; Hartl, Daniel L. (Apr 17, 2007). "Prevalence of positive selection among nearly neutral amino acid replacements in Drosophila". Proc. Natl. Acad. Sci. U.S.A. (Washington, D.C.: National Academy of Sciences)

[7] Stuart J. Russell and Peter Norvig 1995, Prentice Hall, Englewood Cliffs, New Jersey 07632

[8] L. J. Fogel , A. J. Owens and M. J. Walsh, Artificial Intelligence Through Simulated Evolution, 1966, John Wiley & Sons

[9] Eiben, A. E. et al (1994). "Genetic algorithms with multi-parent recombination". PPSN III: Proceedings of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature

[10] Ting, Chuan-Kang (2005). "On the Mean Convergence Time of Multi-parent Genetic Algorithms Without Selection". Advances in Artificial Life

[11] Akbari, Ziarati (2010). "A multilevel evolutionary algorithm for optimizing numerical functions" IJIEC 2 (2011)

[12] Beni, G., Wang, J. Swarm Intelligence in Cellular Robotic Systems, Proceed. NATO Advanced Workshop on Robots and Biological Systems, Tuscany, Italy, June 26–30 (1989)

[13] J Greensmith, A Whitbrook, U Aickelin - Handbook of Metaheuristics, 2010 - Springer

[14] de Castro, Leandro N.; Timmis, Jonathan (2002). Artificial Immune Systems: A New Computational Intelligence Approach. Springer

[15] Kelsey, Johnny; Timmis, Jon (2003). "Immune Inspired Somatic Contiguous Hypermutation for Function Optimisation". Genetic and Evolutionary Computation (GECCO 2003). p. 202.

[16] J Greensmith, U Aickelin, J Twycross - Artificial Immune Systems, 2006 - Springer

[17] J Greensmith, U Aickelin (2007) “The Dendritic Cell Algorithm”

[18] F Gonzalez, D Dasgupta, LF Niño - Artificial Immune Systems, 2003 - Springer

[19] J Brownlee - Complex Intelligent Systems 2007

[20] De Castro, L.N and Von Zuben, F. (2001). “aiNET: An Artificial Immune Network for Data Analysis”, in Data Mining: A Heuristic Approach. Abbas, H, Sarker, R and Newton, C (Eds). Idea Group Publishing

[21] De Castro, L.N and Timmis, J. (2002) An Artificial Immune Network for Multimodal Function Optimisation. Proc. Of IEEE World Congress on Evolutionary Computation. Pp. 669-674

[22] Kennedy, J.; Eberhart, R. (1995). "Particle Swarm Optimization". Proceedings of IEEE International Conference on Neural Networks.

[23] Clerc, M. (2012). "Standard Particle Swarm Optimisation" HAL open access archive.

[24] Shi, Y.; Eberhart, R.C. (1998). "A modified particle swarm optimizer". Proceedings of IEEE International Conference on Evolutionary Computation.

[25] Kennedy, J. (1997). "The particle swarm: social adaptation of knowledge". Proceedings of IEEE International Conference on Evolutionary Computation

[26] Kennedy, J.; Eberhart, R.C. (2001). Swarm Intelligence. Morgan Kaufmann.

[27] Poli, R. (2007). "An analysis of publications on particle swarm optimisation applications"

[28] Poli, R. (2008). "Analysis of the publications on the applications of particle swarm optimisation" (PDF). Journal of Artificial Evolution and Applications 2008

[29] Bonyadi, M. R.; Michalewicz, Z. (2016). "Particle swarm optimization for single objective continuous space problems: a review". Evolutionary Computation.

[30] Taherkhani, M.; Safabakhsh, R. (2016). "A novel stability-based adaptive inertia weight for particle swarm optimization". Applied Soft Computing

[31] Shi, Y.; Eberhart, R.C. (1998). "Parameter selection in particle swarm optimization". Proceedings of Evolutionary Programming VII (EP98).

[32] Eberhart, R.C.; Shi, Y. (2000). "Comparing inertia weights and constriction factors in particle swarm optimization".Proceedings of the Congress on Evolutionary Computation

[33] Carlisle, A.; Dozier, G. (2001). "An Off-The-Shelf PSO" (PDF). Proceedings of the Particle Swarm Optimization Workshop.

[34] Clerc, M.; Kennedy, J. (2002). "The particle swarm - explosion, stability, and convergence in a multidimensional complex space". IEEE Transactions on Evolutionary Computation 6 (1)

[35] Trelea, I.C. (2003). "The Particle Swarm Optimization Algorithm: convergence analysis and parameter selection". Information Processing Letters 85 (6)

[36] Zhan, Z-H.; Zhang, J.; Li, Y; Shi, Y-H. (2011). "Orthogonal Learning Particle Swarm Optimization" IEEE Transactions on Evolutionary Computation 15 (6)

[37] Pedersen, M.E.H.; Chipperfield, A.J. (2010). "Simplifying particle swarm optimization". Applied Soft Computing 10 (2)

[38] Tarek A. El-Mihoub, Adrian A. Hopgood, Lars Nolle, Alan Battersby (August 2006), Hybrid Genetic Algorithms: A Review, Engineering Letters

[39] A. A. Hopgood, Intelligent Systems for Engineers and Scientists, 2nd ed: CRC Press, 2001.

[40] E. Talbi, "A Taxonomy of hybrid metaheuristics," Journal of Heuristics, vol. 8, pp. 541–564, 2002.

[41] P. Moscato, "On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms," California Institute of Technology 1989.

[42] G. M. Morris, D. S. Goodsell, R. S. Halliday, R. Huey, W. E. Hart, R. K. Belew, and A. J. Olson, "Automated docking using a Lamarckian genetic algorithm and an empirical binding free energy function," Journal of Computational Chemistry, vol. 19, pp. 1639-1662, 1998.

[43] B. Julstrom, "Comparing Darwinian, Baldwinian, and Lamarckian search in a genetic algorithm for the 4-cycle problem," in the 1999 Genetic and Evolutionary Computation Conference, Late Breaking Papers, S. Brave and A. S. Wu, Eds. Orlando, USA, 1999, pp. 134-138.

[44] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs third ed: Springer-Verlag, 1996.

[45] G. Bilchev and I. C. Parmee, "The ant colony metaphor for searching continuous design spaces," in AISB Workshop on Evolutionary Computing, vol. 993, Lecture Notes In Computer Science, T. C. Fogarty, Ed. Sheffield, UK: Springer Verlag, 1995, pp. 25-39.

[46] D. Orvosh and L. Davis, "Shall we repair? genetic algorithms, combinatorial optimization, and feasibility constraints," in the Fifth International Conference on Genetic Algorithms. Urbana-Champaign, USA: Morgan Kaufmann, 1993, pp. 650

[47] J. Abela, D. Abramson, M. Krishnamoorthy, A. D. Selva, and G. Mills, "Computing optimal schedules for landing aircraft," in the 12th Conference of the Australian Society for Operations Research. Adelaide, 1993, pp. 71-90.

[48] http://www.artificial-immune-systems.org/algorithms.shtml

[49] https://gandhim.wordpress.com/2010/04/04/particle-swarm-optimization-pso-sample-code-using-java/

[50] https://git-scm.com/

[51] https://www.google.com/drive/

[52] Craparo, Robert M. (2007). "Significance level". In Salkind, Neil J. Encyclopedia of Measurement and Statistics 3. Thousand Oaks, CA: SAGE Publications. pp. 889–891

[53] http://archive.ics.uci.edu/ml/

Appendix

This appendix consists of the user manual and the maintenance manual.

A. User Manual

Pre-requisites

1. One of latest Java distributions installed on the machine
2. Input / Output system (command prompt / terminal etc.)

Using the system

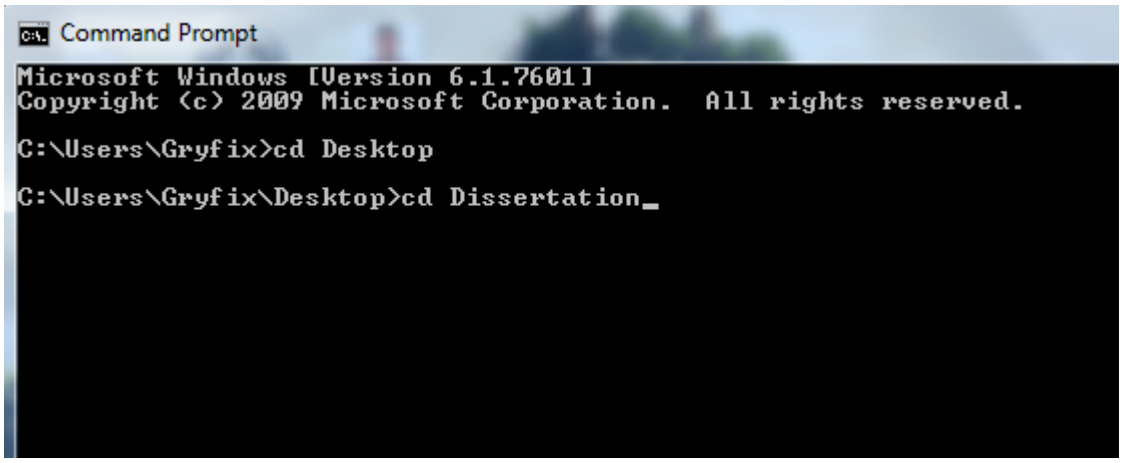
The software can be run through the chosen input / output system since a runnable jar file is provided.

Step 1:

Place runnable_dissertation.jar file and config.txt file in the same directory of choice.

Step 2:

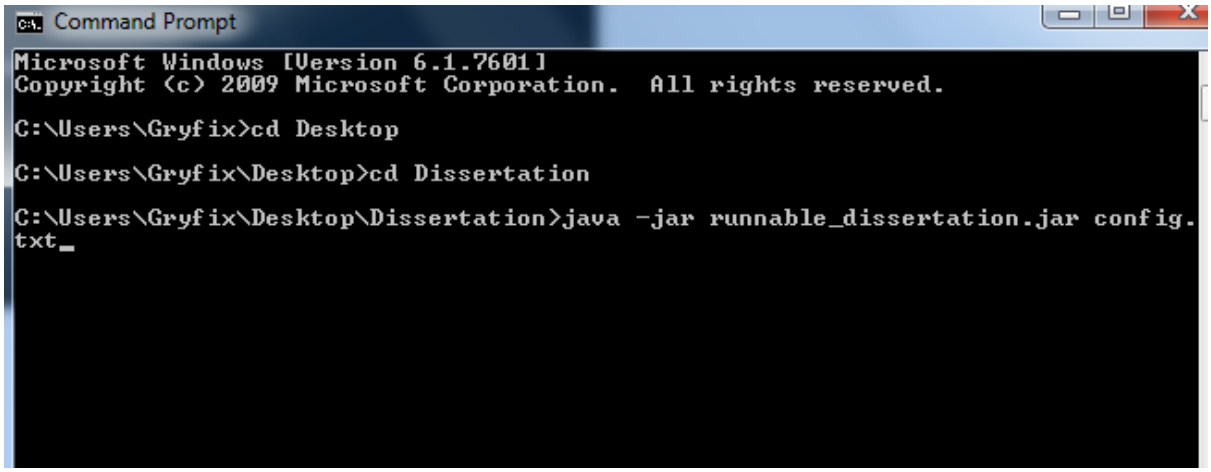
Open console of choice and go to the chosen directory.



Step 3:

To execute the program type “java -jar runnable_dissertation.jar config.txt”.

The first argument of the command -runnable_dissertation.jar is the compiled runnable file and the second argument – config.txt specifies the configuration file which hold different variables.



Step 3:

The software will start execution and will produce output.

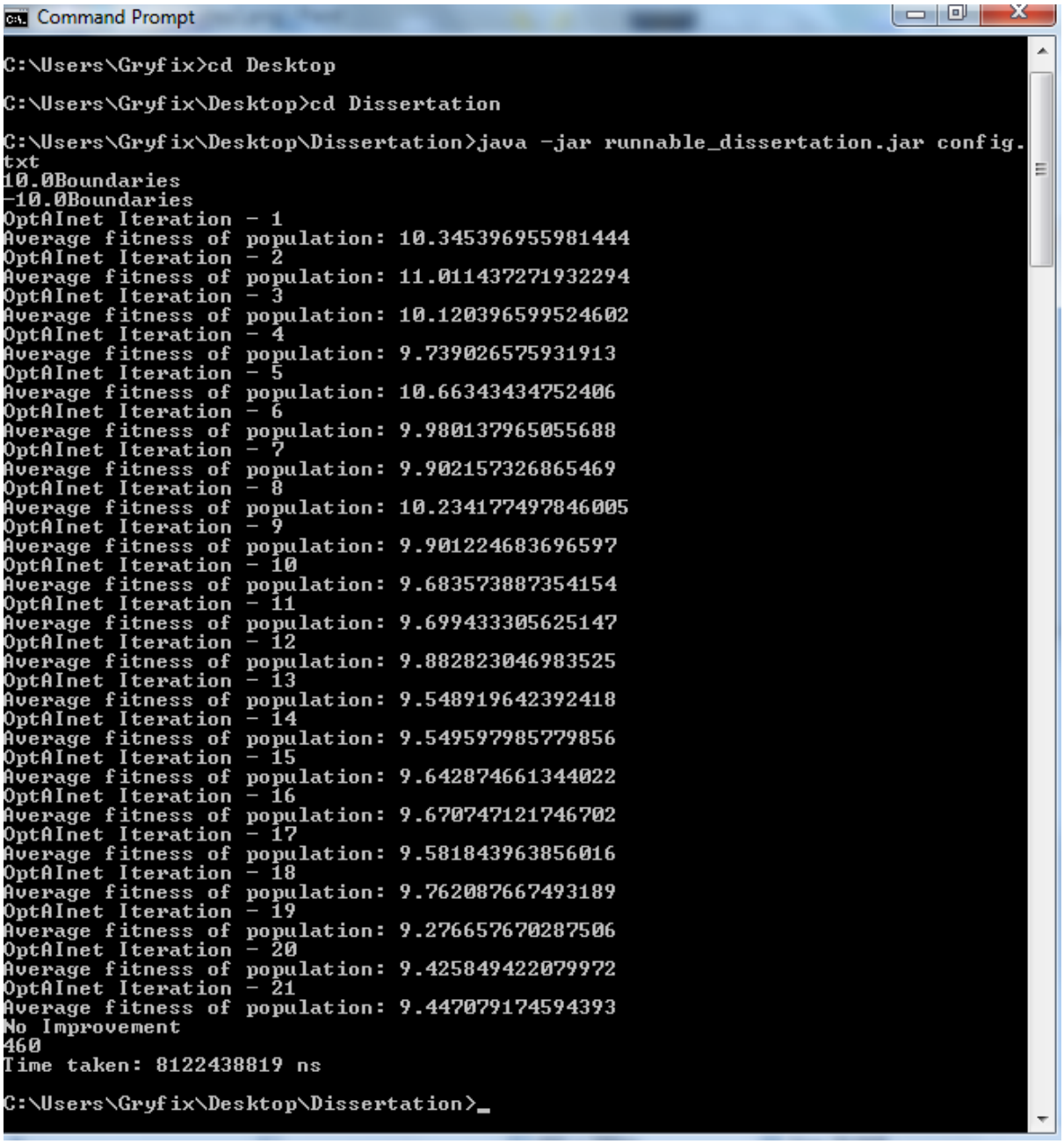
The first two lines of the output shows the boundaries set for the problem.

Then the number of each generation and the average fitness of the population will be shown.

The last 3 lines consist of :

- The condition of termination – usually “No Improvement” which indicates that the algorithm cannot find any better solutions.
- The size of the population
- Time taken for the process to complete measured in nanoseconds.

An example output will look like this:



All libraries used are default, standard for the package of Eclipse IDE so it should be able to be executed by any machine with a recent Java distribution.

B. Maintenance Manual

Software prerequisites

To be able to modify the software a Java IDE like eclipse or NetBeans is required which comes with Java distribution.

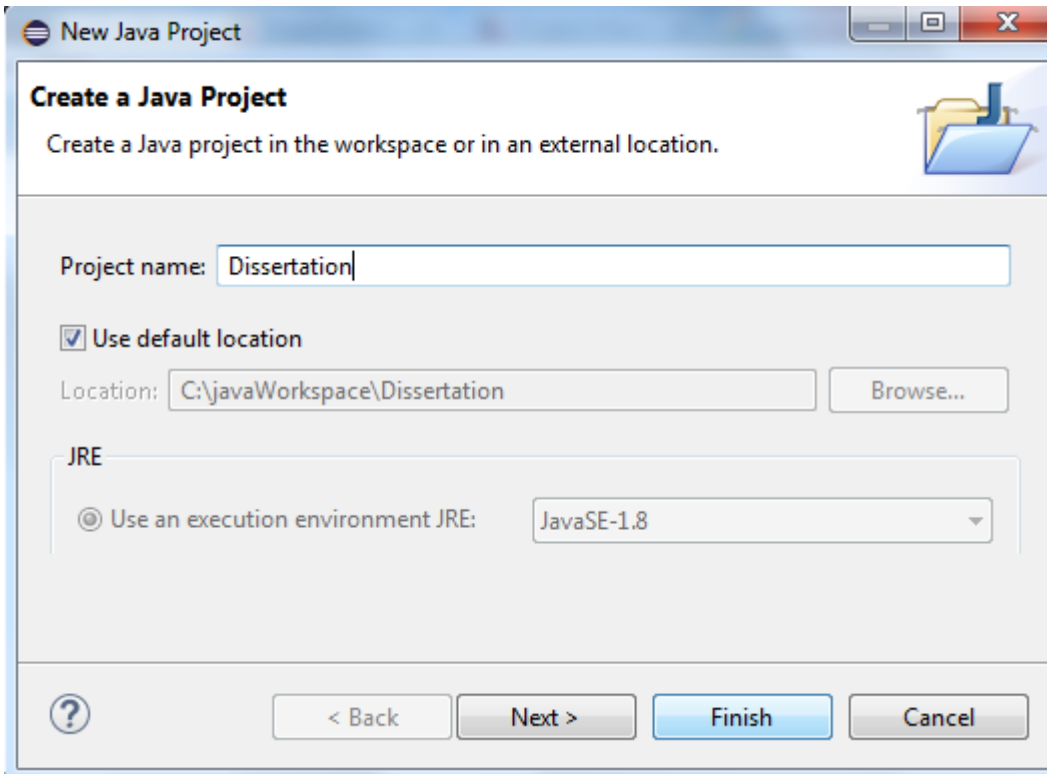
Hardware prerequisites

Approximately 200 kb of disk space, and memory depending on the complexity of the problem being solved. The tests performed in this project were run on a machine that has 2GB RAM.

Installation

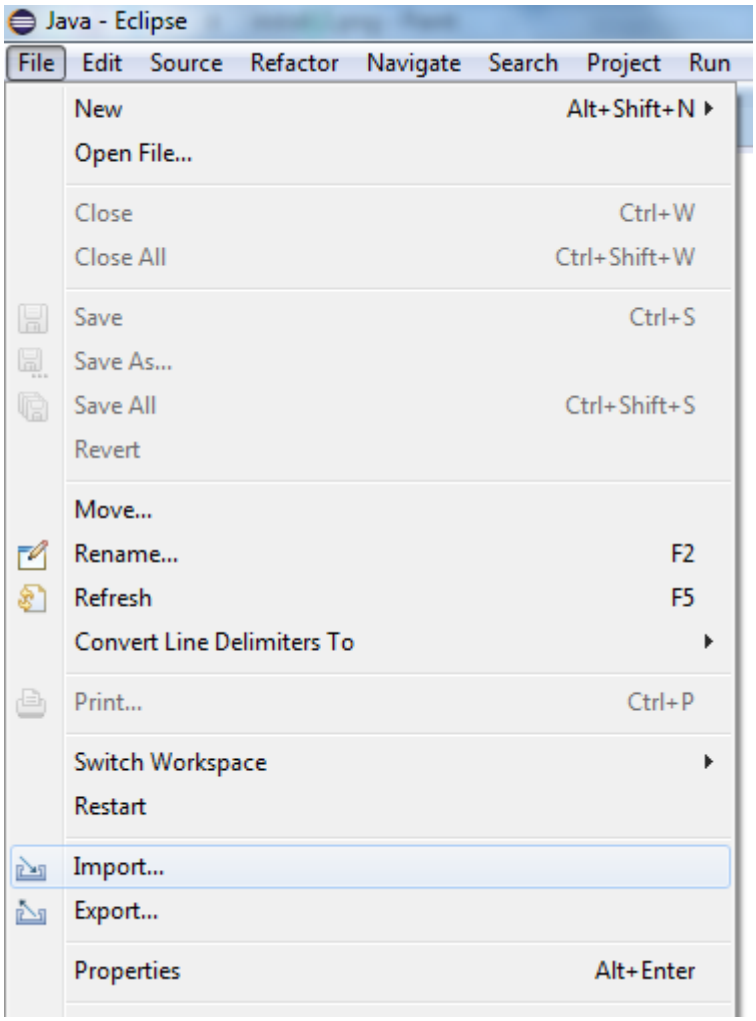
Step 1:

Create a new java project, chose a name and click Finish.



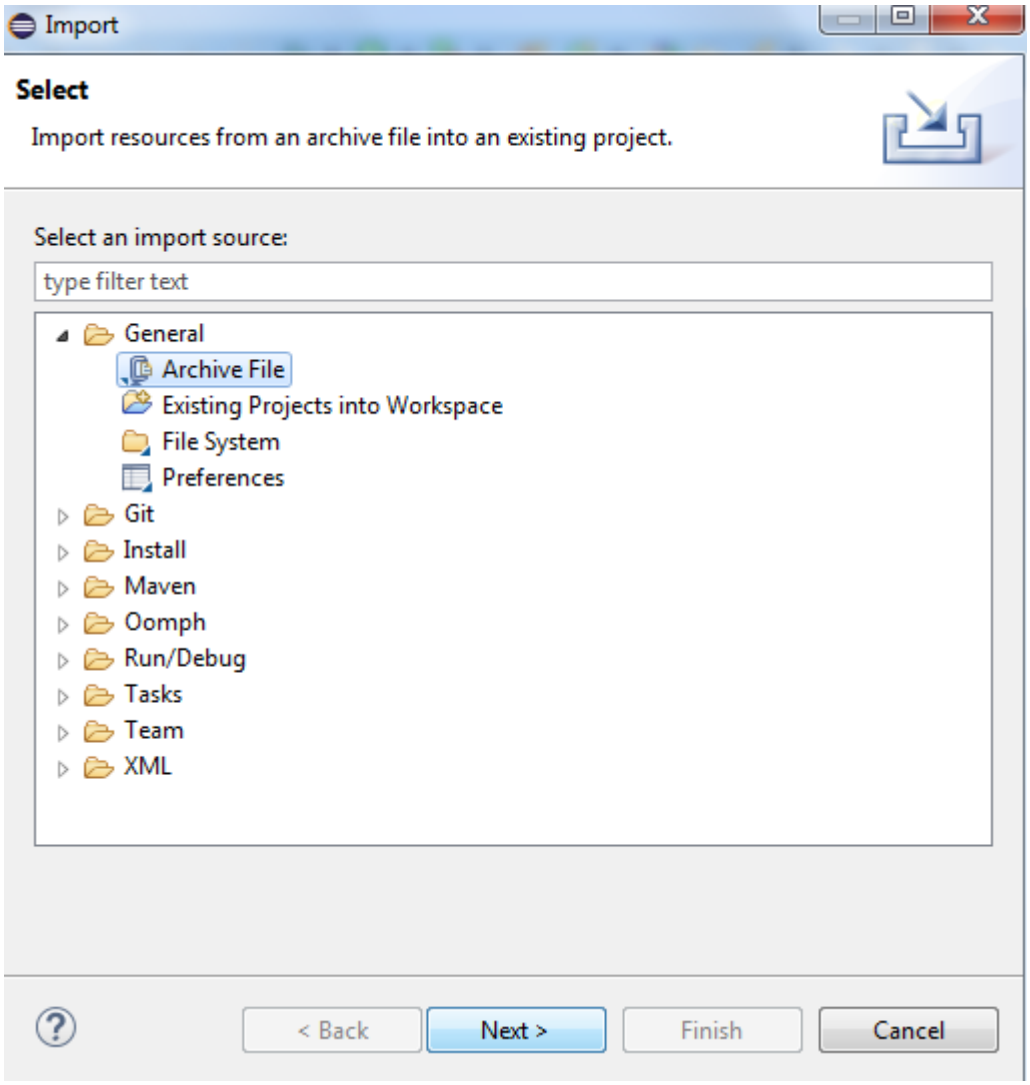
Step 2:

To import the contents of the archive click File > Import.



Step 3:

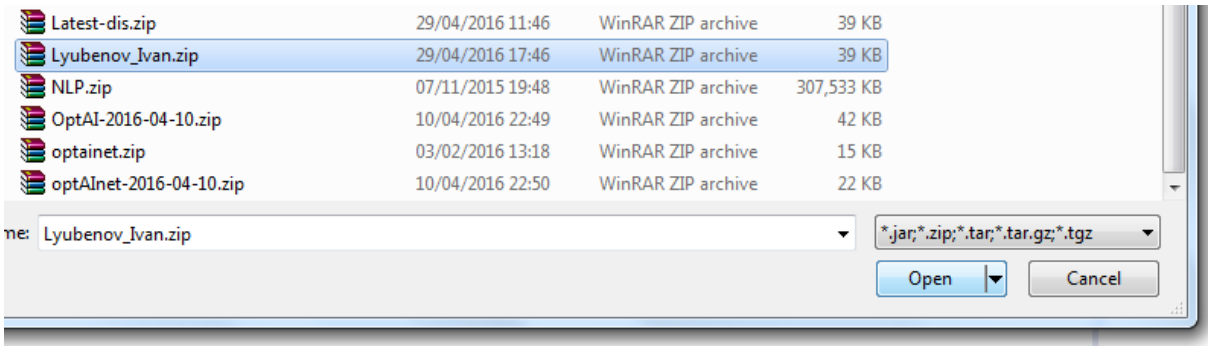
Chose Archive File



Step 4:

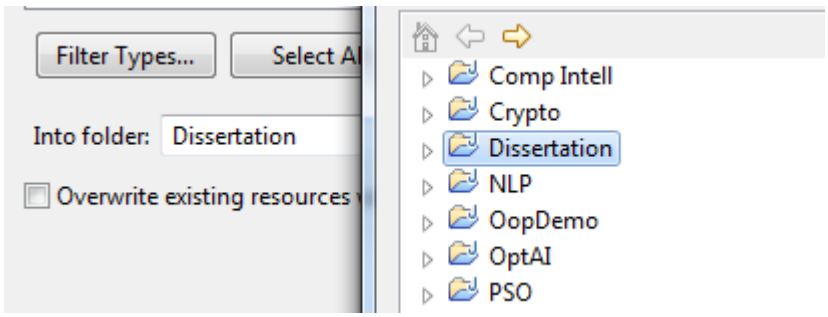
Click the Browse button next to the From archive file input field:

Step 5: Browse to the location of the archive, select it and click Open.



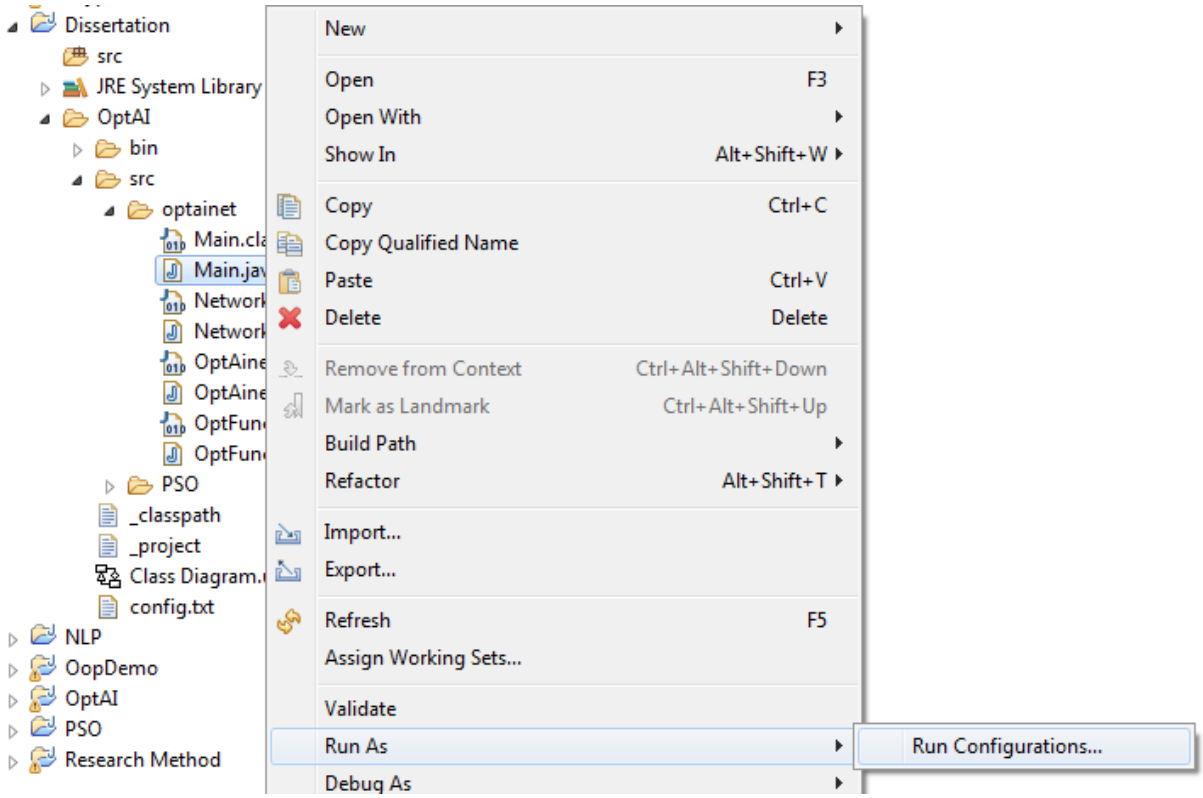
Step 6:

In the Into folder select the project fold created earlier.



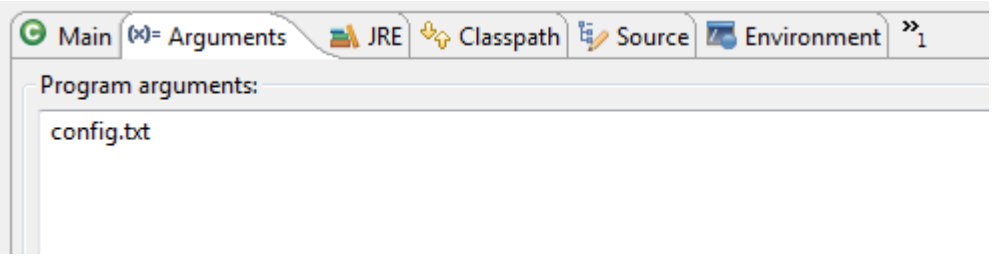
Step 7:

Go to the project folder > OptAi > src > optainet and find the main.java file. Right click it and select Run As > Run Configuration.



Step 8:

Make sure in the Arguments tab the Program arguments field has Config.txt included and if not add it and click Run.



This will compile and run the software.

All the java files can be modified according to the needs. The config.txt file and PSOConstants.java contain the important configuration variables which can be changed and tested. In order to change the optimization function make sure you change it for both Opt-AiNet and PSO respectively in OptFunction.java and ProblemSet.java

Source Code List

File Name	Discription
Main.java	Contains the main file of the software. Contains the Main function.
NetworkCell.java	Contains the NetworkCell class for Opt-AiNet individuals.
OptAinet.java	Contains the main logic and genetic operators.
OptFunction.java	Contains the Opt-AiNet evaluation function.

Location.java	Contains the Location class for PSO particles.
Particle.java	Contains the Particle class for PSO individuals.
ProblemSet.java	Contains the evaluation function of PSO.
PSOConstants.java	Contains Constants for PSO population size and thresholds etc.
PSODriver.java	Used to be the Main method for PSO and now is interface between the two algorithms.
PSOProcess.java	Contains the main logic and genetic operators for the PSO algorithm.
PSOUtility.java	Contains PSOUtility class which is only used when minimizing function.
Velocity.java	Contains the Velocity class for the PSO particles
config.txt	Contains the config variables for Opt-AiNet

Known issues

There are no known issues or bugs of the software.