UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Angle Modulated Population Based Algorithms to solve Binary Problems

by

Gary Pamparà

Submitted in partial fulfillment of the requirements for the degree
Master of Science (Computer Science)
in the Faculty of Engineering, Built Environment and Information Technology
University of Pretoria, Pretoria

April 2012

© University of Pretoria

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Publication data:

Gary Pamparà. Angle Modulated Population Based Algorithms to solve Binary Problems. Master's thesis, University of Pretoria, Department of Computer Science, Pretoria, South Africa, April 2012.

Electronic, hyperlinked versions of this thesis are available online, as Adobe PDF files, at:

`http://cirg.cs.up.ac.za/`

`http://upetd.up.ac.za/UPeTD.htm`

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Angle Modulated Population Based Algorithms to solve Binary Problems

by

Gary Pamparà
E-mail: gpampara@gmail.com

Abstract

Recently, continuous-valued optimization problems have received a great amount of focus, resulting in optimization algorithms which are very efficient within the continuous-valued space. Many optimization problems are, however, defined within the binary-valued problem space. These continuous-valued optimization algorithms can not operate directly on a binary-valued problem representation, without algorithm adaptations because the mathematics used within these algorithms generally fails within a binary problem space. Unfortunately, such adaptations may alter the behavior of the algorithm, potentially degrading the performance of the original continuous-valued optimization algorithm. Additionally, binary representations present complications with respect to increasing problem dimensionality, interdependencies between dimensions, and a loss of precision.

This research investigates the possiblity of applying continuous-valued optimization algorithms to solve binary-valued problems, without requiring algorithm adaptation. This is achieved through the application of a mapping technique, known as angle modulation. Angle modulation effectively addresses most of the problems associated with the use of a binary representation by abstracting a binary problem into a four-dimensional continuous-valued space, from which a binary solution is then obtained. The abstraction is obtained as a bit-generating function produced by a continuous-vaued algorithm. A binary solution is then obtained by sampling the bit-generating function.

This thesis proposes a number of population-based angle-modulated continuous-valued algorithms to solve binary-valued problems. These algorithms are then compared to binary algorithm counterparts, using a suite of benchmark functions. Empirical analysis will show that the angle-modulated continuous-valued algorithms are viable alternatives to binary optimization algorithms.

Keywords: Genetic algorithm, Evolutionary programming, Differential evolution, Particle swarm optimization, Artificial bee colony, Angle modulation, Homomorphous mapping, Binary problem optimization.

Supervisor	**:**	Prof. A. P. Engelbrecht
Department	**:**	Department of Computer Science
Degree	**:**	Master of Science

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Acknowledgements

Sincere thanks and appreciation to the following people for their assistance during this research:

- Professor AP Engelbrecht, my supervisor, for his guidance on this work and providing insights together with the many discussions on this research. His encouragement and patience will not be forgotten – I am forever grateful.
- My family, especially my parents Elvio and Suzette, for the never ending support and listening ability which aided the research development.
- All my friends who encouraged me through the entire process.
- All the CIlib developers and CIRG members, for discussing all ideas and problems, together with finding the appropriate solutions.
- Karen, my wife, for her unending support and love. It is thanks to her compasion, support and understanding that this thesis could be completed.

Contents

List of Figures **vi**

List of Algorithms **viii**

List of Tables **ix**

1 Introduction **1**

1.1 Motivation 1

1.2 Objectives 3

1.3 Contributions 4

1.4 Thesis Outline 5

2 Population Based Continuous Algorithms **7**

2.1 Optimization 8

2.2 Evolutionary Computation 9

2.2.1 Generic Evolutionary Algorithm 10

2.2.2 Genetic Algorithms 13

2.2.3 Evolutionary Programming 19

2.2.4 Differential Evolution 24

2.3 Swarm Intelligence 28

2.3.1 Particle Swarm Optimization 29

2.3.2 Artificial Bee Colony Optimization 37

2.4 Summary 40

3 Population Based Binary Algorithms **41**

3.1 Introduction 41

3.2 Binary representation . . . 42
3.2.1 Binary variable representations . . . 42
3.2.2 Non-binary variable representations . . . 44
3.3 Problems with binary representations . . . 44
3.3.1 Hamming cliffs . . . 45
3.3.2 Curse of dimensionality . . . 45
3.3.3 Loss of precision . . . 46
3.3.4 Bit string interdependencies . . . 46
3.3.5 Computational complexity . . . 47
3.4 Binary algorithms . . . 47
3.4.1 Canonical Genetic Algorithm . . . 47
3.4.2 Evolutionary programming . . . 48
3.5 Continuous algorithms . . . 49
3.5.1 Algorithmic redefinition . . . 49
3.5.2 Binary particle swarm optimization . . . 50
3.5.3 Differential evolution . . . 52
3.5.4 Artificial bee colony . . . 55
3.6 Summary . . . 57

4 Angle Modulation **58**
4.1 Introduction . . . 58
4.2 Signal processing . . . 59
4.3 Angle modulation . . . 60
4.4 Angle modulated population-based algorithms . . . 62
4.4.1 Angle modulated genetic algorithm . . . 63
4.4.2 Angle modulated evolutionary programming . . . 64
4.4.3 Angle modulated differential evolution . . . 64
4.4.4 Angle modulated particle swarm optimization . . . 65
4.4.5 Angle modulated artificial bee colony . . . 65
4.4.6 Angle modulated benefits . . . 65
4.5 Summary . . . 66

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

5 Experimental Approach **67**
5.1 Introduction 67
5.2 Benchmark functions 67
5.2.1 Binary mapped continuous functions 68
5.2.2 Pure binary functions 68
5.2.3 Matching random bit strings 78
5.3 Performance criteria 78
5.4 Algorithm comparisons 79
5.4.1 Genetic algorithm 79
5.4.2 Evolutionary programming 79
5.4.3 Differential evolution 80
5.4.4 Particle swarm optimization 80
5.4.5 Artificial bee colony 81
5.5 Statistical procedure 81
5.6 Summary 82

6 Results and Comparisons **83**
6.1 Introduction 83
6.2 Genetic algorithm results 83
6.2.1 Binary mapped continuous-valued problems 84
6.2.2 Binary-valued problems 86
6.2.3 Overall outcome 89
6.3 Evolutionary programming results 89
6.3.1 Binary mapped continuous-valued problems 90
6.3.2 Binary-valued problems 91
6.3.3 Overall outcome 95
6.4 Differential evolution results 95
6.4.1 Binary-mapped continuous-valued problems 95
6.4.2 Binary-valued problems 99
6.4.3 Overall outcome 102
6.5 Particle swarm optimization results 103
6.5.1 Binary-mapped continuous-valued problems 103
6.5.2 Binary-valued problems 105

6.5.3 Overall outcome . . . 108
6.6 Artificial bee colony results . . . 108
6.6.1 Binary-mapped continuous-valued problems . . . 108
6.6.2 Binary-valued problems . . . 110
6.6.3 Overall outcome . . . 118
6.7 Comparison of angle modulation algorithms to binary algorithm variants 118
6.7.1 Continuous-valued problems . . . 119
6.7.2 Binary-valued problems . . . 120
6.7.3 Summary of ranking results . . . 122
6.8 Scalability . . . 123
6.9 Summary . . . 124

7 Conclusions **125**
7.1 Summary of Conclusions . . . 125
7.2 Future work . . . 128

Bibliography **130**

A List of symbols **148**

B List of acronyms **150**

C Derived Publications **153**

D Binary encoding of continuous values **154**

E Simulation specifications **156**
E.1 Genetic algorithm . . . 156
E.1.1 Continuous simulations . . . 156
E.1.2 Binary simulations . . . 161
E.2 Evolutionary programming . . . 167
E.2.1 Continuous simulations . . . 167
E.2.2 Binary simulations . . . 172
E.3 Differential evolution . . . 178
E.3.1 Continuous simulations . . . 178

E.3.2 Binary simulations . . . 186
E.4 Particle swarm optimization . . . 196
E.4.1 Continuous simulations . . . 196
E.4.2 Binary simulations . . . 201
E.5 Artificial bee colony . . . 207
E.5.1 Continuous simulations . . . 207
E.5.2 Binary simulations . . . 215
E.6 Additional Java sources . . . 225
E.6.1 ForwardingFunction.java . . . 225
E.6.2 BinaryConvertedFitnessCalculator.java . . . 226
E.6.3 NormalizedFitnessCalculator.java . . . 227
E.6.4 IteratedPrisonersDilemma.java . . . 228
E.7 Summary . . . 230

List of Figures

2.1 Crossover operators for discrete recombination 16
2.2 Gaussian and Cauchy distributions 22
2.3 PSO neighborhood topologies 34

4.1 Angle modulation generating function $g(x)$ with default parameters . . . 61
4.2 Allocation of bit values from generating function. 63

5.1 Valid moves for Knight chess piece 70
5.2 8×8 Knight's Coverage solution 72
5.3 8×8 n-Queens solution 73

6.1 GA algorithm performance for binary-mapped continuous problems . . . 85
6.2 AMGA bit string generating function for f_6 86
6.3 GA algorithm performance for binary minimization problems 87
6.4 AMGA generating function for KT_4 89
6.5 GA algorithm performance for binary maximization problems 90
6.6 AMGA diversity for the IPD problem. 90
6.7 EP algorithm performance for binary-mapped continuous problems . . . 92
6.8 EP diversity for selected binary-mapped problems 93
6.9 EP algorithm performance for binary minimization problems 94
6.10 EP algorithm performance for binary maximization problems 96
6.11 AMDE bit string generating function for f_8 97
6.12 DE algorithm performance for binary-mapped continuous problems . . . 98
6.13 DE algorithm diversity for binary-mapped problems 99
6.14 DE algorithm performance for binary minimization problems 100
6.15 DE algorithm performance for binary maximization problems 101

6.16 AMDE diversity and generating function for KT_8 101
6.17 AMPSO generating function for binary-mapped problem f_2 105
6.18 PSO algorithm performance for binary-mapped problems 106
6.19 PSO algorithm diversty for selected binary-mapped problems 106
6.20 AMPSO bit string generating function for KC_6 108
6.21 PSO algorithm performance for binary minimization problems 109
6.22 PSO algorithm performance for binary maximization problems 110
6.23 ABC algorithm performance for binary-mapped problems 112
6.24 ABC diversity for selected binary-mapped problems 113
6.25 Generating functions for selected divergent AMABC problems 113
6.26 ABC algorithm diversity for selected binary minimization problems . . . 115
6.27 ABC algorithm performance for binary minimization problems 116
6.28 ABC algorithm performance for binary maximization problems 118
6.29 ABC algorithm diversity for selected binary maximization problems . . . 119
6.30 Accuracy of matched random bit strings 124
6.31 Rate of deterioration compared to an increase in dimension 124

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

List of Algorithms

1 Generic evolutionary algorithm 10
2 Genetic algorithm 14
3 Uniform crossover 15
4 Generic evolutionary programming 20
5 Differential evolution 25
6 Canonical particle swarm optimization 31
7 Artificial bee colony 38
8 Binary differential evolution. 53
9 Normalization differential evolution. 55
10 Binary artificial bee colony 56
11 Normalization artificial bee colony 57
12 Generic Angle Modulated Population-based Algorithm 64
13 Proposed fitness evaluation for the knight's coverage problem 72
14 Proposed fitness function for n-Queens problem 74
15 Random bit string matching 78

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

List of Tables

3.1 Binary and Gray coding 44

5.1 Continuous-valued benchmark functions 69
5.2 Continuous-valued benchmark function equations 69
5.3 IPD payoff matrix 77

6.1 GA algorithm results for binary-mapped minimization functions 84
6.2 Binary minimization function results for GA algorithms 87
6.3 Binary maximization function results for GA algorithms 88
6.4 EP algorithm results for binary-mapped minimization functions 91
6.5 Binary minimization function results EP algorithms 93
6.6 Binary maximization function results for EP algorithms 95
6.7 DE algorithm results for binary-mapped minimization functions 97
6.8 Binary minimization function results for DE algorithms 102
6.9 Binary maximization function results for DE algorithms 103
6.10 PSO algorithm results for binary-mapped minimization functions 104
6.11 Binary minimization function results for PSO algorithms 107
6.12 Binary maximization function results for PSO algorithms 109
6.13 ABC algorithm results for binary-mapped minimization functions 111
6.14 Binary minimization function results for ABC algorithms 114
6.15 Binary maximization function results for ABC algorithms 117
6.16 AM and non-AM algorithm rankings for continuous-valued problems . . . 120
6.17 AM and non-AM algorithm rankings for binary minimization problems . 121
6.18 AM and non-AM algorithm rankings for binary maximization problems . 122

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Chapter 1

Introduction

This thesis proposes a technique called angle modulation which is used to solve binary-valued optimization problems in continuous-valued space.

1.1 Motivation

Numerous real-world problems have binary-valued solutions or can be formulated such that solutions can be represented using a binary-valued representation. While good optimization algorithms, which cater exclusively for binary-valued problem domains have been developed to solve binary-valued optimization problems[1], transformation of continuous-valued representations into binary-valued representations and solving the problem with a binary optimization algorithm introduces a number of concerns [23]:

- **Hamming cliffs**: Conversion from continuous-valued representations to binary-valued representations may introduce Hamming cliffs. The consequence of a Hamming cliff is a large Hamming distance between the binary representations of consecutive continuous values. The implication is that many bits need to be changed in the binary representation of a continuous value to obtain the succeeding continuous value.

- **Loss of precision**: A continuous value is represented by a finite number of bits. Consequently, precision is lost in the conversion of the continuous value into a

[1]This thesis refers to these algorithms as binary algorithms

binary representation. To increase the precision, the number of bits has to be increased.

- **Curse of dimensionality**: A consequence of increasing the number of bits is that the dimension of the binary-valued search space increases linearly with the number of bits.

- **Search space discretization**: Representation of continuous values as bit strings discretizes the continuous-valued space, loosing important characteristics of the continuous-valued space. Discretization reduces the number of candidate solutions from infinite to finite.

Efficient population-based algorithms have been developed to solve binary-valued optimization problems, for example genetic algorithms (GAs) and evolutionary programming (EP). However, these binary algorithms do not address the concerns above.

On the other hand, a large number of efficient optimization algorithms exist for solving continuous-value optimization problems[2] [79, 82, 143, 147, 155]. Many of these algorithms can not be applied to binary-valued optimization problems without adaptations to the operators of the algorithms themselves. For example, the original particle swarm optimization (PSO) algorithm [82] and many of its variations can not be applied to binary-valued representations. However, the PSO has been adapted to solve binary-valued problems. The resulting binary PSO [83], referred to as the binary particle swarm optimization (BinPSO), unfortunately changed the basic principles of PSO (refer to Section 2.3.1). Similarly, adaptations to differential evolution (DE) and artificial bee colony (ABC) cause changes to the underlying principles of these algorithms.

Population-based binary algorithms have been adapted to solve continuous-valued optimization problems, for example genetic algorithms [34, 70], evolutionary programming [49, 50], and ant colony optimization [10, 11, 93]. The continuous algorithms and variations of these specific continuous representations have shown to be very efficient within continuous-valued spaces. This prompted researchers to apply them to also solve binary-valued problems. However, this requires special operators to be defined specifically for binary representations, but does not address the representation concerns discussed above.

[2]This thesis refers to these algorithms as continuous algorithms

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

This thesis proposes an approach to solve binary-valued problems using continuous algorithms without the need to adapt these algorithms. This is achieved by generating a function within continuous space from which a bit string solution is obtained. The obtained function is sampled at equally spaced intervals to determine a bit string solution which is then evaluated within the original binary-valued space. This process is based on the concept of angle modulation (AM) as used in signal processing [19]. The result is a set of angle modulated population-based algorithms with the ability to solve binary-valued problems within continuous-valued space.

Using these angle modulated algorithms, the issues discussed above are addressed as follows:

- **Representation**: Hamming cliffs are no longer an issue as solutions are found in continuous-valued space and not binary-valued space (the search process does not make use of bit based operations). The curse of dimensionality problem is solved because solutions are always found within a four-dimensional continuous-valued space irrespective of the dimensionality of the binary-valued search space (refer to Chapter 4). The loss of precision and the search space discretization are no longer relevant because the angle modulated algorithms are used to solve binary-valued problems and not continuous-valued problems.

- **Algorithm adaptations**: No modifications of the continuous algorithms are made to produce the angle modulated algorithms. Therefore, the principles of the original algorithms are maintained and no special operations are required to solve binary-valued problems (refer to Chapter 4 for a more detailed clarification).

1.2 Objectives

The main objective of this thesis is to introduce angle modulation (AM) as an approach to apply continuous algorithms to solve binary-valued problems without searching within binary-valued space nor adapting the original continuous algorithms. This objective is obtained by the following sub-objectives:

- To highlight the problems associated with performing optimization in binary-valued problem spaces, and problems with adapting continuous algorithms to be applied to binary-valued spaces.

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

- To introduce that angle modulation is a viable method for solving binary-valued optimization problems using algorithms developed for continuous-valued spaces.
- To apply AM to a collection of population-based continuous algorithms, resulting in the angle modulated genetic algorithm (AMGA), angle modulated evolutionary programming (AMEP), angle modulated differential evolution (AMDE), angle modulated particle swarm optimization (AMPSO), and angle modulated artificial bee colony (AMABC), thereby showing that angle modulation (AM) is a generic approach applicable to any population-based continuous algorithm.
- To compare the performance of angle modulated algorithms to the binary versions of the corresponding continuous algorithms.
- To identify the best performing AM algorithm.
- To study the scalability of AM for high-dimensional binary-valued problems using the best performing angle modulated algorithm.

1.3 Contributions

This research makes the following contributions to knowledge in the fields of optimization, evolutionary computation and swarm intelligence:

- Identification of issues related to binary representations used to solve optimization problems within binary-valued search spaces, and related to adaptation of continuous algorithms to solve binary-valued problems.
- A novel technique, angle modulation, to solve binary-valued problems using continuous algorithms without adapting these algorithms.
- It is shown that the angle modulated algorithms solve the issues related to binary representation and algorithm adaptations.
- Development of adaptations to continuous algorithms to solve binary-valued problems.

- Characterization of the proposed angle modulated algorithms in comparison with binary algorithms.

Additionally, the following artifacts are produced:

- Angle modulated algorithm variations, namely angle modulated genetic algorithm (AMGA), angle modulated evolutionary programming (AMEP), angle modulated differential evolution (AMDE), angle modulated particle swarm optimization (AMPSO) and angle modulated artificial bee colony (AMABC).
- Adaptations of the continuous algorithms, genetic algorithm (GA), evolutionary programming (EP), differential evolution (DE), particle swarm optimization (PSO) and artificial bee colony (ABC), to directly solve binary-valued problems.
- Expansion of the computational intelligence library (CIlib) [27] to include all of these new algorithms, as well as binary versions of these algorithms.

1.4 Thesis Outline

The remainder of this thesis is organized as follows:

- **Chapter 2** focuses on existing population-based continuous algorithms within the evolutionary computation and swarm intelligence paradigms.
- **Chapter 3** discusses issues related to binary representations, describes existing population-based binary algorithms, and adaptations to population-based continuous algorithms for application to binary-valued optimization problems. New adaptations of continuous algorithms are also presented in this chapter.
- **Chapter 4** proposes angle modulation as an approach to apply continuous algorithms to solve binary-valued problems.
- **Chapter 5** describes the experimental approach followed to analyze the performance of the proposed angle modulated algorithms and binary algorithms. The chapter also lists benchmark functions used.
- **Chapter 6** discusses the obtained results, and performs a scalability study.

- **Chapter 7** contains remarks and conclusions together with suggestions for future work building from the research completed within this thesis.

Listed within the bibliography of this thesis are all cited sources. Where possible, all online citations include a digital object identifier (DOI)[3] which references a persistent electronic document.

The appendices provide supplementary resources:

- **Appendix A** provides a list of symbols used in this thesis.
- **Appendix B** defines the acronyms within this thesis.
- **Appendix C** lists the publications derived from this work.
- **Appendix D** defines the binary encoding procedure for continuous-valued variables within binary space as used by Spears [139].
- **Appendix E** contains copies of the simulation specifications for the CIlib library, used to generate the experimental data, as well as additional Java classes required for the simulations.

[3]A DOI is a unique alphanumeric identification string for a digital object, providing a persistent link to it. It represents a permanent URL kept the same way a domain name is. For further information on DOI, see `http://www.doi.org`

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Chapter 2

Population Based Continuous Algorithms

Many hands make light work.

John Heywood

Optimization literature provides a large number of approaches to find solutions to optimization problems. Classical approaches from the operational research community use a single search point, or candidate solution, and adapts this solution either deterministically or stochastically to find a new, hopefully better solution [117]. The computational intelligence community has provided more efficient search algorithms which maintain multiple candidate solutions. An optimum is then searched for from multiple locations in the search space, where candidate solutions are usually stochastically updated, using global information about the search space as obtained from the candidate solutions. This thesis refers to these algorithms as population-based algorithms (PBAs).

The main objective of this chapter is to provide an overview of the population-based algorithms used within this thesis as applied to unconstrained optimization. The remainder of this chapter provides an overview of optimization and formally defines unconstrained optimization in Section 2.1. Sections 2.2 and 2.3 respectively discuss evolutionary and swarm-based algorithms relevant to this thesis. The chapter concludes with a summary in Section 2.4.

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

2.1 Optimization

An unconstrained optimization problem is formally defined as (assuming minimization):

$$\begin{array}{rc} \text{minimize} & f(\mathbf{x}) \\ \text{subject to} & \mathbf{x} \in [x_{min}, x_{max}]^{n_x} \end{array}$$

where

$$f : S \to \mathbb{R} \tag{2.1}$$

is the objective function being minimized, $[x_{min}, x_{max}]^{n_x}$ defines the domain of the objective function, $\mathbf{x}_{min}$ is the lower bound and $\mathbf{x}_{max}$ is the upper bound of the domain, n_x refers to the dimension of the objective function, and $\mathbf{x}$ is a vector of decision variables, x_j for $j = 1, \ldots, n_x$.

The purpose of an optimization algorithm is to find a global minimum, $\mathbf{x}^* \in S$, such that for all $\mathbf{y} \in S$:

$$f(\mathbf{x}^*) < f(\mathbf{y}) \tag{2.2}$$

For convex search spaces equation (2.2) will always be valid. For problem spaces that are not convex, a *local minimum*, $\mathbf{x}^*_N$, is defined as a point within S such that, for some value of $\delta > 0$ and for all values of $\mathbf{x}$ in a region around $\mathbf{x}^*_N$, the following hold:

$$||\mathbf{x} - \mathbf{x}^*_N|| \leq \delta \tag{2.3}$$

$$f(\mathbf{x}^*_N) \leq f(\mathbf{x}) \tag{2.4}$$

Maximization problems can be expressed in terms of minimization by realizing that

$$\text{maximize } f(\mathbf{x}) \equiv \text{minimize } (-f(\mathbf{x})) \tag{2.5}$$

For continuous-valued decision variables, i.e. $x_j \in \mathbb{R}$, the search space, $S \subseteq \mathbb{R}^{n_x}$, is continuous-valued. If the decision variables are integers, i.e. $x_j \in \mathbb{Z}$, then $S \subseteq \mathbb{Z}^{n_x}$. For discrete-valued decision variables, each decision variable have a finite domain, i.e. each $x_j \in dom(x_j)$.

The focus of this research is on binary-valued decision variables where $x_j \in \{0, 1\}$, in which case $S = \{0, 1\}^{n_x}$. This thesis narrows focus further to consider only single-objective, unconstrained, static optimization problems. To solve such binary-valued

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

optimization problems, this thesis considers genetic algorithms, evolutionary programming, differential evolution, particle swarm optimization, and artificial bee colony as described in the sections that follow.

2.2 Evolutionary Computation

Evolution is the process of iteratively improving over time within a constantly changing environment with limited resources, whilst in competition for survival with others. Evolution is therefore an optimization process itself, with the objective of producing an optimal candidate solution.

Biological evolution concerns the changes that organisms (species) undergo over time to improve their chance of survival. There are two distinct theories of evolution that are still, to this day, in full debate [58, 94, 109, 160]:

- **Lamarckism**, regarded as one of the first theories, formalized by Jean-Baptiste Lamarck (1744-1829), focuses on heredity. During the lifetime of an organism, the organism acquires characteristics and these are then passed on to its offspring. The process then continues with the offspring passing their acquired characteristics on to the next generation of offspring. Lamarckism advocates that acquired adaptations are persistent solely on the concept of use and disuse. Over time, organisms continually exercise beneficial characteristics and disuse those which are not. As a result, unused characteristics would eventually disappear from future generations of offspring.

- **Darwinism**: Charles Darwin (1809-1882) proposed a popular theory which focuses on the principle of *natural selection* [33]. Independently from Darwin, Alfred Wallace formulated a similar, slightly different theory than that of Darwin [89]. The philosopher Herbert Spencer summed up the work of Darwin in the phrase "survival of the fittest" and was later taken to be emblematic of Darwin's work. Darwinian evolution focuses on the genetic constitution of an organism, with genes determining the inherited characteristics of the organism. Genes, collectively the genome, define an organism's genotype. The offspring's genes may override the characteristics provided by a parent organism, potentially resulting in different

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

physical characteristics manifesting within the offspring. With reference to humans, such an offspring overriding behavior is the "brown-eye trait" [144] where the offspring may develop an iris pigmentation that is different from that of the parent.

Additionally, Darwin proposed that offspring acquire minor differences, compared to the parent organisms. The differences within the genotype of the offspring, called mutations, may or may not be beneficial to the offspring, thereby increasing or decreasing the survival chance of the offspring organism. Gregor Mendel (1822-1884), father of the field of genetics, substantiated Darwin's theories about the transfer of genotypic information from parent to offspring [132].

Evolutionary computation (EC) [5, 48] refers to computer-based problem solving techniques based on evolution theory. EC is a paradigm that includes algorithms such as GAs, EP, evolutionary strategiess (ESs) and DE. Section 2.2.1 discusses a generic evolutionary algorithm (EA), and describes the main components present within EAs. Sections 2.2.2 to 2.2.4 respectively discuss GA, EP, and DE which are the only EAs that fall in the scope of this thesis.

2.2.1 Generic Evolutionary Algorithm

An EA is an algorithm inspired from the biological evolutionary processes of mutation, reproduction, recombination and selection. An EA maintains a collection, or population, of candidate solutions also known as individuals. Algorithm 1 provides pseudo-code for a generic EA.

Algorithm 1 Generic evolutionary algorithm

```
Let $t = 0$ be the generation counter
Create and initialize an $n_x$-dimensional population, $P(0)$
repeat
    Evaluate the fitness, $f(\mathbf{x}_i(t))$, of each individual, $\mathbf{x}_i(t)$, in the population, $P(t)$
    Perform reproduction operators to create offspring
    Select population $P(t+1)$ from the current population and created offspring
until stopping condition(s) satisfied
```

The components of an EA include:

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Initialization

Candidate solutions are uniformly initialized within the domain of the objective function. To ensure uniform coverage of the search space, it is advised that a robust pseudo-random number generator be used, such as the Mersenne Twister [101].

The population size, n_s, directly influences the computational complexity of the search process. Large values of n_s require a larger computational cost per generation, but a good solution might be found in less generations than for small values of n_s. Smaller population sizes are computationally less expensive per generation. Additionally, larger populations facilitate more exploration than small populations because of better representation of the search space.

Fitness Evaluation

A *fitness function* quantifies the quality of candidate solutions to a given optimization problem. In optimization, the fitness function is usually the objective function being minimized.

The fitness function drives evolutionary operators such as selection and reproduction.

Selection

Selection ensures that desirable characteristics are not lost during the evolutionary process. Selection is used to determine which individuals survive to the next generation and to determine which individuals take part in reproduction.

A consequence of selection is that populations become homogeneous, i.e. individuals converge to the same solution, prematurely. This is referred to as selective pressure [61, 21]. A large selective pressure favors fit individuals within the population, decreasing the population diversity from generation to generation, limiting the exploration within the search space.

The most frequently used selection operators include:

- **Random selection**, where each candidate solution within the population has the exact same probability, $\frac{1}{n_s}$, of being selected, where n_s is the size of the population. The fitness of individuals is not taken into account for the selection process, resulting in the lowest selective pressure.

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

- **Proportional selection**, which relies on the creation of a fitness probability distribution. Sampling this distribution determines which candidate solution will be selected. The selection probability of each candidate solution is calculated as

  $$p_i = \frac{f_\Upsilon(\mathbf{x}_i(t))}{\sum_{n=1}^{n_s} f_\Upsilon(\mathbf{x}_n(t))} \tag{2.6}$$

  where f_Υ is a function that scales the fitness value of the candidate solution. Scaling of the fitness value is required to ensure that the fitness of each candidate solution returns an unbiased probability. The scaling function, assuming maximization, is

  $$f_\Upsilon(\mathbf{x}_i(t)) = \frac{1}{1 + f_{\max}(t) - f(\mathbf{x}_i(t))} \tag{2.7}$$

  where $f_{\max}(t)$ is the largest fitness value at time step t. For minimization fitness values are scaled using

  $$f_\Upsilon(\mathbf{x}_i(t)) = \frac{1}{1 + f(\mathbf{x}_i(t)) - f_{\min}(t)} \tag{2.8}$$

  with $f_{\min}(t)$ being the smallest fitness value at time step t.

  Because proportionate selection biases to the most fit individuals, the selective pressure of the operator is very high.

  A sampling method, e.g. roulette wheel selection, is applied on these probabilities. Roulette wheel selection effectively places the scaled proportions onto an imaginary "wheel", with each scaled proportion occupying a slice of the wheel. The wheel is then spun with a marker point identifying the selected candidate solution. An alternative sampling approach is to use stochastic universal sampling [6].

- **Tournament selection**, which consists of a two stage selection process. A pool of n_t competitors is selected randomly from the population and a winner is then selected.

  The size of the tournament, n_t, determines the selective pressure of the selection operator. Increasing the tournament size results in a greater chance that fit individuals will dominate the tournament, increasing selective pressure. Decreasing the size of the tournament has the opposite effect. If $n_t = 1$, the selection is equivalent to random selection.

- **Rank-based selection**, where individuals are selected based on their fitness rank instead of absolute fitness. Rank-based selection therefore has a lower selection pressure than fitness based selection.
- **Elitist selection**, where the best individuals from the current population are selected to survive, without alteration, directly into the next generation. Having more individuals from the current population survive to the next generation reduces the diversity within the population. The elitist selection operator has a high selective pressure, based on the number of elitist individuals.
- **Replacement selection**, where an offspring individual replaces a parent individual in the next generation.
- **Culling selection**, where the least fit individuals are selected to be replaced.

Reproduction

Reproduction is the process which creates offspring by applying crossover and / or mutation. EAs such as GA, DE and ES implement reproduction through the application of crossover and mutation operators. Other EAs, such as EP, only implement a mutation operator.

The objective of these reproduction operators is:

- **Crossover** is the process of combining the genetic material of parent individuals to form new individuals, i.e. offspring. Crossover reduces the population diversity due to selection pressure through parent selection.
- **Mutation** introduces new material into the population by randomly changing individuals. The introduced material increases the diversity of the population, improving exploration.

2.2.2 Genetic Algorithms

Genetic algorithms simulate genetic evolution through the application of crossover, mutation and selection operators. This section discusses the canonical genetic algorithm, crossover and mutation operators. Selection operators, as discussed above, are applicable to GA.

Canonical Genetic Algorithm

Initially proposed by Fraser [56, 57] in the 1950s, and later by Bremermann [13] and Reed *et al.* [127], it was the work performed by Holland [70] that popularized GA, giving Holland the reputation of being the father of GA.

The operators within a GA are the selection operator (refer to Section 2.2.1), with offspring created through crossover and adapted using mutation. Algorithm 2 lists pseudocode for the GA.

Algorithm 2 Genetic algorithm

```
Let $t = 0$ be the generation counter.
Create and initialize an $n_x$-dimensional population, $P(0)$
Evaluate the fitness, $f(\mathbf{x}_i(t))$, of each individual within the population
repeat
  Select a group of parent individuals for reproduction
  Perform crossover on parent individuals to create offspring
  Mutate the generated offspring
  Evaluate fitness of offspring
  Select population, $P(t+1)$, from the current population and the generated offspring
until stopping condition(s) satisfied
```

Holland [70] proposed the canonical genetic algorithm (CGA), which uses

- a binary representation for individuals,
- parent selection using fitness proportionate selection,
- one-point crossover to create offspring, and
- uniform mutation of the offspring.

Subsequent developments from the CGA have introduced GAs to different problem search spaces and applied different crossover and mutation operators together with alternative selection operators.

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Crossover

Crossover is applied at a given crossover probability, $p_c \in [0, 1]$, also referred to as the crossover rate. Any selection scheme (discussed in Section 2.2.1) may be used to select parents for crossover. Parent selection schemes with large selective pressure favor fit individuals, resulting in the fit individuals participating frequently in crossover, increasing population uniformity as a consequence, and reducing exploration.

Crossover operators are grouped into two main classes:

- **Discrete recombination** [104] produces offspring by recombining genes from the selected parents. Discrete recombination operators can be applied to both discrete-valued and continuous-valued representations [103].
- **Intermediate recombination** produces offspring by using weighted averages of parent genes. Intermediate recombination operators can only be applied to continuous-valued representations.

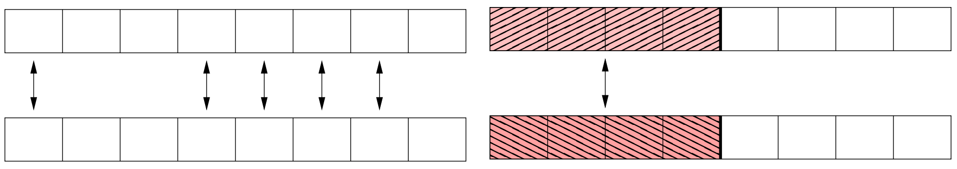
The following discrete recombinations are frequently used:

- **Uniform crossover**, which randomly swaps genetic material between parents to create offspring. Algorithm 3 summarizes the swapping process. In Algorithm 3,

Algorithm 3 Uniform crossover

```
for $j = 1$ to $n_x$ do
  if $U(0, 1) \leq p_x$ then
    $x'_{1,j} = x_{2,j}$
    $x'_{2,j} = x_{1,j}$
  else
    $x'_{1,j} = x_{1,j}$
    $x'_{2,j} = x_{2,j}$
  end if
end for
```

  p_x is the probability of a swap, $\mathbf{x}'$ is an offspring and $\mathbf{x}$ is a parent. Figure 2.1(a) illustrates uniform crossover, where two offspring are produced.


(a) Uniform crossover (b) One-point crossover

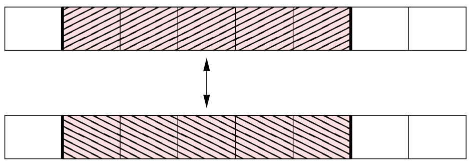

(c) Two-point crossover

Figure 2.1: Crossover operators for discrete recombination

- **One-point crossover**, proposed by Holland [70], randomly selects a gene position as crossover point. This results in two segments for each parent which are swapped to produce two offspring. Figure 2.1(b) illustrates one-point crossover.

- **Two-point crossover**, which is similar to one-point crossover, randomly selects two crossover points instead of a single crossover point. This results in three segments for each parent which are swapped to produce two offspring. Figure 2.1(c) illustrates two-point crossover.

Intermediate recombination results in a blending of multiple parents [45] to produce multiple offspring. Some continuous-valued multi-parent crossover operators are listed below:

- **Linear crossover**, developed by Wright [158], creates a linear combination of two parents $\mathbf{x}_1$ and $\mathbf{x}_2$, resulting in three candidate solutions:

  1. $\mathbf{x}_1(t) + \mathbf{x}_2(t)$
  2. $1.5\mathbf{x}_1(t) - 0.5\mathbf{x}_2(t)$
  3. $-0.5\mathbf{x}_1(t) + 1.5\mathbf{x}_2(t)$

  The best two candidate solutions become the offspring.

- **Arithmetic crossover:** Michalewicz [103] developed a multi-parent crossover operator that requires two or more parents and results in a single offspring from the selected n_u parents:
$$x'_{ij}(t) = \sum_{l=1}^{n_u} \gamma_l x_{lj}(t) \tag{2.9}$$
where $\mathbf{x}'_i$ is the offspring, and $\sum_{l=1}^{n_u} \gamma_l = 1$.

- **Blend crossover** creates a single offspring from two parents. Blend crossover (BLX-α), developed by Eshelman and Schaffer [45] as a variation of arithmetic crossover, creates the offspring using
$$x_{ij}(t) = (1 - \gamma_j)x_{1j}(t) + \gamma_j x_{2j}(t) \tag{2.10}$$
with $\gamma_j = (1 + 2\alpha)U(0, 1) - \alpha$. The BLX-$\alpha$ operator selects a random value in the range
$$[x_{ij}(t) - \alpha(x_{2j}(t) - x_{1j}(t)), x_{2j}(t) + \alpha(x_{2j}(t) - x_{1j}(t))] \tag{2.11}$$
for each dimension of the offspring.

  The BLX-α operator requires that $x_{1j}(t) < x_{2j}(t)$.

- **Parent centric crossover** creates offspring by stochastic aggregation of information from multiple parents. Such crossover operators include simulated binary crossover (SBX) [36, 35] and fuzzy recombination (FR) [155].

Mutation

Mutation is the process of introducing new genetic material into the population, increasing the population's genetic diversity. Each gene within an offspring is mutated at a probability, $p_m \in [0, 1]$. This means that there is a non-zero probability that an offspring may not be mutated. For discrete-valued representations mutation is implemented by randomly selecting a different value for a gene from the domain of the corresponding decision variable. For continuous-valued representations, a mutational step size is added to the current gene value. The mutational step size is the amount of variation, or noise, applied to a gene value. It is important that mutational step sizes are sampled from a zero-mean distribution to prevent genetic drift.

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Mutation may be detrimental to the search process as fit candidate solutions may mutate into undesirable solutions. The mutation rate results in two extremes:

- High mutation rate (and large mutational steps) cause candidate solutions to explore the problem search space. Individuals may step over desirable locations of the search space, or step out of such locations.
- Low mutation rate (and small mutational steps), result in candidate solutions exploiting and refining current positions within the search space. Individuals are more susceptible to becoming trapped within a local optimum, being unable to escape to more favorable regions of the problem space.

Consequently, the mutation rate and mutational step size are used to balance exploration and exploitation. Initially, a large mutation rate is preferred to facilitate exploration, but later in the search process a smaller mutation rate is preferred in order to refine the obtained solutions.

Different mutation operators are available, depending on the representation used. For binary-valued representations, the following mutation operators can be used:

- **Uniform mutation:** Randomly selected genes are negated.
- **Inorder mutation:** Two independent genes, η_1 and η_2, are selected such that $0 \leq \eta_1 \leq \eta_2 \leq n_x$. Uniform mutation is then applied to the genes between η_1 and η_2.
- **Gaussian mutation:** Hinterding [69] suggested that binary representations of continuous-valued decision variables should be converted back to a continuous-valued representation and then mutated with noise sampled from a zero-mean Gaussian distribution. Once mutation has been applied, the continuous-valued decision variable is once again translated to a binary representation. Hinterding [69] showed that Gaussian mutation provided better results than simply negating bit values.

For continuous-valued representations, the following mutation operators can be used:

- **Uniform mutation:** Michalewicz [103] demonstrated that by using a continuous-valued representation instead of a binary representation, better performance is achievable. The following uniform mutation operator was one of the first mutation operators proposed for continuous-valued representations:

$$\widetilde{x}_{ij}(t) = \begin{cases} x'_{ij}(t) + \Delta(t, x_{\max,j} - x_{ij}(t)) & \text{if a random digit is 1} \\ x'_{ij}(t) + \Delta(t, x_{ij}(t) - x_{\min,j}(t)) & \text{if a random digit is 0} \end{cases} \tag{2.12}$$

  where $\Delta(t, x)$ returns a random value from $[0, x]$ and $\widetilde{x}_{ij}(t)$ is the mutated offspring individual, at index j.

- **Headless chicken:** An offspring is created by recombining (through the use of an appropriate crossover operator) an individual with a randomly generated individual [75]. Headless chicken is also applicable to other representations, e.g. binary representations.

- **Non-uniform distribution based mutation:** Genes selected for mutation are adapted as follows:

$$\widetilde{x}_{ij}(t) = x'_{ij}(t) + \Delta x_{ij}(t) \tag{2.13}$$

  where $\Delta x_{ij}(t)$ is sampled from some zero-mean distribution to prevent genetic drift, $x'_i(t)$ is an offspring and $\widetilde{x}_j(t)$ represents the mutated offspring.

Mutation operators for mixed representations (e.g. integer-valued, continuous-valued, discrete-valued) are outside the scope of this thesis.

2.2.3 Evolutionary Programming

EP simulates phenotypic evolution through the application of mutation and selection. As proposed by Fogel [51], EP initially focused on solving finite state machines (FSMs) [47], which are behavioral models consisting of several finite states and associated transitions (or actions) between the states which determine state flow. The initial EP used a discrete-valued representation, but subsequent EP developments use a continuous-valued representation.

Due to the focus of phenotypic evolution, the fitness of an individual is not measured as an absolute value, but rather as an error relative to the current environment and other

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

individuals with small errors associated with fit individuals and large errors with unfit individuals.

Offspring are created through the use of a mutation operator alone, with each parent producing a single offspring. The next population is selected from the union of parents and offspring by using any of the selection operators discussed in Section 2.2.1. Algorithm 4 provides pseudo-code for the EP.

Algorithm 4 Generic evolutionary programming

```
Let $t = 0$ be the generation counter
Create and initialize an $n_x$-dimensional population, $P(0)$
Initialize the strategy parameters
for each individual do
    calculate fitness, relative to current population, $P(t)$
end for
repeat
    for each individual do
        Apply mutation to create offspring
        Calculate relative fitness of offspring
        Add offspring to offspring population, $P'(t)$
    end for
    Select a new population, $P(t+1)$, from $P(t) \cup P'(t)$
until stopping condition(s) satisfied
```

Mutation

Applying mutation to each parent produces a single offspring within EP. Mutation controls the amount of exploration and exploitation through a mutational step size obtained from a zero-mean probability distribution. Early in the search process the mutational step size should be large to facilitate exploration, but should be reduced as the optimization algorithm proceeds to allow offspring to exploit solutions found by parents.

With focus on continuous-valued EP, mutation is defined as

$$x'_{ij}(t) = x_{ij}(t) + \Delta x_{ij}(t) \tag{2.14}$$

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

where $x'_{ij}(t)$ is the j-th decision variable of the offspring created by applying a mutational step size $\Delta x_{ij}(t)$ to the j-th decision variable of the parent $x_{ij}(t)$ for time step t. Mutational step sizes are calculated as

$$\Delta x_{ij}(t) = \sigma_{ij}(t) \cdot \eta_{ij}(t) \tag{2.15}$$

where $\sigma_{ij}(t)$ is a strategy parameter which scales noise sampled from a zero-mean probability distribution $\eta_{ij}(t)$. Sampled noise may be selected from one of several probability distributions:

- **Uniform:** A uniform distribution

  $$\eta_{ij}(t) \sim U(x_{min,j}, x_{max,j}) \tag{2.16}$$

  is sampled where x_{min} and x_{max} respectively provide the lower and upper bounds for the distribution. Because all sampled values are equally likely, little exploitation is available to individuals. A zero-mean is important to prevent genetic drift within the population.

- **Gaussian:** Noise is sampled from a zero-mean normal distribution, with a deviation of 1:

  $$\eta_{ij}(t) \sim N(0, 1) \tag{2.17}$$

  Small deviations result in small mutational step sizes which prevent exploration of the search space but facilitate exploitation.

- **Cauchy:** The Cauchy distribution is similar in shape to the Gaussian distribution but has wider tails as illustrated in Figure 2.2. These wider tails result in larger mutational step sizes when compared to the mutational step sizes obtained by sampling a Gaussian distribution [159]. Larger mutational step sizes allow for better search space exploration, but prevent exploitation of current solutions. Noise is sampled from a zero-mean distribution

  $$\eta_{ij}(t) \sim C(0, \sigma) \tag{2.18}$$

  where the scale parameter σ determines the distribution spread.

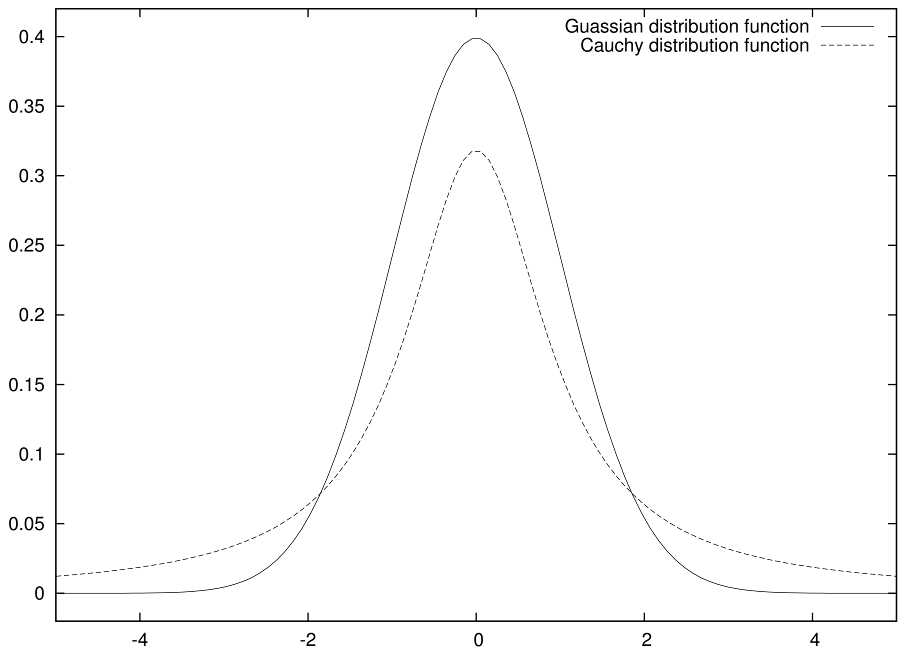


Figure 2.2: Gaussian and Cauchy distributions

- **Lévy:** The distribution curve is similar in shape to that of the Gaussian and Cauchy distributions, but allows for alteration of the distribution shape. Adjusting the shape parameter $v \in (0, 2)$ results in different distribution sampling. When $v = 1$, sampling the distribution provides values from the Cauchy distribution, where $v = 2$ provides values from the Gaussian distribution. It is advantageous to adjust the shape of the Lévy distribution from $v = 1$ to $v = 2$ during the search process to allow for better exploration initially, transforming to better exploitation as the search process completes. The Lévy distribution has an infinite second moment, increasing the chances that offspring will be generated farther away from the parent, unlike with Gaussian mutation [92].

  Noise is sampled from the distribution as:

$$\eta_{ij}(t) \sim L(v) \tag{2.19}$$

Several other probability distributions may also be applied to calculate the mutational step size, including the Exponential distribution [88, 105], Chaos distribution [74] and

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

distributions derived from combinations of other distributions [20].

Selection

Selection determines the individuals that survive to the next generation. Although the objective function provides the absolute fitness of an individual, a relative fitness determines the behavioral error of an individual compared to a pool, $\mathcal{P}$, of n_p randomly selected individuals. A relative fitness score is determined for each individual within the pool by competing with the other individuals within the pool. Individual i is assigned a score $s_i(t)$, calculated as

$$s_i(t) = \sum_{j=1}^{n_p} s_{ij}(t) \tag{2.20}$$

where the score function, $s_{ij}(t)$ is (assuming minimization)

$$s_{ij}(t) = \begin{cases} 1 & \text{if } f(\mathbf{x}'_i(t)) < f(\mathbf{x}'_j(t)) \\ 0 & \text{otherwise} \end{cases} \tag{2.21}$$

Individuals from the parent population and the produced offspring population then compete for survival based on the relative fitness score. Elitist selection is usually used to create the new population, but any of the selection operators discussed in Section 2.2.1 may be used.

Evolutionary programming implementations

Different mutation and selection operators result in variations of the generic EP algorithm. These implementations include:

- **Classical Evolutionary Programming:** Yao *et al.* [159] coined the term classic evolutionary programming (CEP), to refer to an EP which uses Gaussian mutation and culling to select the next population. A self-adaptive strategy parameter scheme is also used to update the strategy parameters using a log-normal method [50]. The strategy parameter update is

$$\sigma_{ij}(t+1) = \sigma_{ij}(t)e^{\tau N_i(0,1)+\tau' N_{ij}(0,1)} \tag{2.22}$$

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

where

$$\tau' = \frac{1}{\sqrt{2\sqrt{n_x}}} \tag{2.23}$$

$$\tau = \frac{1}{\sqrt{2n_x}} \tag{2.24}$$

Offspring are then produced using

$$x'_{ij}(t) = x_{ij}(t) + \sigma_{ij}(t)N_{ij}(0,1) \tag{2.25}$$

The next population is then the union of parents and offspring which is culled to ensure that only the most fit survive.

- **Fast Evolutionary Programming:** Yao *et al.* [159] altered the CEP to use a Cauchy distribution instead of a Gaussian distribution, calling the new algorithm fast evolutionary programming (FEP). As indicated in figure 2.2, the tails of the Cauchy distribution are wider than the tails of the Gaussian distribution. The larger tails mean that the mutational step sizes for the FEP are larger than the mutational step sizes of the Gaussian distribution. The larger mutational step sizes are initially beneficial as larger jumps mean that more of the search space is explored. A disadvantage of the Cauchy distribution is that it produces fewer small mutational step sizes, compared to the Gaussian distribution, resulting in less search space exploitation.

The interested reader is referred to [24, 85, 105] for more EP implementations.

2.2.4 Differential Evolution

Storn and Price [143] published differential evolution (DE) in 1995 as another stochastic population-based EA. DE differs from other EAs in that mutational step sizes are not sampled from a probability distribution but are calculated as stochastically weighted difference vectors of randomly selected individuals. DE also differs from other EAs in that mutation is first applied for each individual to create a *trial vector*, which then takes part in a crossover with the parent to create a single offspring. Selection is implemented as a competition between the parent and the generated offspring. DE is also only defined for use within continuous-valued problem spaces.

Algorithm 5 provides pseudo-code for DE. The sections that follow discuss the operators within DE and include initialization, difference vectors, mutation, recombination and selection.

Algorithm 5 Differential evolution

```
Let $t = 0$ be the generation counter
Create and initialize the population, $P(0)$
**repeat**
  **for** each individual $\mathbf{x}_i(t)$ **do**
    Evaluate the fitness $f(\mathbf{x}_i(t))$
    Generate a trial vector $\mathbf{u}_i(t)$ by applying a mutation operator
    Generate an offspring individual $\mathbf{x}'_i(t)$ by applying a crossover operator
    **if** $f(\mathbf{x}'_i(t))$ is more fit than $f(\mathbf{x}_i(t))$ **then**
      $\mathbf{x}'_i(t)$ is included in the next generation, $P(t+1)$
    **else**
      $\mathbf{x}_i(t)$ is included in the next generation, $P(t+1)$
    **end if**
  **end for**
**until** *stopping condition(s) satisfied*
```

Initialization

A continuous-valued representation is used to represent candidate solutions. As with other EAs, individuals are initialized to uniformly cover the search space.

Difference Vectors

The population of individuals represents knowledge of the current problem search space, with the distances between individuals determining population diversity. The goal of the search process is to obtain a single solution, onto which all population individuals will converge.

The distances between randomly selected, unique individuals determines the mutational step size with which a target individual will be mutated. The calculation of difference vectors allows for the search process to guide itself using knowledge obtained from

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

the current individuals. Note that mutational step sizes approach a Gaussian (Normal) distribution, as defined by the central limit theorem [140], provided that the population is large enough to allow for the creation of a sufficient number of difference vectors [141].

Mutation

The mutation process creates a trial vector, $\mathbf{u}_i(t)$, through the application of one or more difference vectors to a target individual. The generated trial vector is then subsequently used to create a single offspring. Each individual within the current population $\mathbf{x}_i(t)$ generates a trial vector $\mathbf{u}_i(t)$.

Storn and Price [142] proposed two trial vector creation approaches:

- **Scheme 1:** Each parent individual $\mathbf{x}_i(t)$ randomly selects a target individual $\mathbf{x}_{i_1}$ such that $i \neq i_1$. Two additional individuals, $\mathbf{x}_{i_2}(t)$ and $\mathbf{x}_{i_3}(t)$, are then randomly selected such that $i \neq i_1 \neq i_2 \neq i_3$.

  The trial vector is calculated using

  $$\mathbf{u}_i(t) = \mathbf{x}_{i_1}(t) + \beta(\mathbf{x}_{i_2}(t) - \mathbf{x}_{i_3}(t)) \tag{2.26}$$

  where $\beta \in (0, \infty)$ is a scaling factor determining the influence of the difference vector $\mathbf{x}_{i_2}(t) - \mathbf{x}_{i_3}(t)$. This trial vector creation strategy has since become known as the *rand* strategy.

- **Scheme 2:** Similar to the rand strategy, scheme 2 mutates a randomly selected target vector but uses two difference vectors. Trial vector creation then becomes

  $$\mathbf{u}_i(t) = \mathbf{x}_{i_1}(t) + \beta(\mathbf{x}_{i_2}(t) - \mathbf{x}_{i_3}(t)) + \lambda(\mathbf{x}_i(t) - \hat{\mathbf{x}}(t)) \tag{2.27}$$

  where $\hat{\mathbf{x}}(t)$ is the current best individual within the population and λ a scaling factor indicating the influence of additional term. Including the λ term biases the generated trail vector towards the population global best individual, restricting exploration of the search space. This strategy for the creation of the trial vector has become known as the *rand-to-best* strategy.

Recently, several other mutation operators have been created for DE and include [76, 141, 143]:

- **Multiple difference vectors (n_v):** This strategy uses more than one difference vector to create a trial vector, calculated as

$$\mathbf{u}_i(t) = \mathbf{x}_{i_1}(t) + \beta \sum_{k=1}^{n_v} (\mathbf{x}_{i_2,k}(t) - \mathbf{x}_{i_3,k}(t)) \tag{2.28}$$

  where $\mathbf{x}_{i_2,k}$ and $\mathbf{x}_{i_3,k}$ indicate the k-th difference vector. With more difference generated, more directions within the search space can be explored.

- **Current-to-best:** A parent is mutated using at least two different difference vectors. One difference vector is calculated between the parent and the best vector in the current population, while the remaining difference vector(s) are calculated using randomly selected vectors:

$$\mathbf{u}_i(t) = \mathbf{x}_i(t) + \beta(\hat{\mathbf{x}}(t) - \mathbf{x}_i(t)) + \beta \sum_{k=1}^{n_v} (\mathbf{x}_{i_1,k}(t) - \mathbf{x}_{i_2,k}(t)) \tag{2.29}$$

  where $\hat{\mathbf{x}}(t)$ is the current best vector. This strategy results in a linear combination which moves towards the best position and a random position within the search space, balancing exploration and exploitation.

- **Best with two difference vectors:** Introduced by Price [124], the best individual $\hat{\mathbf{x}}$ is selected from the population as the target vector. Two difference vectors are calculated from randomly selected individuals and the trial vector is calculated as

$$\mathbf{u}_i(t) = \hat{\mathbf{x}}(t) + \beta(\mathbf{x}_{i_1}(t) + \mathbf{x}_{i_2}(t) - \mathbf{x}_{i_3}(t) - \mathbf{x}_{i_4}(t)) \tag{2.30}$$

  where $i_1 \neq i_2 \neq i_3 \neq i_4$. It has been observed that the this strategy, using two difference vectors, seems to improve diversity if the size of the population is large enough [116].

Crossover

Crossover is the process that produces a single offspring $\mathbf{x}'_i(t)$ through the recombination of the target individual $\mathbf{x}_i(t)$ and the generated trial vector $\mathbf{u}_i(t)$.

Crossover takes place at selected crossover points $j \in \mathcal{J}$ where $\mathcal{J}$ is the set of selected crossover points within $\mathbf{x}_i(t)$. Crossover points may be selected using one of the following two strategies:

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

- **Binomial selection:** Crossover points are randomly selected, based on a rate of recombination p_r:

$$\mathcal{J} = \{j|j \in \{1, 2, \ldots, n_x\}, U(0,1) \leq p_r\} \tag{2.31}$$

  Having a large p_r value result in more crossover points being selected, with small values of p_r resulting in fewer crossover points. Fewer crossover points result in offspring that are similar, if not duplicates of the parent resulting in stationary individuals within the search space. To prevent such a situation, the set of crossover points contains a randomly selected point $j^* \sim U(1, n_x)$ that ensures at least one crossover point will always be present in the generated offspring.

- **Exponential selection:** Starting from a randomly selected point within the individual, the individual is then treated like a circular array. Adjacent points are selected until the recombination probability p_r is larger than a sampled uniform random number between 0 and 1. At least one crossover point will be included within the set of crossover points.

From the selected set of crossover points, a single offspring is created using

$$x'_{ij}(t) = \begin{cases} u_{ij}(t) & \text{if } j \in \mathcal{J} \\ x_{ij}(t) & \text{otherwise} \end{cases} \tag{2.32}$$

Selection

To determine which individuals survive into the next generation, the offspring $\mathbf{x}'_i(t)$ and the parent $\mathbf{x}_i(t)$ are compared using a greedy comparison. If the fitness of the offspring is better than the fitness of the current parent, the offspring replaces the parent in the next generation. However, if the fitness of the offspring is worse than that of the parent, the parent survives to the next generation, discarding the offspring (assuming minimization):

$$\mathbf{x}_i(t+1) = \begin{cases} \mathbf{x}'_i(t) & \text{if } f(\mathbf{x}'_i(t)) < f(\mathbf{x}_i(t)) \\ \mathbf{x}_i(t) & \text{otherwise} \end{cases} \tag{2.33}$$

2.3 Swarm Intelligence

In nature the primary focus for any organism is survival. Continued survival then also increases the chances for an organism to reproduce, yielding offspring. Survival of an

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

organism means that the organism must constantly adapt to an ever changing environment, an environment that introduces problems for an organism to overcome in order to survive. For an organism to sustain itself, it must feed. Foraging for food requires that an organism should use the least amount of energy to obtain a maximal amount of food. A single organism will more than likely spend most of its existence foraging, but if a group of similar organisms work together, the amount of effort required per individual will be less, satisfying the need to use less energy for the same benefit. The organism then maximizes the reward obtained from cooperation with similar organisms. The foraging behavior of ants [38] is an example of such cooperation, whereby the entire colony of ants benefits from the foraging of a percentage of the colony. Similar benefits are visible when birds flock [40], when fish school for protection [68] or to feed and when bees dance to enlist other bees to forage at a certain location [114].

Organisms that cooperate socially to achieve a common goal display a behavior that emerges from the simple interactions between individual organisms. The perceived behavior, as a result of the interaction, is far greater than what would be possible for a single organism and is known as *emergent behavior* [30, 68]. Reynolds [131] observed that simple movements by simple individuals can collectively result in an emergent behavior through his "boid" experiments.

Organism interaction need not be direct. *Stigmergy* [100, 148] refers to an indirect form of communication between organisms, and is observable whilst predators hunt. The prey will try to outsmart the predators and the predators in turn will change their strategies, without communication, to try ensure a kill. The predators form a *swarm* of cooperating organisms, displaying an emergent hunting behavior. Modeling simple interactions between organisms to produce a complex, emergent behavior is the basis for computational swarm intelligence (SI) [12].

The remainder of this section discusses two such SI paradigms. Section 2.3.1 discusses particle swarm optimization which models flocking behavior of birds while Section 2.3.2 discusses the artificial bee colony, a model of the foraging behavior of ants.

2.3.1 Particle Swarm Optimization

Kennedy and Eberhart [82] introduced particle swarm optimization (PSO) in 1995. The PSO is a stochastic population-based search algorithm, based on a simplified model of

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

bird flocking behavior. The PSO [26, 43, 82] maintains a group, or *swarm* of homogeneous candidate solutions to an optimization problem. Within the swarm, the individual candidate solutions are then referred to as *particles*. Particles move around the problem search space by adhering to the following simple behaviors:

- mimic the speed and direction of a neighboring particle, and
- to move towards the best state experienced by the particle since the beginning of the search process.

Particles move through the problem search space by applying a step size to their current position. The step size defines direction together with magnitude and is also referred to as a *velocity*. Each particle within the swarm has a predefined momentum, causing each particle to continue moving in a specific direction with a given magnitude. The magnitude and direction of a particle's velocity are altered by combining the particle's memory of its own previously found best position within the search space and the information of its neighbors.

Kennedy and Eberhart [82] defined the canonical PSO, as given by pseudo-code in Algorithm 6. The remainder of this section discusses aspects of the algorithm including initialization, particle attractors, velocity updates, position updates and neighborhood best selection strategies.

Initialization

Particles should uniformly span the entire problem search space, just like individuals within an EA (Section 2.2.1). Unlike EA individuals, each particle i maintains a memory of the best personal position observed from the first time step and a velocity determining the current momentum of the particle. The initial velocity of a particle may be:

- **Random:** The initial velocity is sampled from a uniform distribution. Large random values result in large initial step sizes, potentially letting the particles leave the problem search space. Smaller random values result in smaller step sizes. Defining a random initial step size may require more iterations for the PSO, as initial momentums may need adjustment to correct the momentum of a particle, or to bring particles that have left the boundaries of the search space due to initial random values [41].

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Algorithm 6 Canonical particle swarm optimization

```
Create and initialize a $n_x$-dimensional swarm
Calculate the fitness of each particle
repeat
  for each particle $i = 1, \ldots, n_s$ do
    Update the personal best for particle $i$
    Update the neighborhood best for particle $i$
  end for
  for each particle $i = 1, \ldots, n_s$ do
    Update the velocity of particle $i$
    Update the position of particle $i$
    Calculate the fintss of particle $i$
  end for
until stopping condition(s) satisfied
```

- **Zeroed:** The initial velocity is set to a zeroed vector. Starting particles from a rest position allows the particles to determine the initial velocity from the neighboring particles, and is preferable as the initial spread of particles already provides a good representation of the problem search space. Following the flocking behavior of birds, a zeroed initial velocity supports the model that flocking may start from a stationary position.

Particle attractors

Particles move around the problem search space by mimicking their own successes and those of their neighbors. In other words, a neighboring particle and the previous best particle position attract the particle.

The previous best position of a particle, known as the *personal best position* or *pbest*, is a memory of the best position found by a particle thus far in the search process. Considering a minimization optimization problem, the *pbest* $\mathbf{y}_i(t)$ for particle i at the next time step is then determined as

$$\mathbf{y}_i(t+1) = \begin{cases} \mathbf{x}_i(t+1) & \text{if } f(\mathbf{x}_i(t+1) < f(\mathbf{y}_i(t)) \\ \mathbf{y}_i(t) & \text{if } f(\mathbf{x}_i(t+1) \geq f(\mathbf{y}_i(t)) \end{cases} \tag{2.34}$$

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

where $f : \mathbb{R}^{n_x} \to \mathbb{R}$ is the fitness function.

The best position within the particle neighborhood is known as the neighborhood best $\hat{\mathbf{y}}_i(t)$ or *nbest*. The neighborhood best position depends on the neighborhood topology used, together with a position selection which can either be from the personal best positions of all neighborhood particles or from current particle positions [43]. If the neighborhood best is selected from personal best positions, particles will be attracted to their best positions since the first time step, resulting in a memory-based mechanism that facilitates more exploitation. Using current positions, i.e. no memory, forces particles to use current experience and results in more fluctuations in *pbest* initially, thereby facilitating more exploration. The neighborhood topology determines particle interactions, with the following topologies being the most frequently used [81, 119]:

- **Global best:** All particles within the swarm are interconnected, with the entire swarm regarded as the neighborhood for each particle. Considering minimization problems, the neighborhood best is then (assuming a memory-based solution)

$$\begin{aligned} \hat{\mathbf{y}}_i(t) &= \hat{\mathbf{y}}(t) \in \{\mathbf{y}_0(t), \ldots, \mathbf{y}_{n_s}(t)\} \\ \text{such that } \hat{\mathbf{y}}(t) &= \min\{f(\mathbf{y}_0(t)), \ldots, f(\mathbf{y}_{n_s}(t))\}, \forall i = 1, \ldots, n_s \end{aligned} \tag{2.35}$$

  where n_s is the size of the swarm.

  The global best position *gbest* attracts all particles within the swarm towards a single position within the search space. As a result of all particles moving towards the *gbest*, less of the search space is then explored. The global best topology is also referred to as the *star* topology, illustrated in Figure 2.3(a). When the PSO uses the global best topology, it is also referred to as the *gbest* PSO.

- **Local best:** Particles are organized into overlapping neighborhoods using particle indices, and not spatial information, to form a ring lattice topology. The ring topology uses particle indices to determine the neighbors of each particle. For a neighborhood size of $n_{\mathcal{N}_i}$ a neighborhood may consist of an odd number of at least three particles. Each neighborhood contains a neighborhood best particle $\hat{\mathbf{y}}_i(t)$ or *local best* with the neighborhood best possibly being shared across overlapping neighborhoods.

  Due to the fact that each neighborhood has its own attractor, the effect of each neighborhood best is localized to that neighborhood. Indirectly, due to the overlap of neighborhoods, information about the best positions found is gradually transferred to the rest of the swarm. Therefor, convergence is usually slower compared to the *gbest* topology but more exploration of the search space is achieved.

The *lbest* topology, also referred to as the *ring* topology, is illustrated in Figure 2.3(b). The *lbest* PSO is a common term referring to a PSO using the *lbest* topology.

- **Von Neumann:** Particles are organized into a square lattice structure, illustrated in Figure 2.3(c), where the neighbors of a particle include particles above, below and to the left and right. The Von Neumann topology has empirically been shown to have a better performance than that of the *gbest* and *lbest* topologies on a variety of problems [43, 84, 119].

Other neighborhood topologies can be found in [81, 84, 102].

Velocity updates

The velocity of a particle represents a step size, consisting of a direction and magnitude, with which a particle may adjust its current position within the search space. Velocity vectors are generated through the application of the velocity update equation.

Kennedy and Eberhart [82] introduced the velocity update equation as

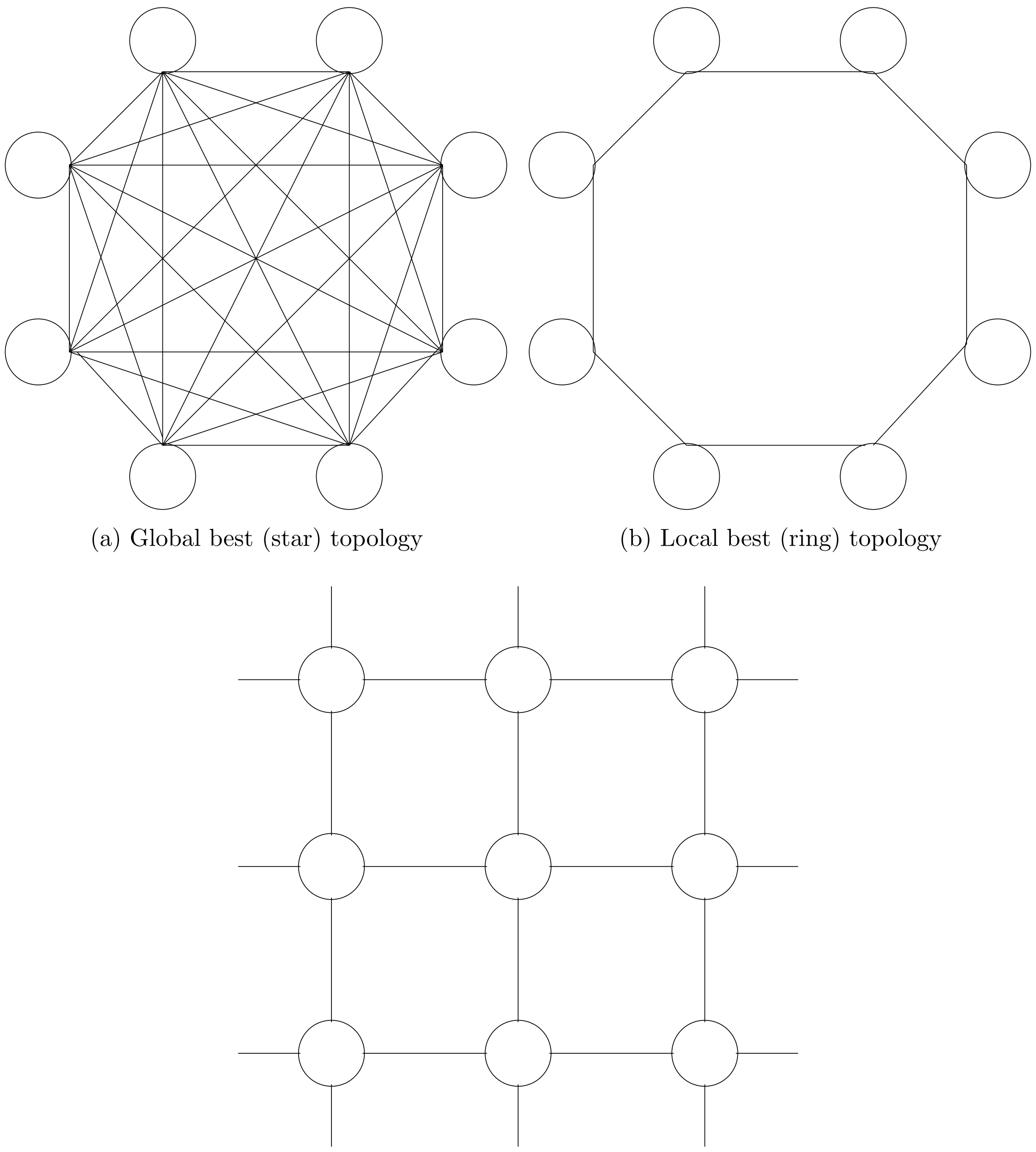
$$v_{ij}(t+1) = v_{ij}(t) + c_1 r_{1,j}(t)[y_{ij}(t) - x_{ij}(t)] + c_2 r_{2,j}[\hat{y}_{ij}(t) - x_{ij}(t)] \tag{2.36}$$

where each of the three terms defines a unique component:

- The previous velocity is also referred to as the *momentum*, or *inertia* component and biases the particle to continue on its current trajectory through the search space.

- The *cognitive component*, $\mathbf{y}_i(t) - \mathbf{x}_i(t)$, quantifies the success of particle i by determining the distance from the particle's best position to the current particle position. The cognitive component indicates the particle's attempt to move towards the best previously obtained solution. Kennedy and Eberhart [82] referred to the cognitive component as a *nostalgia* component for the particle.

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA



(a) Global best (star) topology

(b) Local best (ring) topology

(c) Von Neumann topology in two dimensions

Figure 2.3: PSO neighborhood topologies

- The *social component*, $\hat{\mathbf{y}}_i(t) - \mathbf{x}_i(t)$, determines the performance of particle i in relation the neighborhood best particle. The social component attracts the particle

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

towards the position of the best particle within the particle's neighborhood.

Within equation (2.36), c_1 and c_2 are positive acceleration constants that respectively scale the cognitive and social components, and $\mathbf{r}_1(t), \mathbf{r}_2(t) \sim U(0,1)^{n_x}$ are random values, sampled from a uniform distribution. The weights, $c_1\mathbf{r}_1(t)$ and $c_2\mathbf{r}_2(t)$ then scale the cognitive and social components by a stochastic amount.

To find a good solution with the PSO, particles need a balanced amount of problem search space exploration and the exploitation. Applying the velocity update equation produces a step size, which determines the trade-off between exploration and exploitation of the problem search space. Large step sizes facilitate exploration of, whereas small step sizes facilitate exploitation of regions within the search space. The acceleration constants, c_1 and c_2, scale the influence of the cognitive and social components within the velocity update equation. The constant c_1 determines the amount of confidence a particle has in its own personal best position, whereas c_2 determines the amount of trust a particle has in its neighbors. When $c_2 = 0$, a particle relies solely on its own experiences thereby becoming an independent hill climber. A better solution within the search space replaces the personal best of the particle when found. With $c_1 = 0$ a particle relies on the experiences of its neighbors, drawing all particles to a single point, $\hat{\mathbf{y}}_i$. Exploration and exploitation of particles requires that these acceleration constants are balanced. Larger values of c_1 (i.e. $c_1 > c_2$) will result in particles being attracted more to their personal best positions, facilitating exploration. Larger values of c_2 (i.e. $c_2 > c_1$) attract particles towards the neighborhood best, facilitating exploitation. When both c_1 and c_2 are large, mutational step sizes become large and particles jump around through the search space, potentially missing good areas of the search space. Smaller values of c_1 and c_2 allow particles to explore more of the search space before being attracted towards existing good regions.

Early applications of PSO showed that velocities would increase to very large values, resulting in particles leaving the domain of the search space. Velocity clamping was introduced by Eberhart *et al.* [41] to prevent excessive increases in step sizes. Velocities that exceed a defined maximum velocity $\mathbf{V}_{max}$ are clamped using

$$v_{ij}(t+1) = \begin{cases} V_{\max,j} & \text{if } v'_{ij}(t+1) \geq V_{\max,j} \\ -V_{\max,j} & \text{if } v'_{ij}(t+1) \leq -V_{\max,j} \\ v'_{ij}(t+1) & \text{otherwise} \end{cases} \tag{2.37}$$

where $v'_{ij}(t+1)$ is calculated using equation (2.36). The values of $V_{\max,j}$ control the granularity of the search process because increasing step sizes are clamped. Large values of $V_{\max,j}$ aid in exploration due to larger step sizes. Smaller step sizes facilitate exploitation, but have the additional consideration that for small values particles may not sufficiently explore good regions of the search space. Smaller $V_{\max,j}$ values may also require more time steps to reach an optimum. Values for $V_{\max,j}$ are problem dependent and although $\mathbf{V}_{\max}$ may prevent particle step sizes from exploding, it still does not prevent the particle moving outside the valid domain of the search space.

Shi and Eberhart [136] introduced an inertia weight w to the inertia component of equation (2.36) to control the exploration and exploitation of the swarm, and as a mechanism which would remove the need for velocity clamping [42]. The resulting velocity update equation is

$$v_{ij}(t+1) = wv_{ij}(t) + c_1 r_{1,j}(t)[y_{ij}(t) - x_{ij}(t)] + c_2 r_{2,j}[\hat{y}_{ij}(t) - x_{ij}(t)] \tag{2.38}$$

where w is the inertia weight scaling the influence of the previous velocity. With $w > 1$ velocities increase over time, increasing to a maximum velocity. This in turn facilitates exploration as particles readily move throughout the search space and usually results in the swarm diverging with particles unable to move back to towards promising regions of the search space. When $w < 1$ velocities of particles decrease over time, eventually tending to zero and effectively facilitating exploitation of current positions. As with $\mathbf{V}_{\max}$, the value of the inertia weight is problem dependent [137].

Although the inclusion of the inertia weight did address exploitation and exploration control, it did not eliminate the need for velocity clamping [42] as divergent behavior of particles was still observed. The divergent behavior is a consequence of the dependency between w, c_1, c_2 and that the values of these parameters and the consequent particle behavior can not be considered in isolation. Van den Bergh and Engelbrecht [152, 154] have shown that the dependency between w, c_1 and c_2 for convergent particle trajectories results in the following relation:

$$w > \frac{1}{2}(c_1 + c_2) - 1 \tag{2.39}$$

If the above relation does not hold, then particle trajectories will be divergent. Trelea [149] independently derived a similar relation between weight value and the acceleration coefficients.

Position updates

The position update moves the particle within the search space to a new position by applying a step size to the current position. The position update equation for particle i is

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1) \tag{2.40}$$

Synchronous versus Asynchronous

Two strategies exist which dictate the rate at which information about best found positions is disseminated. These strategies, referred to as the synchronous and asynchronous strategies, differ in the order of updating particle positions and best positions.

For the synchronous update (as in Algorithm 6), the position of each particle is first updated, then followed by an update of the personal best and neighborhood best positions. The asynchronous update, updates the personal and neighborhood best positions of each particle immediately after the position update. The asynchronous update therefore allows for faster feedback of potential good solutions in the problem search space. Conversely, the synchronous update has slower feedback propagation because the evaluation of neighborhood and personal bests only occur once per time step.

Charlisle and Dozier [18] reasoned that asynchronous updates are more beneficial to the *lbest* PSO, as the faster feedback is better for loosely connected neighborhoods, while the synchronous strategy is more appropriate to the *gbest* PSO.

2.3.2 Artificial Bee Colony Optimization

In the biological model, honey bees forage for food around a single bee hive. When a food source is located, the honey bees will forage until the food source is depleted and will inform awaiting honey bees at the hive about the food source. The honey bees transfer knowledge of food source locations through a *waggle dance* which is a series of movements that convey direction and distance information to the honey bees waiting at the dancing area within the hive. Once depleted, a food source is abandoned and the honey bees begin searching for the next food source. A *profitable food source* [133] has a close proximity to the bee hive, a large amount of food available, and requires the least amount of effort to forage. All foraging honey bees try to find food sources that are

Algorithm 7 Artificial bee colony

```
Create and initialize a $n_x$-dimensional swarm
repeat
    Place employed bees on food sources and determine fitness
    Determine probabilities for onlooker bees and place on food sources
    Foraging bees perform a local search to refine food source
    Abandon exhausted food sources
    Send out scouts to determine new food sources
until stopping condition(s) satisfied
```

highly profitable because these food sources directly benefit the entire bee hive.

Observations of honey bee foraging behavior facilitated the development of the artificial bee colony (ABC) algorithm. The ABC is a population-based algorithm developed by Karaboga *et al.* [7, 77] and Basturk *et al.* [78, 79] which focuses on the recruitment of bees and the abandonment of exhausted food sources. Within the problem search space, candidate solutions are referred to as *food sources* and are foraged by a group of bees known as *employed bees*. The employed bees simulate the foraging that occurs at a food source with each individual having knowledge of a single food source location. *Unemployed bees* are currently searching for food sources. Two groups of unemployed bees exist:

- **Scouts**, which fly through the search space randomly looking for food sources.
- **Onlooker** bees, which await the return of employed bees to the hive in order to learn the location of a food source.

Algorithm 7 provides pseudo-code for the ABC. The sections that follow discuss algorithm initialization, bee recruitment for foraging, and the abandonment of food sources.

Initialization

The first step in the initialization process is to divide the colony into two equal groups of n_e employed bees and n_o unemployed bees. The initial n_e food sources, $\mathbf{x}_i \in \{1, 2, \ldots, n_e\}$,

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

are randomly generated to uniformly cover the search space. Employed bees do not maintain a food source, but rather maintain a modified view $\mathbf{v}_i$ of a food source position. Symbolically, that is $\mathbf{v}_i = h(\mathbf{x}_i)$ with the modified view of a food source calculated as:

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) \tag{2.41}$$

where $k \in \{1, 2, \ldots, n_e\}$, $j \in \{1, 2, \ldots, n_x\}$ with $k \neq i$ and $\phi_{ij} \in U[-1, 1]$ scales the distance from the food source. Food source locations remain within the search space until they are depleted. New food sources are created within the search space as described in *food source abandonment*, which follows.

The profitability, or fitness, of each employed bee solution is determined by evaluating the objective function with the modified solution. If the fitness of the produced result is better than the previous result in memory, the new modified solution replaces the previous solution. The remaining onlooker bees await employed bees to return to the hive.

Recruitment and foraging

Foraging of food sources creates a cycle of employed bees returning to the hive to enlist the aid of onlooker bees, returning to a newly modified view of the food source to forage, until the food source is depleted. Onlooker bees select the food source $\mathbf{x}_i$ based on the probability

$$p_i = \frac{f(\mathbf{x}_i)}{\sum_{n=1}^{n_e} f(\mathbf{x}_n)} \tag{2.42}$$

associated with the food source $\mathbf{x}_i$, where f is the objective function determining the profitability of a solution. When an onlooker be selects a food source, it also produces a modified solution view of the food source $\mathbf{x}_i$ using equation (2.41), updating its memory if required. Both employed and onlooker bees update their memory based on a greedy criterion which exploits the current solutions in the problem search space.

Food source abandonment

Foraging continues until no further improvement of a food source $\mathbf{x}_i$ occurs, indicating food source depletion. If solutions around $\mathbf{x}_i$ do not improve after a *limited* number of time steps, the food source is abandoned. Employed bees at abandoned food sources

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

then become scouts and attempt to discover a new food source, calculated as a random position within the domain of the problem:

$$x_{ij} = x_{min,j} + U(0,1)(x_{max,j} - x_{min,j}) \tag{2.43}$$

where $\mathbf{x}_{min}$ and $\mathbf{x}_{max}$ respectively are the lower and upper bounds of the problem domain, and $U(0,1)$ is a uniform random number. It has been suggested that the number of scouts within a population of bees should not exceed 5% - 10% [133] of n_e. The abandonment of food sources allows other regions of the search space to be explored, facilitating search space exploration.

2.4 Summary

This chapter started with a formal definition of optimization problems. Algorithms from the paradigms of evolutionary computation and swarm intelligence were then discussed with a focus on algorithms used to solve continuous-valued optimization problems. With in the EC paradigm, genetic algorithms, evolutionary programming and differential evolution have been discussed. Within the swarm intelligence paradigm, particle swarm optimization and the artificial bee colony algorithms were then discussed.

These algorithms have been selected as they were specifically developed to solve continuous-valued optimization problems. The next chapter discusses how these algorithms can be adapted to solve binary-valued optimization problems.

Chapter 3

Population Based Binary Algorithms

This chapter elaborates on the use of a binary-valued variable representation, together with the inherent problems of the representation. A discussion of binary optimization algorithms follows, including a discussion of continuous algorithms, adapted to operate within the binary problem space. Issues regarding such adaptations for continuous algorithms are then highlighted.

3.1 Introduction

Variable representation for optimization algorithms differs based on the problem search space defined by the optimization problem. Variable representations include continuous-, discrete- and binary-valued, and each variable within and optimization problem need not be the same, allowing for mixed variable representations. Efficient algorithms are available which can solve optimization problems with different variable representations, specifically with binary-valued variable representations. Binary-valued optimization problems, such as feature selection [16, 90, 121], multiple sequence alignment [96], iterated prisoner's dilemma [59, 60, 146], combinatorial circuit design [29, 97, 128], and constraint-satisfaction problems such as the n-queens problem [150], have had binary optimization algorithms, such as the canonical GA [70], EP [50], PSO [28] and ant colony optimization (ACO) [146] provide solutions.

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Recently developed optimization algorithms, using continuous-valued variables, proved successful in solving continuous-valued optimization problems. This chapter also investigates ways in which continuous-valued optimization algorithms can be used to solve binary-valued problems, by highlighting issues related to the use of binary representations, existing binary algorithms, and modifications to continuous algorithms to allow operation in binary problems spaces.

The remainder of the chapter is outlined as follows: Section 3.2 discussing the different aspects related to the use of binary numbers as a representation for decision variables, with the problems relating to the use of binary representations discussed in Section 3.3. Binary algorithms are then discussed in Section 3.4, together with existing approaches to allow continuous-valued optimization algorithms to operate within the binary-valued problem space. Issues related to the use of these algorithms within a foreign, non-continuous, problem space are discussed in Section 3.5. Finally, the chapter concludes with a summary in Section 3.6.

3.2 Binary representation

A variety of different binary representations exist to represent decision variables within a binary-valued space. Listed in Section 3.2.1 are the different, more popular, binary representations, with translation techniques to represent continuous-valued variables within the binary space described in Section 3.2.2.

3.2.1 Binary variable representations

The most common representations for binary-valued variables are

- **The binary number system**, also referred to as the base-2 number system with an alphabet set of two elements, $\{0, 1\}$. A decimal number may be encoded as a binary number, represented as a string of binary digits (bits), where each bit is a single value from the set $\{0, 1\}$. The base-2 number system is a positional number system, whereby the position of each bit determines an order of magnitude difference of a value of two for the encoded value.

  The rightmost bit of a bit string represents the decimal value of 2^0 and is also

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

referred to as the least significant bit (LSB). For bit positions left of the LSB, the exponent of the decimal value 2 increases by one. In other words, the general form of a bit string is

$$2^n 2^{n-1} 2^{n-2} \dots 2^2 2^1 2^0 \tag{3.1}$$

with the leftmost bit referred to as the most significant bit (MSB).

Mathematical operations such as multiplication, division, addition and subtraction are valid for the binary number system and fractional values are also possible, with a general form of

$$2^n 2^{n-1} 2^{n-2} \dots 2^2 2^1 2^0 . 2^{-1} 2^{-2} \dots 2^{-m} \tag{3.2}$$

Additionally, the binary number system, similarly to the decimal number system, struggles to represent certain fractional values allowing for non-terminating, recurring fractions.

- **Gray coding**, developed as a bit string representation which requires a single bit position value to change in order to represent a subsequent encoded value. That is, subsequent encoded values have a Hamming distance of one. Any bit string, represented using the base-2 system, can be translated to a gray coded bit string using

$$g_1 = b_1 \tag{3.3}$$
$$g_2 = b_{i-1} \cdot \bar{b}_i + \bar{b}_{i-1} \cdot b_i \tag{3.4}$$

  where b_i is the bit at position i in the bit string $b_1 b_2 \dots b_n$; b_1 is the MSB of the base-2 bit string. $\bar{b}_i$ denotes the complement value of b_i, with $+$ denoting bitwise OR and $\cdot$ denoting bitwise AND.

  Table 3.1 illustrates an example gray coding, compared to the same decimal value encoded using the base-2 system.

- **Binary coded decimal**, is a bit string encoding which encodes a single integer value into a bit string of length four, representing the integers 0 to 9. If a fifth bit is used in the encoding, then the fifth bit indicates the sign of the encoded integer value.

Table 3.1: Binary and Gray coding for bit strings of length 3.

Integer	Base 2	Gray
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

3.2.2 Non-binary variable representations

When decision variables are not represented in binary, a translation, or mapping, is required to transform the decision variable into a binary representation. Such a mapping, translating a single continuous-valued decision variable to an n_d-dimensional bit string is

$$\phi_j : \mathbb{R} \to \{0,1\}^{n_d} \tag{3.5}$$

where ϕ_j is the mapping function which translates the continuous-valued variable into a bit string. The mapping function requires that the continuous-valued search space have a finite range of $[\mathbf{x}_{min}, \mathbf{x}_{max}]$. To translate a generated bit string back into a continuous-valued variable, a reverse mapping function is required [4]:

$$\Phi_j : \{0,1\}^{n_d} \to [x_{min,j}, x_{max,j}] \tag{3.6}$$

An example reverse mapping function is

$$\Phi_j(\mathbf{b}) = x_{min,j} + \frac{x_{max,j} - x_{min,j}}{2^{n_d} - 1}\left(\sum_{l=1}^{n_d-1} b_{j(n_d-l)} 2^l\right) \tag{3.7}$$

3.3 Problems with binary representations

Different concerns with the use of a binary representation exist, as mentioned in section 1.1. Such concerns include Hamming cliffs, the curse of dimensionality, loss of precision, dependencies between individual bits and the computational complexity of the representation. These problems are discussed in detail in this section.

3.3.1 Hamming cliffs

The *hamming distance* [66] between two bit strings of equal length is the number of positions at which corresponding symbols are different. In other words, the hamming distance between two bit strings is the minimum number of bits that needs to be substituted in order to change one bit string into another.

A *hamming cliff* is created when a hamming distance greater than one exists between the bit string representations of two adjacent numeric values. For example, consider the integers, 7 and 8. Examining the bit string representations of the integers results in a hamming distance of 4

$$\begin{aligned} 7_{10} &= 0111_2 \\ 8_{10} &= 1000_2 \end{aligned}$$

The resulting hamming distance means that four bits need to change to translate the bit string representation of the integer 7 into the bit string representation of the integer 8. Consequently, Hamming cliffs require that several bits change to represent a successive encoded values. Due to the difficulty in obtaining successive encoded values, binary algorithms may struggle to develop a sought bit string. In an attempt to avoid Hamming cliffs, Gray coding may be used by a binary algorithm, where the hamming distance is always one. While Gray coding resolves the Hamming cliffs problem, the remainder of the problems discussed below still persist.

3.3.2 Curse of dimensionality

The curse of dimensionality, a term coined by Bellman [8, 9], also known as the Hughes effect [71] or Hughes phenomenon, refers to the problem caused by the exponential increase in volume associated by adding extra dimensions to a problem space. Köppen [86] highlighted that, as the volume of the search space hypercube increases, the geometrical and statistical properties alter, creating potential problems for a search process.

Binary-valued problem spaces are sensitive to problem dimension modifications as the addition of a single bit doubles the number of potential candidate solutions to the

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

optimization problem. Encoding a continuous-valued variable as a bit string results in a bit string of length m, but encoding n continuous-valued variables results in a bit string of length $n \times m$. As the number of bits increases, the dimension of the binary search space increases, together with the possibility that many bit changes will be necessary to move the bit string from one solution to the next. Additionally, more individuals may be required by a PBA to search these higher dimensional spaces.

3.3.3 Loss of precision

Translation of continuous-valued variables into binary representation, as well as the reverse translation, causes a loss of precision in the continuous-valued space.

Holland [70] and De Jong [34] provided the first applications of a GA using a mapping from continuous to binary problem space. Obtaining a precise optimum is then not possible, with the maximum attainable accuracy for a bit string conversion to a continuous-valued space defined to be

$$\frac{x_{max,j} - x_{min,j}}{2^{n_d} - 1} \tag{3.8}$$

for each dimension j, where n_d is the length of the bit string, $x_{max,j}$ and $x_{min,j}$ are the defined maximum and minimum respective values for the continuous-valued space. The precision loss in the continuous-valued space may be reduced by increasing n_d, consequently increasing the dimension of the binary problem. It is very important to note that the discretization of a continuous-valued decision variable reduces the number of possible values that a continuous-valued decision variable may take, to a finite number. This in turn results in a finite number of candidate solutions, as each variable now has fewer possibilities.

3.3.4 Bit string interdependencies

When a continuous-valued decision variable is mapped to a binary representation, a bit string is obtained with very strong interdependencies among the individual bits. Altering a single bit within the resulting bit string will change the encoded continuous value with bits in more significant positions creating larger changes.

It is, therefore important that the m-dimensional bit string of each decision variable is regarded as an indivisible unit [23]. Operators within algorithms could be made aware

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

of bit string unit boundaries, possibly preventing bit changes in the encoded values of other dimensions.

3.3.5 Computational complexity

The translation process, via the mapping functions discussed in Section 3.2.2, incurs an additional computational cost. When translating a continuous-valued space into a binary-valued space, each decision variable converts into a bit string. Then, when reversing the encoding back into a continuous-valued space, a computation is also required. Although the conversion process is simple, multiple translations between problem spaces adds a significant computational cost to the optimization algorithm. This may not be preferable for the search process, and consideration will be necessary to determine if this additional complexity negatively affects the performance of the optimization algorithm.

3.4 Binary algorithms

Algorithms originally developed to operate within the binary problem space are also affected by the problems associated with binary representations. In all cases where a continuous-valued representation is encoded as a binary representation, the problems of Hamming distance, curse of dimensionality, loss of precision, strong bit interdependencies, and computational complexity exist. This section discusses how GAs and EP are affected by the problems discussed in section 3.3.

3.4.1 Canonical Genetic Algorithm

The canonical GA (refer to Section 2.2.2), also referred to as the binary genetic algorithm (BinGA), has successfully evolved solutions to continuous-valued problems where the decision variables are encoded as binary strings. While operators have been developed for GAs to work on binary representations, the following issues have to be noted:

- Hamming cliffs have the consequence that many mutations are required to move from one solution to the next. As the population converges onto a solution, the mutation rate is generally reduced. If a large Hamming distance exists between the candidate solution and the optimum, the mutation probability has to be increased.

Large mutation probabilities may, however, cause too much disruption within the population, causing individuals to move away from an optimum.

- The increase in problem dimension due to binary representation of continuous-valued variables increases the computational cost [23] of the mutation operator as more bits have to be mutated. Crossover also becomes a more computationally expensive operation as the problem dimension increases. The population may require more individuals to potentially cover more of the problem search space. With a large problem dimension, a larger mutation rate may be required to facilitate exploration. Larger mutation rates may disrupt the population, preventing convergence onto an optimum.

- As discussed in section 3.3.4, the bits that represent a continuous-valued decision variable must be treated as a unit. Binary crossover operators can therefore not select crossover points within such units, but should select crossover points on the boundaries of the bit strings representing the different continuous-valued decision variables.

3.4.2 Evolutionary programming

The first EP, or binary evolutionary programming (BinEP), used a binary representation for the individuals to evolve state transitions for a FSM (refer to section 2.2.3). For the mutation operator of EP, the following can be noted:

- An increase in problem dimension due to the binary representation of continuous-valued decision variables increases the computational cost of offspring creation through mutation. More individuals may be required to better represent the problem search space. Mutations may also need to be larger to ensure proper exploration, potentially preventing convergence onto an optimum.

- Hamming cliffs require many mutations to move from one solution to the next. The rate of mutation should be reduced as the population converges, but large differences between a candidate solution and an optimum requires more mutation. Because EP only uses mutation to create offspring, large mutations will result in a divergence in the population, away from an optimum.

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

3.5 Continuous algorithms

Numerous efficient algorithms have been developed to solve continuous-valued optimization problems. Although these algorithms have been adapted to solve binary-valued optimization problems, problems related to binary representation are still present in addition to new problems due to such adaptations. The main objective of this section is to discuss algorithms developed for continuous-valued problem spaces, and how these algorithms have been adapted to binary-valued problem spaces. Additionally, problems associated with these adaptations are also discussed. The rest of this section is outlined as follows: Section 3.5.1 discusses issues with the redefinition of algorithms, with Sections 3.5.2 – 3.5.4 discussing binary algorithm versions of PSO, DE and ABC.

3.5.1 Algorithmic redefinition

One of the first issues to address when a continuous algorithm is applied to solve a binary-valued optimization problem is a change in representation of candidate solutions from continuous-values to binary-values. Because the operators of continuous algorithms are designed specifically for continuous-valued representations, these operators have to be adapted to work on binary-valued representations.

However, adaptation of operators or processes of an algorithm may result in the following problems:

- Some operators, for example the velocity update in PSO, the food source update of ABC, and the mutational step of DE, can only be applied to vectors of continuous values. In order to be applied to binary-valued representations, new abstract definitions of these operators have to be defined.

- Such redefinition of operators may result in a change of the underlying behavior of the algorithm, violating the fundamental principles upon which the algorithm was developed. For example, binary particle swarm optimization (refer to section 3.5.2) changes the meaning of the velocity update equation from a step size to a probability of a change in bit value.

- Caution needs to be taken in the application of operators because not all operators can directly operate on a binary representation due to the mathematics used within the operator. An example of such a violation is:

$$0 - 1 \tag{3.9}$$

where '$-$' is the binary subtraction operator. Any violation resulting from an adaptation will result in the binary version of the algorithm to not be compatible with the original, unmodified algorithm.

3.5.2 Binary particle swarm optimization

The original PSO was developed for continuous-valued problem spaces only. The BinPSO, developed by Kennedy and Eberhart [83], enabled the PSO to also solve binary-valued optimization problems by altering:

- the particle representation to a bit string representation,
- the position update equation, and
- changing the meaning of the velocity from a step size to a probability of a bit change.

It should be noted that the velocity vectors of particles within BinPSO remain continuous-valued vectors.

Each of these changes are discussed in more detail below, with their associated problems. As discussed in section 3.3, the BinPSO inherits the issues caused by using binary representations for candidate solutions.

The position update equation changes to

$$x_{ij}(t+1) = \begin{cases} 1 & \text{if } r_j(t) < sig(v_{ij}(t+1)) \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

where $r_j(t) \sim U(0,1)$ determines the value of bit j, and $sig(v_{ij}(t+1))$ represents the probability of changing the bit value. The velocity is therefore used to calculate the probability of a bit change as follows,

$$sig(v) = \frac{1}{1+e^{-v}} \tag{3.11}$$

where $v \in (-\infty, \infty)$. These changes alter the intention of the PSO by letting the velocity vector no longer represent an incrementally updating step size for a particle. The velocity now represents a probability of change in bit value from which the position update for a particle is based. The behavior of the entire swarm therefore changes because velocities no longer provide a search trajectory defined by the incremental update of a position vector. Consequences are that the theoretical analysis of PSO are no longer applicable, and analysis of the influence of PSO parameters on performance with respect to the exploration – exploitation trade-off no longer applies.

It is very important to note that repeated application of equation (3.10) on the same particle to produce a bit string may result in different bit strings. This is due to the uniform random number, sampled for each dimension, and used to determine the bit value. Furthermore, producing different binary representations for the same candidate solution results in different fitness values for the same particle, providing conflicting evaluations of the quality of solutions. Particles may be attracted to poor regions of the search space because a poor solution was associated with a good fitness value.

Use of the sigmoid function to determine a probability of a bit change from the velocity introduces the following desirable properties:

$$\lim_{x\to\infty} sig(x) = 1 \tag{3.12}$$
$$\lim_{x\to-\infty} sig(x) = 0 \tag{3.13}$$
$$sig(0) = 0.5 \tag{3.14}$$

Velocities in the range $(-\infty, \infty)$ are therefore mapped to the range $(0, 1)$ [83], with zero velocities mapping to a probability of 0.5. Exploration is therefor achieved for velocities of a small magnitude, with exploitation facilitated as the magnitude of velocities increases. However, if the magnitude of velocities is too large, the sigmoid function becomes saturated. To prevent saturation, control parameters have to be properly set. The influence of control parameters on the performance of BinPSO changes compared to the continuous PSO (as discussed in section 2.3.1):

- To ensure initial exploration and later exploitation, the initial weight should be small, increasing over time. A small weight facilitates exploration initially by preventing the previous velocity from influencing the new velocity, resulting in more bit value changes.

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

- The initial velocity should be zero. Zero initial velocities result in 0.5 probabilities, i.e. equal chance for bit 0 and 1, resulting in maximum exploration.

- Velocity clamping, $\mathbf{V}_{max}$, thresholds should be small initially, increasing over time. Clamping the velocity to smaller values will facilitate more bit flips resulting in better exploration, but allowing for exploitation later as the clamping threshold increases.

- Acceleration constants, c_1 and c_2, should be small initially, increasing over time. Small values for the acceleration constants allow the velocity to remain close to zero, facilitating more bit flips and exploration.

Due to these changes, any suggested guidelines for the selection of control parameter values as developed for the canonical PSO do not apply to the BinPSO.

Clerc [25] proposed alternative changes to the PSO for solving binary-valued optimization problems, by redefining the meaning of the arithmetic operators used in the velocity and position update equations. The redefinition of the operators allowed Clerc to solve the traveling salesman problem (TSP), where the velocity is interpreted as a list of permutations and the particle position is a possible candidate tour for the TSP. Although such a redefinition of operators was successful in achieving a solution, the redefinition results in an algorithm that is vastly different to the continuous PSO.

Subsequent research has developed more general, discrete versions of the PSO, including SetPSO [22, 106] and the discretized PSO [25, 91, 107, 129, 164].

3.5.3 Differential evolution

As for the PSO, DE relies on vector algebra in the mutation operator. Therefore, the original DE was only applicable to solve continuous-valued optimization problems. However, the DE has since been adapted to solve binary-valued optimization problems. Two such binary DE variations, as proposed by Engelbrecht and Pamparà [44], are the binary differential evolution (BinDE) and normalization differential evolution (NormDE) algorithms. This section provides an overview of these binary DE algorithms, and discusses the issues related to these adaptations. It should be noted that both binary DE variations suffer from the issues related to using binary candidate solutions, as discussed in section 3.3.

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Binary differential evolution

The BinDE borrows concepts from the BinPSO, discussed in section 3.5.2. Individuals within the population use a continuous-valued representation to represent a vector of probabilities, and no longer an actual position in the search space. BinDE now searches for optimal bit flipping probabilities and these probabilities are used to generate a bit string, using

$$p_{ij}(t) = \begin{cases} 1 & \text{if } U(0,1) < sig(x_{ij}(t)) \\ 0 & \text{otherwise} \end{cases} \tag{3.15}$$

The binary-valued solution, $\mathbf{p}_i$, is then evaluated using the fitness function defined for the binary-valued optimization problem. The calculated fitness is then associated with the continuous-valued individual, $\mathbf{x}_i$. Algorithm 8 provides the pseudo-code for the BinDE.

Algorithm 8 Binary differential evolution.

```
Let $t = 1$ be the generation counter
Create and initialize the population, $P(0)$
repeat
  for each parent individual, $\mathbf{x}_i(t)$ do
    Generate a trial vector, $\mathbf{u}_i(t)$, by applying the mutation operator
    Generate an offspring, $\mathbf{x}'_i(t)$, by applying the crossover operator
    Generate bit string, $\mathbf{g}_i(t)$, for the individual, $\mathbf{x}_i(t)$, using equation (3.15)
    Generate bit string, $\mathbf{g}'_i(t)$, for the offspring, $\mathbf{x}'_i(t)$, using equation (3.15)
    if $f(\mathbf{g}'_i(t))$ is better than $f(\mathbf{g}_i(t))$ then
      $\mathbf{x}'_i(t)$ is included in the next population, $P(t+1)$
    else
      $\mathbf{x}_i(t)$ is included in the next population, $P(t+1)$
    end if
  end for
until stopping condition(s) satisfied
```

The adaptations of DE required by BinDE introduce an additional computational cost, separate from the issues discussed in section 3.3, as both parent and offspring require the creation of a bit string to be used in the fitness calculation. Unlike BinPSO, where the initial velocity vector should be zeroed, BinDE candidate solutions should be uniformly initialized over the problem search space. If the candidate solutions were all zero vectors, no movement would occur, with difference vectors resulting in zero vectors.

It is very important to note that BinDE has the same bit string generation effect as BinPSO, whereby repeated bit string generation of the same individual may result in different bit strings and fitness values. BinDE is also susceptible to saturation of the sigmoid function. The remainder of the DE is largely unchanged with control parameter values mirroring those of the continuous DE.

Normalization differential evolution

NormDE is a strategy similar to BinDE, but does not alter the behavior nor the principles of the DE algorithm. NormDE defines an alternative mechanism to generate a bit string, whereby each continuous value within the target individual, or generated offspring, is normalized into the range $[0, 1]$. The normalization, to represent a probability value, is calculated as

$$u_{ij}(t) = \frac{x_{ij}(t) - t_{u,min}}{t_{u,max} - t_{u,min}}(t_{s,max} - t_{s,min}) + t_{s,min} \tag{3.16}$$

where $t_{s,max}$ and $t_{s,min}$ are the target scaled maximum and minimum values, with $t_{u,max}$ and $t_{u,min}$ the maximum and minimum unscaled values. The scaling process transforms the range $[t_{u,min}, t_{u,max}]$ into the range $[t_{s,min}, t_{s,max}]$. These normalized values are then used to determine the bit string solution as follows:

$$g_{ij}(t) = \begin{cases} 0 & \text{if } u_{ij}(t) < 0.5 \\ 1 & \text{otherwise} \end{cases} \tag{3.17}$$

As with BinDE, a computational overhead occurs for the translation of the candidate solution into the binary solution. Outlier values within the continuous-valued representation can skew the normalization process, resulting in most normalized probabilities being grouped around a single probability value. Additionally, the bit string generation within NormDE suffers the same bit string generation problems as BinPSO (discussed in section 3.5.2) and BinDE.

As with BinDE, the DE algorithm is largely unchanged, with the fitness calculation being the only change required. Algorithm 9 provides pseudo-code for the NormDE.

Algorithm 9 Normalization differential evolution.

```
Let $t = 1$ be the generation counter
Create and initialize the population, $P(0)$
repeat
    for each parent individual, $\mathbf{x}_i(t)$ do
        Generate a trial vector, $\mathbf{u}_i(t)$, by applying the mutation operator
        Generate an offspring, $\mathbf{x}'_i(t)$, by applying the crossover operator
        Generate bit string, $\mathbf{g}_i(t)$, for the individual, $\mathbf{x}_i(t)$, using equation (3.17)
        Generate bit string, $\mathbf{g}'_i(t)$, for the offspring, $\mathbf{x}'_i(t)$, using equation (3.17)
        if $f(\mathbf{g}'_i(t))$ is better than $f(\mathbf{g}_i(t))$ then
            $\mathbf{x}'_i(t)$ is included in the next population, $P(t+1)$
        else
            $\mathbf{x}_i(t)$ is included in the next population, $P(t+1)$
        end if
    end for
until stopping condition(s) satisfied
```

3.5.4 Artificial bee colony

Basturk *et al.* [7] and Karaboga *et al.* [78, 79, 80] developed the ABC. Because the ABC algorithm (refer to section 2.3.2) uses vector algebra in the equations to employ onlooker and employed bees, the original ABC algorithm is only applicable to continuous-valued problem spaces. The same strategies used to adapt the DE [44] to solve binary optimization problems (refer to section 3.5.3) can be used to adapt the original ABC algorithm to solve binary-valued optimization problems. These binary ABC algorithms [110] are discussed in this section, including issues caused by the adaptations.

Binary artificial bee colony

In a similar fashion to BinDE, binary artificial bee colony (BinABC) is adapted by borrowing concepts from BinPSO. The artificial bees use a continuous-valued vector representation, which is used as a vector of probability values. These probability values are translated into a bit string using equation (3.15). This results in the same issues pointed out for BinDE, in that the objective of BinABC is now to evolve optimal bit

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

flipping probabilities.

BinABC incurs a computational cost to generate bit strings for fitness evaluation, together with the possible saturation of the sigmoid function. Initial positions for food sources should be placed uniformly throughout the search space, because bees exploit the area around food sources in the search space. All binary representational issues, discussed in section 3.3, are still concerns with Hamming cliffs potentially preventing the exploration of the search space around a food source location, requiring a number of bit flips to find a location adjacent to the current food source.

Algorithm 10 provides pseudo-code for the BinABC algorithm, where the only algorithmic change is the addition of a bit string generation required for fitness evaluation.

Algorithm 10 Binary artificial bee colony

```
Create and initialize a $n_x$-dimensional swarm
repeat
    Place employed bees on food sources
    Generate a bit string using equation (3.15), and evaluate the fitness
    Determine probabilities for onlooker bees and place on food sources
    Employed bees perform a local search to refine food source
    Abandon exhausted food sources
    Send out scouts to determine new food sources
until stopping condition(s) satisfied
```

Normalization artificial bee colony

Normalization artificial bee colony (NormABC) is similar to BinABC, where the continuous variables of an artificial bee determine the generated bit string. The fitness of the generated bit string is evaluated using the binary-valued optimization problem, and is then associated with the artificial bee.

The candidate solution, $\mathbf{v}_i$, produced using equation (2.41) is normalized using equation (3.16) and a corresponding bit string is generated using equation (3.17). This bit string is used to evaluate the fitness of the candidate solution, which is then associated with the artificial bee. The generated bit string also experiences similar problems regarding computational cost as with NormDE, together with binary representational issues

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

not being addressed. Algorithm 11 provides pseudo-code for NormABC.

Algorithm 11 Normalization artificial bee colony

```
Create and initialize a $n_x$-dimensional swarm
repeat
    Place employed bees on food sources, using equation (2.41)
    Normalize employed bee using equation (3.16)
    Generate a bit string for employed bee using equation (3.17), and evaluate fitness
    Determine probabilities for onlooker bees and place on food sources
    Employed bees perform a local search to refine food source
    Abandon exhausted food sources
    Send out scouts to determine new food sources
until stopping condition(s) satisfied
```

3.6 Summary

The focus of this chapter was to discuss existing continuous algorithms that have been adapted to solve binary-valued optimization problems. The chapter started with a discussion of issues introduced by using binary representations for candidate solutions and to translate continuous-valued variables to bit strings. Existing algorithms specifically developed for binary-valued problem spaces were described, including problems that these algorithms experience in binary-valued problem spaces. Algorithms developed specifically for continuous-valued spaces were then discussed with a focus on how these algorithms have been adapted to solve binary-valued optimization problems. Problems cause by such adaptations were also discussed.

While these algorithms have been shown to be able to solve binary-valued optimization problems, the introduced problems are of great concern. The next chapter introduces a new novel approach in which continuous algorithms can be used to solve binary-valued optimization problems without adapting the algorithms in any way.

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Chapter 4

Angle Modulation

Previous chapters reviewed optimization algorithms for binary-valued problem spaces. While these binary algorithms showed good performance in the literature, binary representations have a number of concerns, as discussed in previous chapters. Many efficient continuous algorithms exist, which are not subjected to these issues. This chapter proposes an approach to apply these efficient continuous algorithms to solve problems with binary decision variables, without the need to change any aspect of the algorithm nor the representation scheme used for the candidate solution.

4.1 Introduction

For most continuous algorithms, it is not possible to apply these algorithms to binary-valued optimization problems without changing the operators of the algorithm to be applicable to binary representations. As discussed in Chapter 3, such operator changes usually implies a change in the foundational principles on which the algorithm is based.

This chapter proposes a homomorphous mapping [87] as a means to map a binary-valued space to a continuous-valued space. The problem is then solved in the continuous-space with candidate solutions mapped back into binary-valued space. Such a homomorphous mapping have the following advantages:

- The issues related to binary representations (as discussed in section 3.3) are no longer applicable, because solutions are found within a continuous-valued space.
- There is no need to change any aspect of the continuous algorithm used.n

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Angle modulation is proposed as the homomorphous mapping.

The rest of this chapter is organized as follows: Section 4.2 discusses signal processing, from which the angle modulation concept has been borrowed. Section 4.3 discusses the angle modulation homomorphous mapping, enabling the mappings between problem spaces. A number of population-based algorithms that utilize angle modulation as homomorphous mapping are presented in Section 4.4. Section 4.5 concludes the chapter.

4.2 Signal processing

Since angle modulation has signal processing as its basis, this section provides a short overview of signal processing. Within the telecommunications industry [19], a sender entity transfers information to a receiver entity. There is a need to transfer digital signals between entities, but the communication medium available within traditional telecommunication systems is analog. It is necessary that a digital, binary-valued signal be transformed into a continuous-valued analog signal for transmission across the transmission medium to the awaiting receiver. The receiver then has to translate this analog signal back into a digital signal. A transmission signal transmits information across an analog medium, also known as a carrier, from sender to receiver. The transmission signal is a wave form which encodes digital information. Modulation is the process of encoding information within the transmission signal, with the inverse process, decoding the signal, known as demodulation.

Using radio signals as an example for the modulation and demodulation process, a frequency modulation (FM) radio [1] tunes into a specific radio frequency to obtain a broadcast. The frequency defines the base, or baseband, signal for the application of modulation. In order to transmit an information message, $x_m(t)$, on a sinusoidal carrier, x_c, the information message has a restricted amplitude, or height, of one, i.e. $|x_m(t)| \leq 1$.

The sinusoidal carrier is then defined as

$$x_c(t) = A_c \cos(2\pi f_c t) \tag{4.1}$$

where f_c is the baseband frequency of the signal and A_c is the amplitude at the given time step t. The modulated signal, $y(t)$, encodes the message $x_m(t)$ into the baseband

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

signal by

$$\begin{aligned} y(t) &= A_c \cos\left(2\pi \int_0^t f(\tau)d\tau\right) \\ &= A_c \cos\left(2\pi \int_0^t [f_c + f_\Delta x_m(\tau)]d\tau\right) \end{aligned} \tag{4.2}$$

where the *instantaneous frequency*, $f(\tau)$, is the product of $x_m(t)$ and a frequency deviation, f_Δ, representing the maximum shift from the baseband f_c in one direction.

Demodulation extracts the original signal from the transmitted signal, removing the baseband. The demodulation process for FM requires two oscillators, with the phase of one oscillator shifted by 90 degrees [52]. The shifted oscillator, referred to as the *quadrature component*, together with the unshifted in-phase oscillator then extract the original signal. Such a modulation process, whereby the angle, or phase, of the carrier is altered is known as *angle modulation*.

The modulation and demodulation processes result in a bidirectional mapping [53, 55] between the digital and the analog spaces. Simplistically, the transformations between spaces are then

$$f : A \rightarrow B \tag{4.3}$$

$$f^{-1} : B \rightarrow A \tag{4.4}$$

where A is the digital space and B is the analog space. The function f defines the mapping process and f^{-1} defines the *dual*. Together, the functions define a *homomorphous mapping* between the spaces.

4.3 Angle modulation

Using a mapping process, as discussed in the previous section, to translate between continuous and binary spaces, results in angle modulation implementing a $\mathbb{R}^n \rightarrow \mathbb{B}^n$ homomorphous mapping.

The proposed angle modulation function is:

$$g(x) = \sin(2\pi(x - a) \times b \times \cos(2\pi(x - a) \times c) + d \tag{4.5}$$

where x is the input value requiring translation, and parameters a, b, c and d influence the shape of function g as follows:

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

- a determines the amplitude of function g,
- b determines the frequency, or period, of the sin function in function g,
- c determines the frequency, or period, of the cos function in g, and
- d determines the vertical shift of g.

Figure 4.1 illustrates g for $a = 0, b = 1, c = 1$ and $d = 0$.

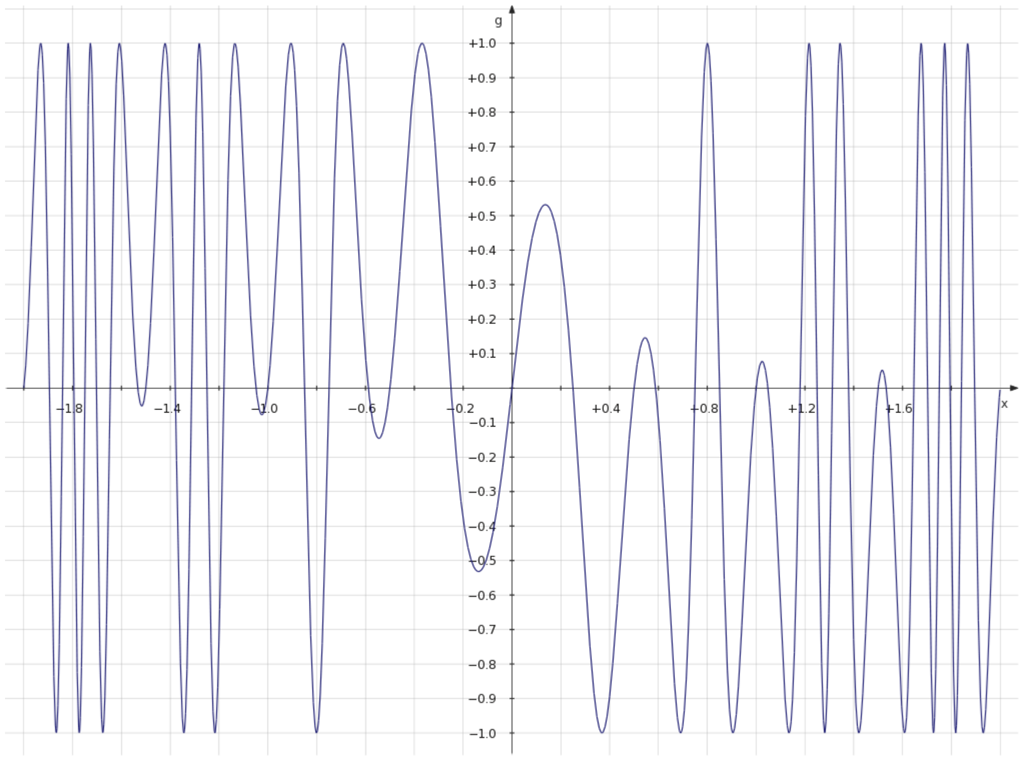


Figure 4.1: Angle modulation generating function $g(x)$ with default parameters

From the principles of angle modulation, function g is proposed in this thesis as a homomorphous mapping between a four-dimensional continuous-valued space and an n-dimensional binary-valued space,

$$g : \mathbb{R}^4 \to \{0, 1\}^n \tag{4.6}$$

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

where n is the required number of bits for the candidate solution in binary-valued space. A binary-valued candidate solution is obtained by sampling g at equal intervals (refer to Section 4.4).

In order to solve a binary-valued optimization problem using angle modulation, the problem is therefore reduced to finding values for the parameters a, b, c and d that will produce an optimal bit string generating function, g.

4.4 Angle modulated population-based algorithms

The homomorphous mapping defined in section 4.3 now allows algorithms, developed for continuous-valued spaces, to solve binary-valued problems. By using the continuous-valued optimization algorithm to find the best set of parameters for the bit string generating function in equation (4.5) in continuous-valued space, the binary optimization problem is then solved indirectly through the adjustment of the bit generating function.

Each set of continuous-valued parameters, substituted into equation (4.5), define a candidate solution within the $\mathbb{R}^4$ continuous space, where $\vec{x}_i^{\mathbb{R}} = (a_i, b_i, c_i, d_i)$. The bit generating function for each candidate solution in the continuous-valued space is then $g(\vec{x}_i^{\mathbb{R}})$, which is used to produce a candidate solution within the binary-valued space, $\vec{x}_i^{\mathbb{B}}$, as follows:

$$x_{ij}^{\mathbb{B}} = \begin{cases} 1 & \text{if } g(\vec{x}_i^{\mathbb{R}}, j) \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.7}$$

where $g(\bullet, j)$ means the output of $g(\vec{x}_i^{\mathbb{R}})$ sampled at position j, as illustrated in figure 4.2. If samples are taken in increments of 0.1 on the x-axis, the following example bit string is generated:

Interval:	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0	1.1
Generated bit:	1	1	1	0	0	1	0	0	1	0	1	0

The binary candidate solution consists of n bits, requiring n sample points, with each point selected at a fixed interval from zero. The fitness of the binary-valued candidate solution is then determined as $f(\vec{x}_i^{\mathbb{B}})$, where f is a fitness function defined within the binary-valued space, and the calculated fitness value is then associated with the candidate solution, $\vec{x}_i^{\mathbb{R}}$, in continuous-valued space. Algorithm 12 provides a generic population-based AM algorithm.

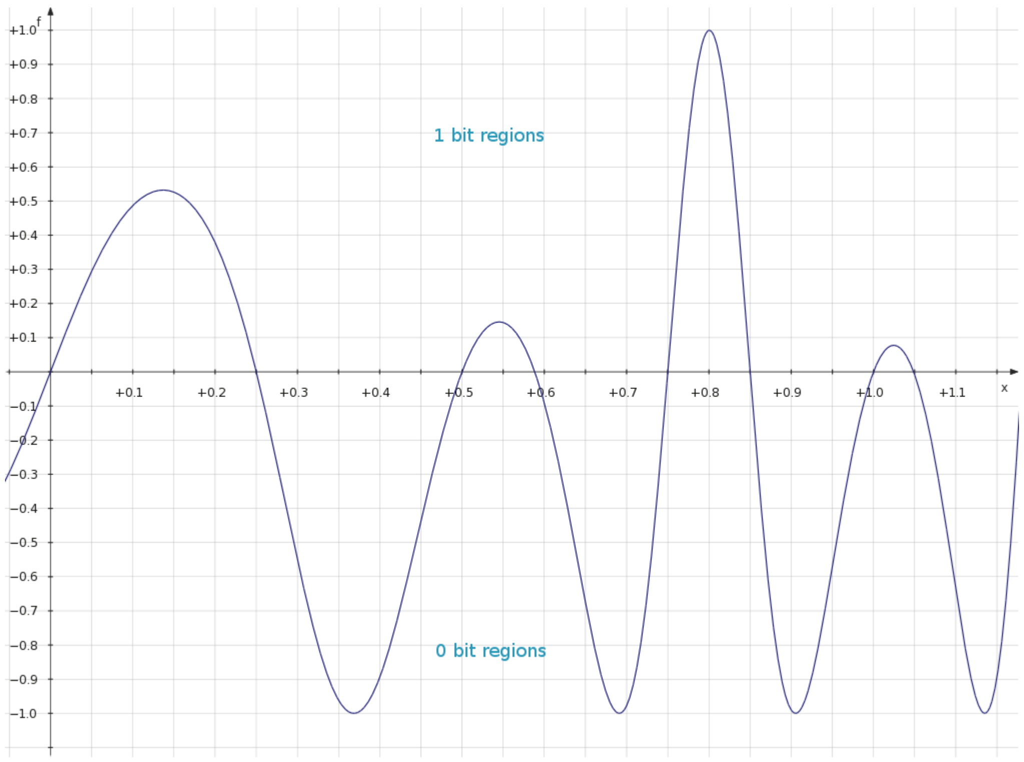


Figure 4.2: Allocation of bit values from generating function.

The remainder of this section proposes a number of new angle modulated population-based algorithms to solve binary-valued optimization problems. Each angle modulated algorithm is obtained by simply replacing the population-based algorithm in Algorithm 12 with a specific continuous population-based algorithm. It is now possible that any n-dimensional binary-valued problem can be solved in a 4-dimensional continuous-valued space.

4.4.1 Angle modulated genetic algorithm

By replacing the population-based continuous algorithm in Algorithm 12 with any continuous GA an angle modulated genetic algorithm (AMGA) is obtained. Using a continuous GA has the advantage that efficient operators developed for continuous-valued representations, can now be used to solve a binary-valued optimization problem. For example, the blending crossover operators such as parent centric crossover [36] and mutation oper-

Algorithm 12 Generic Angle Modulated Population-based Algorithm

```
Perform required initialization for the population-based continuous algorithm
repeat
  Perform an iteration of the population-based continuous algorithm
  for each candidate solution, $\vec{x}_i^{\mathbb{R}} = (a_i, b_i, c_i, d_i)$ do
    Substitute the parameters $a_i$, $b_i$, $c_i$ and $d_i$ into function $g$ to create the bit string generating function
    Generate a bit string of $n$ bits, $\vec{x}_i^{\mathbb{B}}$, using equation (4.5) at equally spaced sample points
    Evaluate the fitness, $f(\vec{x}_i^{\mathbb{B}})$, using the fitness function defined for the binary-valued optimization problem
    Let $f(\vec{x}_i^{\mathbb{R}}) = f(\vec{x}_i^{\mathbb{B}})$
  end for
until stopping condition(s) satisfied
```

ators that sample mutational step sizes from some probability distribution [62, 69, 156] can now be applied.

4.4.2 Angle modulated evolutionary programming

Substitution of the population-based continuous algorithm in Algorithm 12 with a continuous EP allows for more efficient EPs to be used to solve binary-valued optimization problems. Efficient mutation operators, such as the Gaussian, Cauchy and Exponential mutation, are now applicable, yielding solutions within a binary-valued space. Additionally, self-adaptive strategy parameters are also now available to a binary-valued problem space. The resulting algorithm is referred to as angle modulated evolutionary programming (AMEP).

4.4.3 Angle modulated differential evolution

Differential evolution was originally developed to solve continuous-valued optimization problems only. As discussed in Chapter 3, versions of DE were developed to solve binary-valued optimization problems directly. However, as indicated, these DE variations also suffer from the issues discussed in section 3.3. By replacing the population-based continuous algorithm in Algorithm 12 with a continuous DE, binary-valued problems can now be solved using efficient continuous DE strategies without suffering from the issues discussed in section 3.3. This includes the self-adaptive strategies in [108]. The resulting algorithm is referred to as angle modulated differential evolution (AMDE).

4.4.4 Angle modulated particle swarm optimization

As with DE, PSO was originally developed to solve continuous-valued optimization problems. However, replacing the population-based continuous algorithm in Algorithm 12 with a PSO, binary-valued optimization problems can now be solved. This is done without having to change any aspect of the PSO algorithms. No specialized schemes are needed to convert continuous-valued particles to binary representations as is done in the BinPSO (refer to section 3.5.2). The resulting algorithm is referred to as angle modulated particle swarm optimization (AMPSO) [113].

4.4.5 Angle modulated artificial bee colony

Similarly to DE and PSO, the original ABC algorithm was not designed to solve binary-valued optimization problems. While Section 3.5.4 does propose versions of ABC that do apply to binary-valued spaces, these algorithms also change the fundamental principles of ABC. Instead, the continuous ABC algorithm can be substituted into Algorithm 12, and then be used to solve binary-valued optimization problems without any modification to ABC. The resulting algorithm is referred to as angle modulated artificial bee colony (AMABC).

4.4.6 Angle modulated benefits

In addition to the advantages of each angle modulated approach as highlighted above, the proposed angle modulated algorithms also address other problems related to binary representations:

- **Hamming cliffs:** Creation of these cliffs is no longer a concern as the bit string is not the direct solution to the binary-valued optimization problem. The optimization problem is solved in continuous-valued space, and not in binary-valued

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

space. The bit string is obtained by sampling the function, g, where the coefficients represent the candidate solution.

- **Loss of precision:** Binary representations of continuous-valued decision variables is no longer necessary. This results in no loss of precision for the continuous-valued decision variables, and prevents discretization of the search space. Preventing search space discretization allows for an infinite number of possible candidate solutions and not only a finite number, which would be the case if the search space were discretized.
- **Curse of dimensionality:** Four continuous-valued decision variables are needed for the bit string generating function g. From this function, n bits may then be obtained by sampling the function. Because the candidate solution is only 4-dimensional, the issues related to increasing dimension are not applicable.

4.5 Summary

This chapter proposed angle modulation as a homomorphous mapping between continuous-valued and binary-valued spaces. By incorporating angle modulation into algorithms developed to solve continuous-valued optimization problems, a set of new, angle modulated population-based continuous algorithms have been developed to directly solve binary-valued optimization problems without changing any aspect of the continuous algorithm. The proposed angle modulated algorithms do not suffer the issues with reference to binary representations (discussed in section 3.3) and algorithm adaptations (refer to section 3.5). The next chapters provide an empirical analyses of these angle modulated algorithms.

Chapter 5

Experimental Approach

This chapter discusses the experimental setup and design used in Chapter 6 to evaluate the performance of the angle modulated algorithms. The algorithms are evaluated on a suite of scalable benchmark functions.

5.1 Introduction

In order to evaluate the algorithms proposed in this thesis, it is necessary to define a set of benchmark optimization problems and the performance measurements to be used to quantify the performance of these algorithms. This is respectively done in Sections 5.2 and 5.3. All the algorithms used in this thesis have a number of control parameters. Section 5.4 lists the values used for the control parameters of each algorithm. The statistical procedure used in Chapter 6 is described in Section 5.5. Section 5.6 concludes the chapter.

5.2 Benchmark functions

To assess the feasibility of AM as an effective method to provide solutions to binary-valued optimization problems requires different classes of binary-valued optimization problems. This thesis considers the following main categories of problems:

- **Binary mapped continuous-valued functions:** Continuous-valued decision variables of continuous functions are mapped to a binary representation, and solutions to the continuous-valued functions are then found within a binary-valued search space. The found solution is then converted back into a continuous-valued representation.

- **Pure binary functions:** A number of binary optimization problems are included where the decision variables are binary-valued.
- **Random bit strings:** A number of random bit strings are generated, and the purpose of the AM algorithms is to match these bit strings. The randomly created bit strings are used to perform a scalability study in Chapter 6.

The remainder of this section discusses these problems in more detail in the sections that follow.

5.2.1 Binary mapped continuous functions

Table 5.1 lists the continuous-valued functions used in this study, together with the function domain and dimension. Table 5.2 provides the mathematical equation of each benchmark function [145, 159]. Appendix D describes the process used to map decision variables from continuous-valued to binary-valued space.

All the benchmark functions have a global minimum, $f(\mathbf{x}^*) = 0.0$. From the list of continuous-valued benchmark functions, it is important to note that the Griewank function, f_7, becomes simpler as the dimensionality of the function increases [95]. Also, Rosenbrock becomes multi-modal for dimensions larger than four [134]. For the experiments, algorithm iterations were limited to 10000 iterations.

5.2.2 Pure binary functions

Binary problems examined within this study include the Knight's tour (KT), Knight's coverage (KC), N-Queens (Q), the binary-valued multidimensional knapsack problem (MDK) and the iterated prisoner's dilemma (IPD). Similar to the binary mapped functions of Section 5.2.1, algorithm iterations are limited to 10000. The remainder of this section discusses each binary benchmark function, providing binary representations for problem candidate solutions.

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Table 5.1: Continuous-valued benchmark functions

Function	Dimensions	Common name	Function domain	Modality
f_1	30	Spherical	(-5.12, 5.12)	Unimodal
f_2	2	2D Rosenbrock	(-2.048, 2.048)	Unimodal
f_3	5	Step Function	(-5.12, 5.12)	Unimodal
f_4	30	Quartic	(-1.28, 1.28)	Unimodal
f_5	2	Foxholes	(-65536.0, 65536.0)	Multimodal
f_6	2	Schaffer's F6	(-100.0, 100.0)	Multimodal
f_7	30	Griewank	(-300.0, 300.0)	Multimodal
f_8	30	Ackley	(-30.0, 30.0)	Multimodal
f_9	30	Rosenbrock	(-2.048, 2.048)	Multimodal
f_{10}	30	Rastrigin	(-5.12, 5.12)	Multimodal

Table 5.2: Continuous-valued benchmark function equations

f_1	$f_1(\mathbf{x}) = \sum_{i=1}^{n_x} x_i^2$
f_2	$f_2(\mathbf{x}) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2$
f_3	$f_3(\mathbf{x}) = 6 \cdot \sum_{i=0}^{5} \lfloor x_i \rfloor$
f_4	$f_4(\mathbf{x}) = \sum_{i=1}^{n_x} i \cdot x_i^4 + U(0,1)$
f_5	$f_5(\mathbf{x}) = \frac{1}{0.002} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6}$ $a_{ij} = \begin{bmatrix} -32.0, -16.0, 0.0, 16.0, 32.0 \\ -32.0, -16.0, 0.0, 16.0, 32.0 \end{bmatrix}$
f_6	$f_6(x) = 0.5 + \frac{(sin^2\sqrt{x^2+y^2}) - 0.5}{(1.0 + 0.001(x^2+y^2))^2}$
f_7	$f_7(x) = \frac{1}{4000}\sum_{i=1}^{n_x}(x_i - 100)^2 - \prod_{i=1}^{n_x} cos(\frac{x_i - 100}{\sqrt{i}}) + 1$
f_8	$f_8(x) = -20 \cdot \exp\Big(-0.2\sqrt{\frac{1}{n} \cdot \sum_{i=1}^{n_x} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n_x} cos(2\pi \cdot x_i)\Big) + 20 + e$
f_9	$f_9(x) = \sum_{i=1}^{n_x} 100((x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$
f_{10}	$f_{10}(x) = \sum_{i=1}^{n_x}(x_i^2 - 10cos(2\pi x_i) + 10)$

Knight's tour

The knight's tour (KT) [63] is a combinatorial optimization problem where the knight chess piece must move on a chessboard, according to the rules of chess, and visit each

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

square on the board exactly once. The problem is an instance of the Hamiltonian path problem [31] and results in solutions known as *open paths*. If the problem requires the knight to finish the tour on the same square from which it started, the solution results in a *closed path* similar to the Hamiltonian cycle problem [31]. Variations to the problem [115] involve different sized chessboards, including regular chessboards of different sizes and irregular (non-rectangular) chessboards.

Candidate solution representations for the KT problem within the binary-valued problem space consists of the legal moves a knight may make on the chessboard, as illustrated in Figure 5.1, where three bits represent a single move from the knight's eight valid movements. The candidate solution then represents a valid tour as a sequence of moves, with the length of the candidate solution equal to $3n$ bits, where n is the required number of moves to traverse the chessboard for a closed path, or $n - 1$ for an open path. Starting from a potentially random initial square, the knight moves around the chessboard where each subsequent move depends on the knight's current position on the chessboard. The tour created by a knight is highly epistatic.

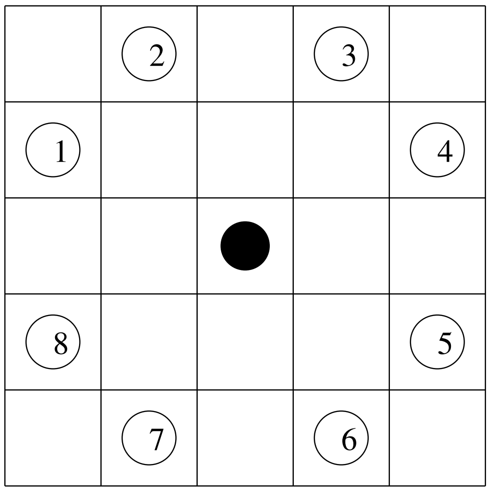


Figure 5.1: Assignment of "moves" for the Knight's Tour. Black circle represents position of Knight, with numbered circles indicating valid Knight moves.

The fitness of the candidate solution is determined by the number of valid moves, up until the point of failure where the Knight could not complete the tour. Valid tours, or solutions, are found when the Knight can complete an entire tour without a movement failure. Invalid tours may apply a repair strategy which attempts to convert and invalid tour into a valid one, as proposed by Gordan and Slocum [63]. This study includes such a repair strategy. Whitley *et al.* [157] investigated the implications of repair strategies during fitness evaluations: The repair strategy enforces valid tours, enforcing that the population does not focus on invalid candidate solutions. This study considers chessboard sizes 4 to 8, resulting in problems KT_4, KT_5, KT_6, KT_7, and KT_8 with respective problem dimensions of 16, 25, 36, 49 and 64.

Knight's coverage

The knight's coverage (KC) problem [46] is a chess based problem which attempts to use a minimal number of knight chess pieces to *cover* a maximal number of chessboard squares. The coverage of a square may be

- knight is on the square, or
- knight may move to the square, or
- square is not covered.

The knights coverage is therefore a placement problem where a minimum number of knights are to be placed on squares such that all squares are covered. The complexity of the problem increases with increase in the size of the chessboard. Figure 5.2 illustrates the placement of knights on a chessboard which is completely covered.

AT&T Research Labs maintains a collection of integer sequences where the sequences represent solutions to combinatoric problems. Within these sequences, sequence A006075 [130] maintains solutions to the knight's coverage problem for chessboard sizes ranging from 1×1 to 26×26. The sequence's author states the solutions to chessboards greater in size than 15×15 are not guaranteed to be optimal.

Each binary-valued candidate solution represents a complete chessboard state, where a bit value of 1 denotes that a knight is present on the associated chessboard square. Therefore, the dimension of the binary-valued solution is $n \times n$. A value of 0 indicates that the square does not have a knight. The fitness of a candidate solution is determined using Algorithm 13, where fitness is calculated as a function of the number of covered squares. If a square is not covered, a penalty of 1000 is added to the fitness. If square

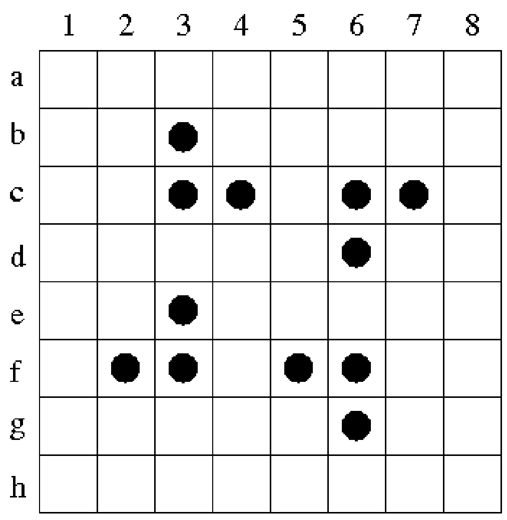


Figure 5.2: Solution to 8×8 Knight's Coverage Problem with a final fitness value of -9200. Black circles indicate positions of Knights.

is covered by the knight itself, a score of 100 is added to the total sum, with squares in positions that the knight may move to reducing the sum by 200. The objective is therefore to minimize the total sum penalties and scores.

Algorithm 13 Proposed fitness evaluation for the knight's coverage problem

```
for each column on the game board do
    for each row on the game board do
        if square is covered by a knight then
            $fitness = fitness + 100$
        else if square is not covered by a knight, but is reachable by another knight then
            $fitness = fitness + 1000$
        else
            $fitness = fitness - 200$
        end if
    end for
end for
```

Franken and Engelbrecht [54] provided solutions to the knight's coverage problem using PSO. For chessboard sizes 4 to 8, the studied problems are KC_4, KC_5, KC_6, KC_7

and KC_8, resulting in dimensions of 16, 25, 36, 49, and 64 respectively.

n-Queens

On a chessboard with sides of n squares, the aim of the n-Queens (Q) problem is to place n queens on the chessboard such that no queen may "take" another queen, based on the rules of chess. The n-Queens problem is also a placement problem where exactly n queens must be placed on the chessboard, and is also a constraint satisfaction problem.

Placement of queens onto the chessboard becomes more difficult with an increase in chessboard size. Different meta-heuristics have been used to find optimal placement of the queens [32, 37, 99, 150]. However, all techniques present sub-optimal solutions with an increasing chessboard size. Figure 5.3 provides a sample solution to an n-Queens problem with eight queens.

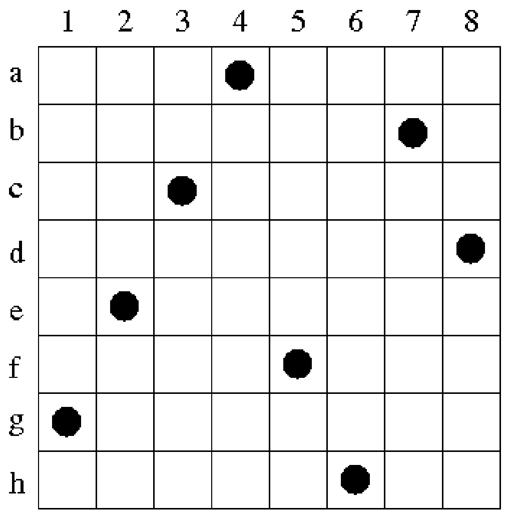


Figure 5.3: Example of Queen placement on a 8×8 board. Black circles represent placement of Queens.

A candidate solution represents a possible chessboard state, using a bit string. To represent the chessboard state, a bit within the candidate solution represents a queen on the board when the bit has a value of 1. A value of 0 indicates that no queen exists on the particular chessboard square.

Algorithm 14 is used to calculate the fitness of a candidate solution. To satisfy the

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

primary constraint that exactly n queens must exist on the chessboard, any solution not providing n queens, i.e. n bits with a value of 1, is regarded as invalid. A large fitness score is associated with invalid candidate solutions.

Solutions to the n-Queens problem can be found in [37, 99, 150]. Chessboard 4 to 8 are only considered, resulting in problems Q_4, Q_5, Q_6, Q_7 and Q_8 and dimensions of 16, 25, 36, 49, and 64 respectively.

Algorithm 14 Proposed fitness function for n-Queens problem

```
Initialize chessboard from provided bit string
$fitness = 0$
if number of queens, $n_q \neq n$ then
  $penalty = 30000$
else
  $penalty = 0$
end if
for each row on the chessboard do
  for each column on the chessboard do
    if square on chessboard contains queen then
      $fitness$ = sum of rule violations + $penalty$
    end if
  end for
end for
```

Binary multi-dimensional knapsack

Also known as the backpack or rucksack problem, the multidimensional knapsack problem (MDK) [125] is a constraint satisfaction problem where a container, the knapsack, holds a finite amount of weight. Objects are placed into the knapsack, each with an associated weight and a positive profit value. The aim of the knapsack problem is to place objects into the knapsack, filling the knapsack, whilst maximizing the profit associated with the objects. More formally, the objective is to

$$\max \sum_{i=1}^{n} p_i x_i \tag{5.1}$$

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

where p_i defines the profit associated with the weight, w_i, of object i, whilst maintaining

$$\sum_{i=1}^{n} w_i x_i \leq c \tag{5.2}$$

where $x_i \in \{0, 1\}$ and c is the capacity of the knapsack. The knapsack problem is NP-hard [67], although in a weak sense [122] because the problem is solvable in pseudo-polynomial time through dynamic programming [8].

The candidate solution for the multidimensional knapsack problem represents all objects available for inclusion into a knapsack, i.e. n objects in total. Objects selected for inclusion have a bit value of 1 and unselected objects have a bit value of 0. The fitness function for the knapsack is

$$f = \begin{cases} \sum_{i=1}^{n} p_i x_i & \text{if } \sum_{i=1}^{n} w_i x_i \leq c \\ -1 & \text{otherwise} \end{cases} \tag{5.3}$$

Therefore, the profit directly determines the fitness for a candidate solution, which is then maximized. If the capacity of the knapsack is exceeded, the solution is not valid and is assigned an negative fitness value.

For the purposes of this study, the two experimental problem instances [163] are

- **10 items:** The defined capacity of the knapsack is 269. The objects placed within the knapsack have the following weights and profit values

  $$\mathbf{w} : \{95, 4, 60, 32, 23, 72, 80, 62, 64, 46\}$$
  $$\mathbf{p} : \{55, 10, 47, 5, 4, 50, 8, 61, 85, 87\}$$

  The target fitness value for the problem, referred to as MDK_{10} is 295.

- **20 items:** The capacity of the knapsack is 878, with objects having weight and profit values of

  $$\mathbf{w} : \{92, 4, 43, 83, 84, 68, 92, 82, 6, 44, 32, 18, 56, 83, 25, 96, 70, 48, 14, 58\}$$
  $$\mathbf{p} : \{44, 46, 90, 72, 91, 40, 75, 35, 8, 54, 78, 40, 77, 15, 61, 17, 75, 29, 75, 63\}$$

  The target fitness value of MDK_{20} is 1024.

The multidimensional knapsack problem has successfully been solved by Zhang and Wei [161], Peng *et al.* [120], Raidl [126], Zhao *et al.* [162], Guo *et al.* [65] and Shao *et al.* [135].

Iterated prisoner's dilemma

The prisoner's dilemma (PD) problem [123] is a fundamental problem within game theory which illustrates the ideals of self-preservation, even when it is more beneficial to cooperate. The problem is a game where the police arrest two suspects. The police do not have enough evidence to ensure a conviction, so each suspect is independently visited to propose a *deal*. The police provide each suspect with two choices: (1) betray the other suspect, or (2) remain silent. Each suspect has no knowledge of the actions taken by the other until the investigation concludes. The question is then, how should the suspects act? The following scenarios are possible:

- **A betrays B, B is silent:** If suspect A betrays suspect B by assisting the police, suspect B will go to prison whilst letting suspect A go free.
- **A and B are silent:** Both suspects remaining silent would result in both suspects going to prison for a limited time and are then also liable to pay a fine.
- **A and B betray each other:** Both suspects go to prison, but for a shorter period than if one betrayed the other.

Playing the PD repeatedly for a number of different rounds results in the iterated prisoner's dilemma (IPD) [2, 39]. Different strategies have been developed for the IPD [3, 15, 72, 73] with the most common being *All-C*, where each suspect cooperates regardless of the actions of the other suspect, and *All-D* where each suspect always defects. Axelrod [2] performed numerous experiments, encouraging other researchers to suggest strategies for the IPD. A successful strategy, known as tit-for-tat [3] developed from a human-centered competition, simply repeats the decision made by the other suspect in the previously played game.

The aim of the problem is to devise a strategy for the game. The game is divided into different rounds which are individual bits within the candidate solution of length n_x, implying n_x different rounds. For each round the player may either chose to cooperate (bit value 1) or to defect (bit value 0). To determine the effectiveness of the candidate solution representing the game strategy, it is compared against every other candidate solution. The fitness of a candidate solution is determined by a pay-off matrix (Table 5.3) where each candidate solution is awarded points for the decision in the current round.

Axelrod's pay-off matrix [2] shows that the maximum number of points both players can be awarded for each round occurs only if both cooperate.

Table 5.3: Payoff matrix of Axelrod. Tuple indicates the pay-off received by each player when choosing to defect or cooperate.

	Cooperate	Defect
Cooperate	$(3,3)$	$(0,5)$
Defect	$(5,0)$	$(1,1)$

For the purposes of this study, sixty four consecutive games were considered, resulting in a bit string of sixty four individual bits representing the candidate solution, which is the strategy of a single player within the IPD. A full IPD strategy, consisting of two players, is therefore one hundred and twenty eight bits long. Evaluation of an IPD strategy requires the following:

- **Maximum personal pay-off:** A player attempts to follow a greed strategy and, because the decision of the other player is unknown, always defects to ensure the largest possible personal pay-off. This is known, in this study, as the personal best IPD, or IPD_b.

- **Maximum collective pay-off:** The collective pay-off is the pay-off obtained by the entire population of IPD players. The maximum possible collective pay-off is when both players cooperate. As a result, the target strategy for the population is then to cooperate more, when compared to the personal pay-off strategy. This strategy is referred to as IPD_c.

Consequently, the evaluation of the performance of each IPD strategy takes into account the pay-off from both personal and collective perspectives. The evaluation of the candidate solution from both perspectives allows the personal pay-off to be put into context of the collective pay-off of the entire population, preventing bias to a single perspective. Previous work providing solutions to the IPD include Franken *et al.* [55], Glomba *et al.* [59], Golbeck [60], Tekol *et al.* [146] and Bukhari *et al.* [17].

5.2.3 Matching random bit strings

Random bit strings of increasing length were generated and the coefficients of the generating function g were then evolved to determine if AM would be able to reproduce the random bit string. The fitness of each candidate solution (the AM four-tuple) is calculated using Algorithm 15.

Algorithm 15 Random bit string matching

```
Let $t$ be a randomly generated bit string of length $n$
Let $fitness = 0$
Substitute AM four-tuple into equation (4.5), yielding $g'$
Generate bit string $b$ using a set of intervals and $g'$
for each bit $b_i$ in $b$ do
  if $b_i = t_i$ then
    $fitness+ = 1$
  end if
end for
```

Scalability of the AM approach is expressed as the ability to accurately produce arbitrary bit strings of increasing length. Scalability is therefore quantified as the percentage of bits correctly reproduced by using AM.

5.3 Performance criteria

To determine the quality of solutions produced within the binary problem space, the fitness value of the best solution and the number of iterations are used. Results are reported as averages and standard deviations over 30 independent runs.

Where applicable, the best solution to a problem is also provided in order to aid in understanding the representation of the candidate solution through the use of a graphical representations.

5.4 Algorithm comparisons

This study evaluates a number of AM algorithms and compares their performance with their non-AM binary versions. These algorithms were implementated within the computational intelligence library (CIlib) [63, 111, 118, 151], an open-source Java [64] library. Algorithm control parameters are not optimal, as the purpose of this study is to determine if AM is applicable generally and not only to algorithms with tuned control parameters. This section summarizes the values used as control parameters for each algorithm. The control parameters for both the binary version and AM version remained the same, unless otherwise stated, to allow for unbiased comparisons.

5.4.1 Genetic algorithm

The following GA parameters have been used:

- Population size: 40 individuals
- Cross-over probability: $p_c = 0.5$
- Mutation probability: $p_m = 0.7$
- Uniform cross-over
- Mutation
  - Gaussian Mutation with deviation of 1 (for continuous problems)
  - Random Mutation (for binary problems)

5.4.2 Evolutionary programming

The control parameter settings for the algorithm are as follows:

- Population size: 40 individuals
- Tournament selection with a tournament size of 10 individuals
- Strategy parameters: Initially set to a value of 1

- Update strategy: self-adaptive update (lognormal)
- Mutation
  - Gaussian mutation with deviation of 1 (for continuous problems)
  - Random Mutation (for binary problems)

5.4.3 Differential evolution

Within the DE, the selected control parameters are:

- Population size: 40 individuals
- p_c: 0.25
- Random trial vector selection
- Crossover with binomial crossover point selection
- Scaling factor: $\beta = 1.0$
- Number of difference vectors: 1

The selected value for the crossover probability ensures that at least one dimension within the candidate solution of the angle modulated differential evolution (AMDE) is altered by the recombination operator.

5.4.4 Particle swarm optimization

The experimental work for the PSO focused on comparing the results of the BinPSO (see section 3.4) and angle modulated particle swarm optimization (AMPSO) (see section 4.4.4). The guidelines proposed by van den Bergh [152] provided the selected PSO control parameters used in the initialization and execution of the BinPSO and AMPSO.

The PSO configured parameters are as follows:

- Acceleration coefficients (c_1 and c_2): 1.496180
- Inertia weight: 0.729844

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

- V_{max}: 4.0
- Social network structure: Von Neumann network topology
- Size of swarm (n_s): 40 particles

Studies have been performed to determine the influence of the swarm topology [14, 153]. From these studies, it was decided to use the Von Neumann network topology for the experiments in this study. The Von Neumann topology has shown to be a robust topology, used in a wide range of problems [43, 84, 119].

5.4.5 Artificial bee colony

The following control parameter values were used for angle modulated artificial bee colony (AMABC), BinABC and NormABC:

- Swarm size: 40 bees
- Population split for employed and onlooker bees: 50:50
- Selection strategy: Roulette wheel selection
- Forage limit: 500 iterations

5.5 Statistical procedure

Numerous experiments, where an experiment refers to a single algorithm applied to one problem, have been completed. For each experiment, 30 independent runs of the algorithm on the optimization problem have been performed. A null hypothesis (H_0) and an alternate hypothesis (H_1) have been stated for each experiment. A Mann-Whitney U test [98] was used to determine if the null hypothesis can be accepted at a 95% confidence level. The following symbols denote three possible alternate hypotheses:

1. $\mathbf{H_1^{\neq}}$: Alternate hypothesis identifying that the means of the two observed samples are unequal, $\eta_1 \neq \eta_2$
2. $\mathbf{H_1^{<}}$: Alternate hypothesis identifying that the mean of the first sample (η_1) is less than the mean of the second sample (η_2), $\eta_1 < \eta_2$

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

3. $\mathbf{H_1^{>}}$: Alternate hypothesis identifying that the mean of first sample (η_1) is greater than the mean of the second sample (η_2), $\eta_1 > \eta_2$

The main hypotheses considered for this study, depending on the type of optimization problem, are (for algorithm A and B):

- Minimization: $H_0^{\text{A} = \text{B}}$, $H_1^{\text{A} \neq \text{B}}$, $H_1^{\text{A} < \text{B}}$
- Maximization: $H_0^{\text{A} = \text{B}}$, $H_1^{\text{A} \neq \text{B}}$, $H_1^{\text{A} > \text{B}}$

5.6 Summary

This chapter provided the outline for the experimental work to evaluate the performance of the AM algorithms, in comparison to the binary algorithm versions. A suite of benchmark functions, for both the AM and non-AM algorithms, was then introduced including both continuous-valued functions and binary-valued problems.

The following chapter discusses the results of the experiments, and provides outcomes for the stated hypotheses.

Chapter 6

Results and Comparisons

This chapter presents results obtained by applying the binary and angle modulated algorithms on the benchmark problems listed in Chapter 5. The results are analyzed and answers to the stated hypotheses are given.

6.1 Introduction

Presentation of the results of experiments conducted is done per algorithm paradigm. For each algorithm paradigm, both the results for the binary mapped continuous functions and the pure binary functions are given and discussed. This is done in sections 6.2 to 6.6 respectively for GA, EP, DE, PSO and ABC. These results lead to the main goal of this study, i.e. to conclude on the hypothesis, $H_0^{AM=Non-AM}$, which is done in section 6.7. A scalability study of AM, considering AMPSO, is done in section 6.8. Section 6.9 concludes the chapter.

6.2 Genetic algorithm results

The sections that follow present the experimental results for the AMGA and BinGA. Section 6.2.1 presents the results for the set of continuous-valued minimization problems, and Section 6.2.2 presents the results for the binary benchmark problems.

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

6.2.1 Binary mapped continuous-valued problems

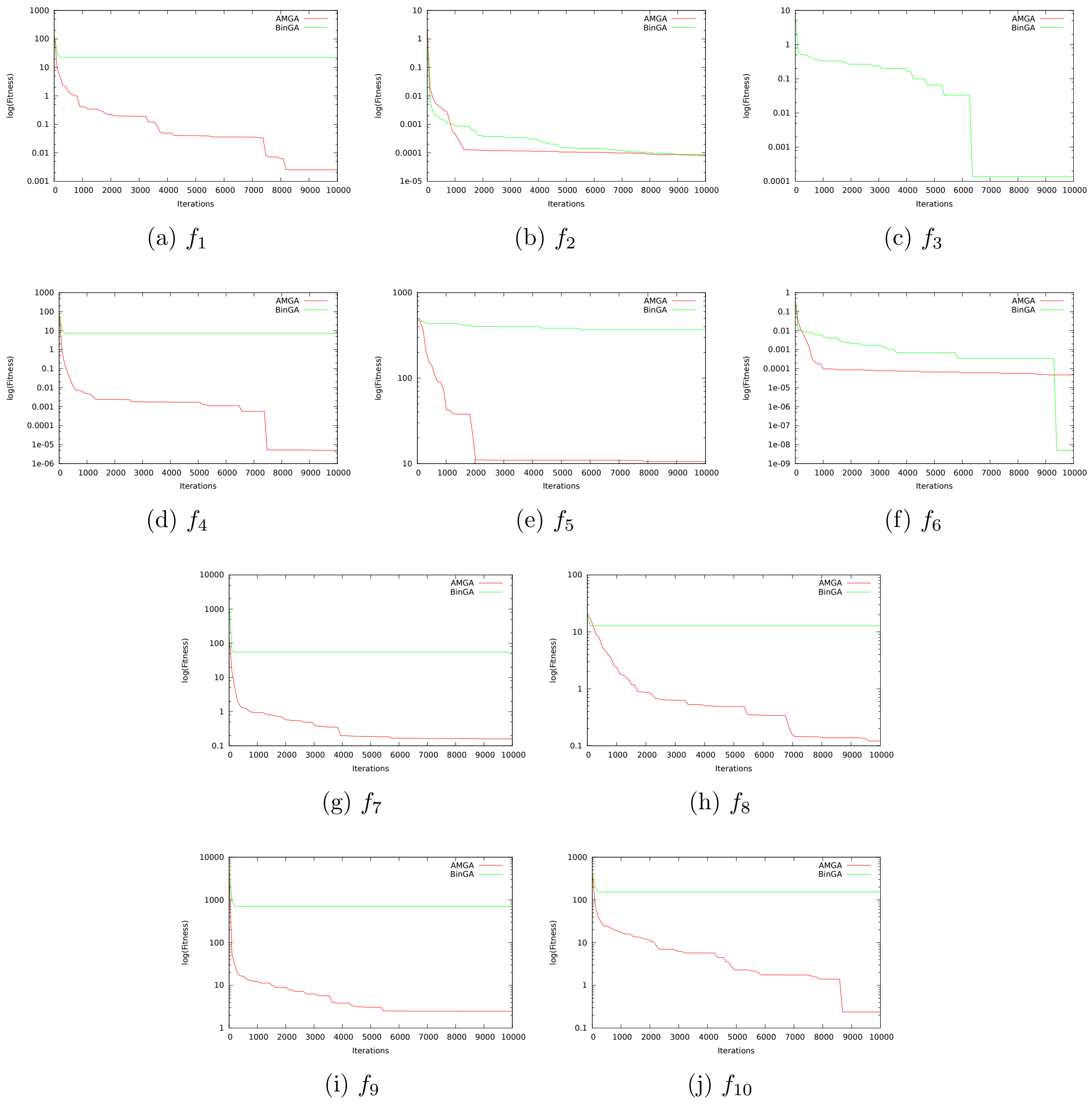
Table 6.1 summarizes the results for the problems listed in table 5.1, and provides outcomes to the formal hypotheses. The null-hypothesis is defined as $H_0^{AMGA=BinGA}$, i.e. that the performances of AMGA and BinGA are statistically the same. The two alternate hypotheses are $H_1^{AMGA \neq BinGA}$, i.e. that the performances of AMGA and BinGA differ with statistical significance, and $H_1^{AMGA<BinGA}$ which states that the performance of AMGA is better than that of BinGA.

From the observed results, the AMGA and BinGA are statistically significantly different for eight of the problems, and for seven of these benchmark problems AMGA is significantly better than BinGA. Benchmark function f_6 indicates that BinGA produced better results than AMGA and is confirmed by the obtained p-value. The remaining functions, f_2 and f_3, show no statistically significant difference with the obtained results (refer to the p-values), with both algorithms providing similar results. Functions f_2 and f_3 are both low dimensional, unimodal problems indicating that the problem search space is not convoluted [95], and it is then reasonable that both algorithms would not have difficulty in finding good solutions. The superiority of AMGA over BinGA for the ten problems is also indicated by the overall rank as given in the last row of Table 6.1.

Table 6.1: Binary-mapped continuous minimization function results for AMGA and BinGA, measured at a confidence of 95%. Bold p-values indicate statistical significance.

f	AMGA	BinGA	p-value	
			$H_1^{\neq}$	$H_1^{<}$
f_1	0.002548 ± 0.008667	23.109983 ± 9.972582	$\mathbf{1.84 \times 10^{-11}}$	$\mathbf{9.21 \times 10^{-12}}$
f_2	0.000085 ± 0.000047	0.000088 ± 0.000101	0.25225	0.876878
f_3	0.0 ± 0.0	0.0 ± 0.0	1.0	1.0
f_4	0.000003 ± 0.000010	7.574234 ± 3.002702	$\mathbf{2.21 \times 10^{-11}}$	$\mathbf{1.10 \times 10^{-11}}$
f_5	10.971792 ± 2.472990	$368.255910 \pm 205.634145$	$\mathbf{2.05 \times 10^{-11}}$	$\mathbf{1.02 \times 10^{-11}}$
f_6	0.000047 ± 0.000051	0.0 ± 0.0	**0.000023**	0.999988
f_7	0.158921 ± 0.329618	56.897319 ± 22.475286	$\mathbf{2.91 \times 10^{-11}}$	$\mathbf{1.45 \times 10^{-11}}$
f_8	0.126729 ± 0.340376	13.155477 ± 2.169059	$\mathbf{2.14 \times 10^{-11}}$	$\mathbf{1.07 \times 10^{-11}}$
f_9	3.025986 ± 6.650647	$728.516896 \pm 258.295984$	$\mathbf{1.27 \times 10^{-11}}$	$\mathbf{6.35 \times 10^{-12}}$
f_{10}	0.247909 ± 0.235457	158.992204 ± 43.822424	$\mathbf{2.11 \times 10^{-11}}$	$\mathbf{1.05 \times 10^{-11}}$
Avg Rank	1.1	1.8		

AMGA was able to achieve solutions faster than the BinGA for nine of the problems, with Figure 6.1 illustrating that AMGA was still able to refine the solutions after stagnating for several iterations.



(a) f_1 (b) f_2 (c) f_3

(d) f_4 (e) f_5 (f) f_6

(g) f_7 (h) f_8

(i) f_9 (j) f_{10}

Figure 6.1: AMGA and BinGA performance on binary-mapped continuous problems.

Figure 6.2 illustrates the generating function for the one problem (f_6) where AMGA failed to achieve better results, indicating that the AMGA produced poor four-tuple

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

values. The figure also shows that almost all bit values should be zero, but the GA was not able to produce a four tuple that would result in a generated all zero bit string.

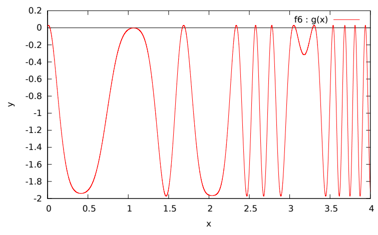


Figure 6.2: AMGA bit string generating function for f_6

6.2.2 Binary-valued problems

This section provides the results for the set of binary benchmark problems, discussing the results for the minimization problems first, followed by the results obtained for the maximization problems. In each section, the goal is to determine if AMGA can outperform BinGA.

Minimization

Table 6.2 summarizes the results for the binary minimization problems. KC results show no real significant difference, confirmed by the p-values, between the AMGA and the BinGA, with the exception of KC_6 where the BinGA outperformed the AMGA. For the set of Q problems, AMGA was able to provide results significantly better than that of BinGA in four of the five problems. The only problem where the AMGA did not provide better results was for the Q_4 problem, where performance was the same as BinGA and Q_4 is a lower dimensional binary problem with a dimension of 16. The large means for problems Q_7 and Q_8 indicate that the BinGA was not able to achieve any valid solutions. Overall, AMGA achieved an average rank of 1.3 which is better than the BinGA rank of 1.6.

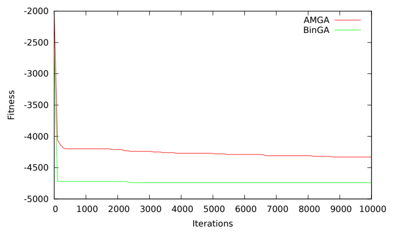
Figure 6.3 provides performance graphs between AMGA and BinGA for selected binary minimization problems to illustrate different trends. BinGA converged onto a

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

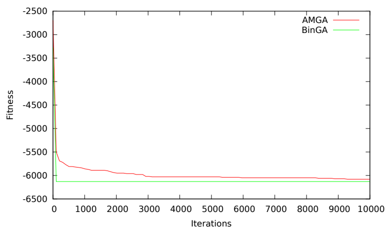
Table 6.2: Binary minimization function results for AMGA and BinGA, measured at a confidence of 95%. Bold p-values indicate statistical significance.

f	AMGA	BinGA	p-value	
			$H_1^{\neq}$	$H_1^{<}$
KC_4	-2000.0 ± 0.0	-2000.0 ± 0.0	1.0	1.0
KC_5	-3500.0 ± 0.0	-3500.0 ± 0.0	1.0	1.0
KC_6	-4330.0 ± 187.818694	-4740.0 ± 122.051430	$\mathbf{9.299 \times 10^{-10}}$	0.999999
KC_7	-6080.0 ± 202.399400	-6130.0 ± 321.794537	0.541488	0.732436
KC_8	-8000.0 ± 0.0	-7870.0 ± 502.510936	0.192766	0.096618
Q_4	0.0 ± 0.0	0.0 ± 0.0	1.0	1.0
Q_5	0.0 ± 0.0	1.733333 ± 1.552158	$\mathbf{2.85 \times 10^{-7}}$	$\mathbf{1.42 \times 10^{-7}}$
Q_6	0.0 ± 0.0	5.933333 ± 2.318342	$\mathbf{9.26 \times 10^{-13}}$	$\mathbf{4.63 \times 10^{-13}}$
Q_7	0.0 ± 0.0	$29000.533333 \pm 5474.304388$	$\mathbf{2.7 \times 10^{-14}}$	$\mathbf{1.35 \times 10^{-14}}$
Q_8	2.0 ± 0.0	30000.0 ± 0.0	$\mathbf{1.68 \times 10^{-14}}$	$\mathbf{8.42 \times 10^{-15}}$
Avg Rank	1.3	1.6		

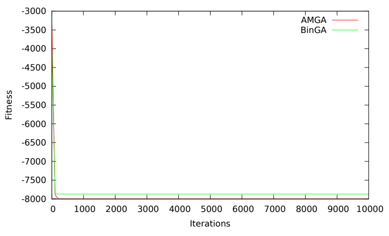
solution faster than the AMGA for most problems, but AMGA was able to refine found solutions.



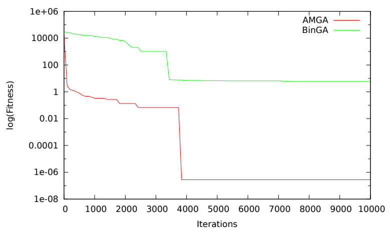
(a) KC_6



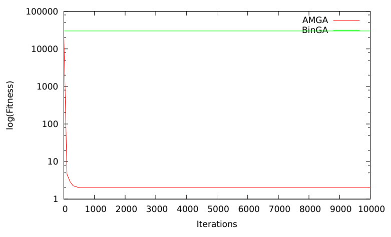
(b) KC_7



(c) KC_8



(d) Q_6



(e) Q_8

Figure 6.3: AMGA and BinGA performance on binary minimization problems.

Maximization

Results for the binary maximization problems are given in Table 6.3. The AMGA performed better than the BinGA for only two of the nine problems. For six of the problems the BinGA performed best, while similar performance was observed for one problem (MDK_{10}). The poor performance of AMGA on the KT and IPD problems indicates that the generating function struggles to produce bits that are reliant on other bit substrings. This is the case with KT and IPD, where the next decision in the game depends on previous decisions. Low mean fitnesses, and p-values confirm the poor performance. The diversity of the AMGA population for the KT and IPD problems remains a large value, indicating that the population has not converged onto a solution. The poor performance of AMGA observed for the IPD results, together with the evolved solutions, confirms that the strategy for the best individual and the collective population was predominately to always defect.

Table 6.3: Binary maximization function results for AMGA and BinGA, measured at a confidence of 95%. Bold p-values are statistically significant.

f	AMGA	BinGA	p-value	
			$H_1^{\neq}$	$H_1^{>}$
MDK_{10}	295.0 ± 0.0	295.0 ± 0.0	1.0	1.0
MDK_{20}	1024.0 ± 0.0	1022.2 ± 3.24791	**0.010425**	**0.005188**
KT_4	11.866666 ± 1.04166	12.433333 ± 0.504006	**0.037954**	0.982877
KT_5	15.766666 ± 0.935260	16.933333 ± 1.080655	**0.000136**	0.999953
KT_6	32.166666 ± 0.379049	33.033333 ± 0.319841	$\mathbf{7.61 \times 10^{-10}}$	0.999999
KT_7	46.0 ± 0.0	45.033333 ± 0.999425	$\mathbf{1.01 \times 10^{-5}}$	$\mathbf{5.08 \times 10^{-6}}$
KT_8	57.433333 ± 1.135123	58.133333 ± 1.041660	**0.00736**	0.995918
IPD_b	4906.666 ± 3225.644	7546.666 ± 598.695	**0.006582**	0.99667
IPD_c	98133.333 ± 64512.887	296606.433 ± 23666.892	$\mathbf{1.4 \times 10^{-11}}$	0.999999
Avg Rank	1.5	1.1		

Figure 6.4 illustrates a generating function for KT_4 which produces a poor performance, compared to the performance of BinGA. The GA failed to produce a four tuple to correctly solve the problem because the problem applies a correction strategy during fitness evaluation to prevent invalid tours. This correction is needed for the binary algorithms, but results in a poor AM performance. Furthermore, AM generates bit strings from left (MSB) to right (LSB) and any bits in the left positions have a much larger

impact on the bit string value, especially if incorrectly generated. The generated bit values may also be incorrect due to the conditional comparison against a sampled random number.

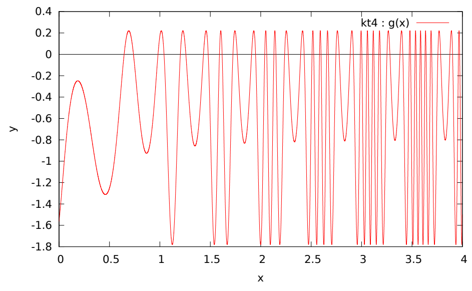


Figure 6.4: AMGA generating function for KT_4

The average ranks shown in Table 6.3 show that BinGA performed better than AMGA for binary maximization problems, with respective ranks of 1.1 and 1.5. The only problems where AMGA performed better were KT_7 and MDK_{20}. Figure 6.5 illustrates the performance for selected binary maximization problems and indicate that there is almost no noticeable difference between AMGA and BinGA performances, except for the IPD problems where a large difference between the algorithms was shown. The population diversity, given in Figure 6.6, fluctuates around a value of 1.2 for AMGA and 4.4 for BinGA, indicating that convergence was too fast and a solution was maintained from an early iteration, without any refinement.

6.2.3 Overall outcome

For the AMGA and BinGA results, the overall performances and average ranks indicate that AMGA is better than BinGA for almost all problems, excluding KT and IPD. Therefore, based on the average ranks, AMGA performed better than BinGA.

6.3 Evolutionary programming results

This section compares the performance of angle modulated evolutionary programming (AMEP) with BinEP. Section 6.3.1 discusses the experimental results for the binary

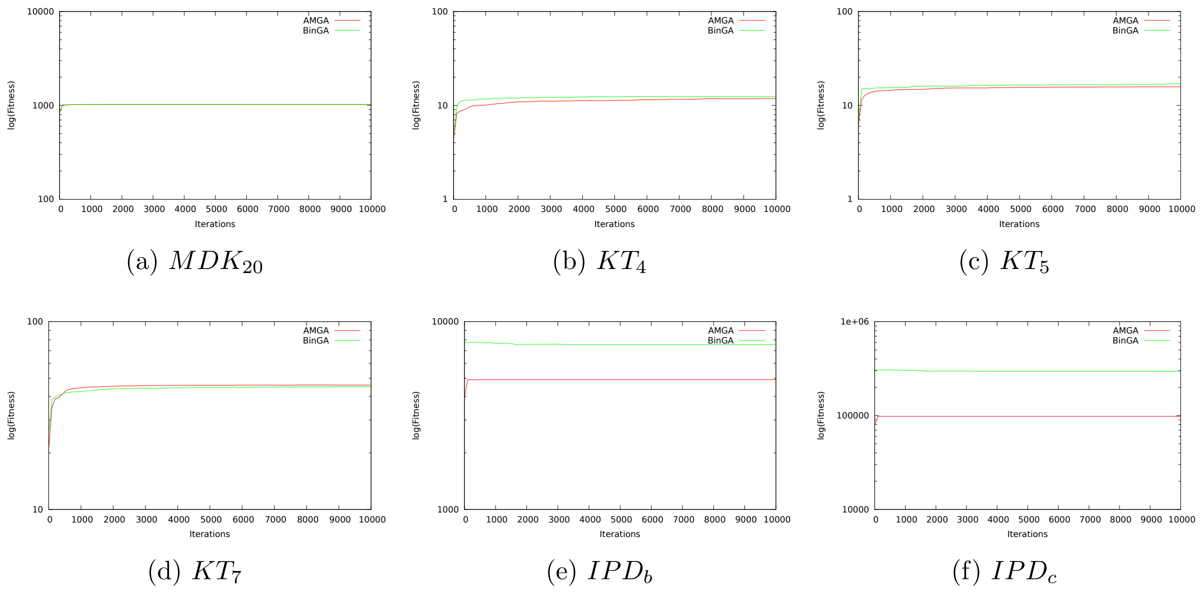


Figure 6.5: AMGA and BinGA performance for selected binary maximization problems.

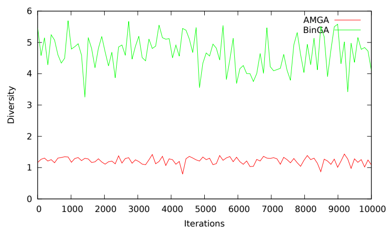


Figure 6.6: AMGA diversity for the IPD problem.

mapped continuous-valued problems, while Section 6.3.2 considers the binary problems.

6.3.1 Binary mapped continuous-valued problems

The results for the binary mapped continuous-valued problems are given in Table 6.4. As evident from the p-values, AMEP provided a better performance than BinEP for nine of the ten problems. Only for f_8 did the BinEP perform better than AMEP, even though mean fitnesses did not deviate much. The p-values indicate that AMEP managed to approach more optimal solutions, unlike BinEP. AMEP acheieved better performances,

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

resulting in an average rank of 1.1 whereas the average rank of BinEP is 1.9.

Table 6.4: Binary-mapped continuous-valued minimization function results for AMEP and BinEP, measured at a confidence of 95%. Bold p-values indicate statistical significance.

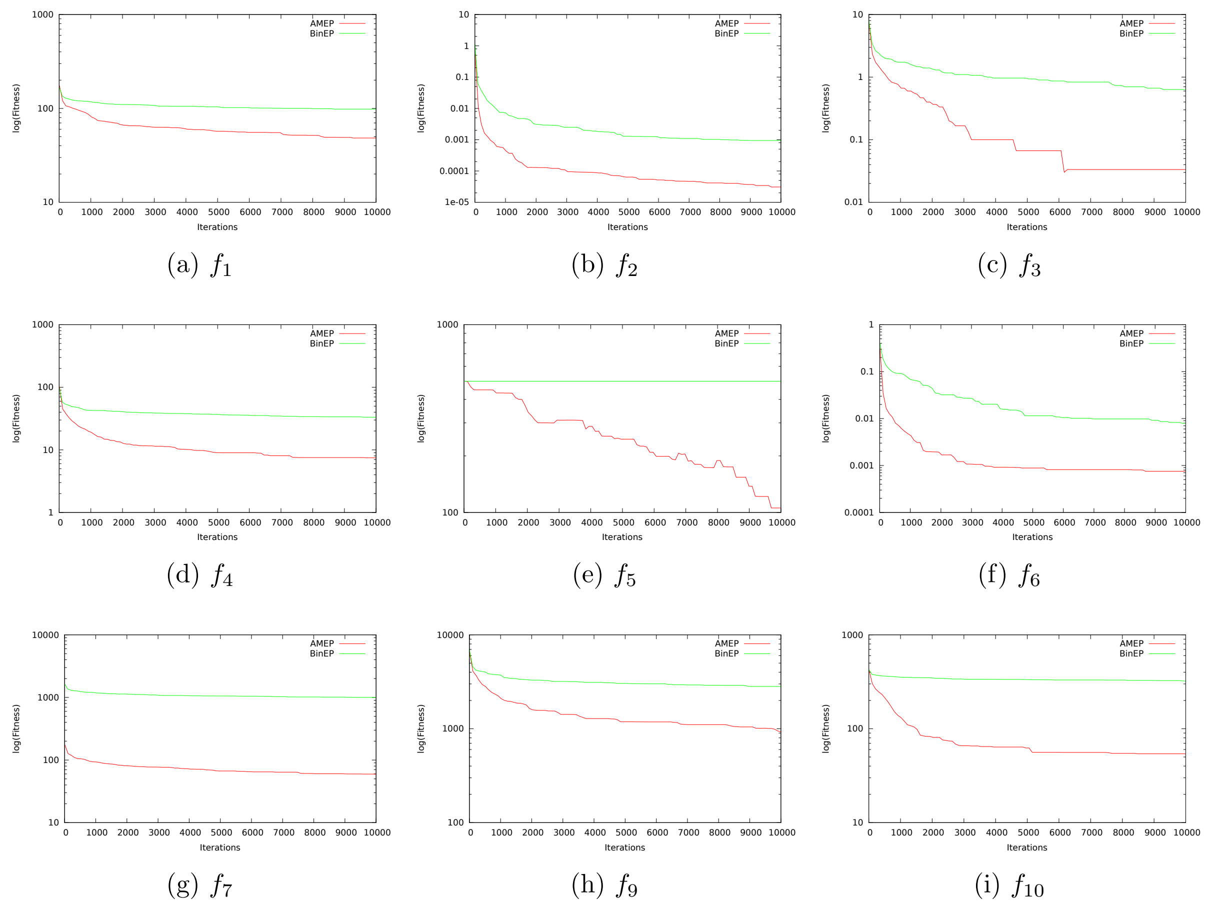
f	AMEP	BinEP	p-value	
			$H_1^{\neq}$	$H_1^{<}$
f_1	48.388353 ± 31.563956	98.263043 ± 8.390228	$\mathbf{1.77 \times 10^{-10}}$	$\mathbf{8.87 \times 10^{-11}}$
f_2	0.000031 ± 0.000037	0.000944 ± 0.000711	$\mathbf{1.91 \times 10^{-10}}$	$\mathbf{9.57 \times 10^{-11}}$
f_3	0.033333 ± 0.182574	0.633333 ± 0.490132	$\mathbf{1.065 \times 10^{-6}}$	$\mathbf{5.32 \times 10^{-7}}$
f_4	7.455879 ± 5.658761	33.256413 ± 6.630407	$\mathbf{1.61 \times 10^{-10}}$	$\mathbf{8.06 \times 10^{-11}}$
f_5	$105.622591 \pm 181.300265$	499.999999 ± 9.495927	$\mathbf{8.77 \times 10^{-10}}$	$\mathbf{4.38 \times 10^{-10}}$
f_6	0.000754 ± 0.002562	0.007920 ± 0.010586	**0.030794**	**0.015311**
f_7	59.656050 ± 29.375037	$1001.294175 \pm 82.935230$	$\mathbf{1.69 \times 10^{-17}}$	$\mathbf{8.45 \times 10^{-18}}$
f_8	20.077233 ± 0.004231	19.756093 ± 0.048469	$\mathbf{6.43 \times 10^{-12}}$	0.999999
f_9	$927.358756 \pm 997.768526$	$2814.936347 \pm 359.502199$	$\mathbf{5.57 \times 10^{-10}}$	$\mathbf{2.42 \times 10^{-13}}$
f_{10}	54.086319 ± 37.598022	322.117086 ± 18.492874	$\mathbf{2.99 \times 10^{-11}}$	$\mathbf{1.49 \times 10^{-11}}$
Avg Rank	1.1	1.9		

Figure 6.7 provides performance plots for binary-mapped functions where AMEP provided an improved performance. BinEP failed to improve the solutions for problems f_1, f_5, f_7, f_9, and f_{10} whereas AMEP was able to refine solutions further.

Diversity profiles for problems where BinEP failed to refine solutions are shown in Figure 6.8. These diversity profiles show that BinEP diversity did not change over the course of the experiment, indicating that BinEP converges too quickly onto a solution from an early iteration, without any refinement in the solution quality. The diversity for AMEP however, shows irregular jumps for some problems as iterations continue, particularly illustrated in Figure 6.8(c), which indicates that diversity was introduced into the population as iterations increased.

6.3.2 Binary-valued problems

Results for the minimization and maximization problems are respectively given in Table 6.5 and Table 6.6.



(a) f_1 (b) f_2 (c) f_3

(d) f_4 (e) f_5 (f) f_6

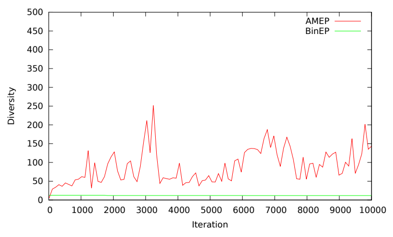
(g) f_7 (h) f_9 (i) f_{10}

Figure 6.7: AMEP and BinEP performance on binary-mapped continuous problems.

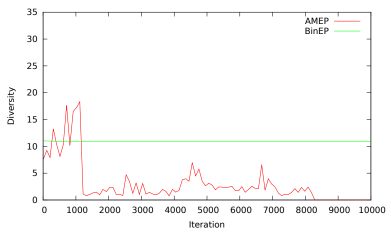
Minimization

As indicated by the p-values in Table 6.5, AMEP performed better than BinEP for eight of the ten binary minimization problems. For the other two problems, KC_4 and Q_4, both AMEP and BinEP provided the same accuracy. AMEP performed better than BinEP on the larger dimensional problems.

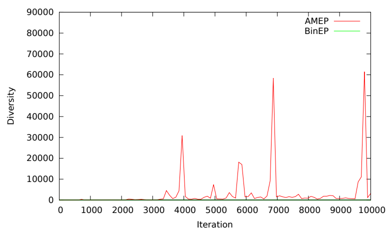
Figure 6.9 illustrates the performance of AMEP and BinEP for selected binary minimization problems and shows that AMEP was able to refine solutions better than what BinEP could. BinEP could not provide valid solutions to problems Q_7 and Q_8, returning a heavily penalized fitness value from an early iteration.



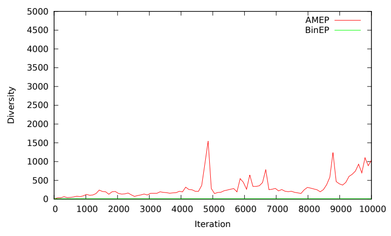
(a) f_1



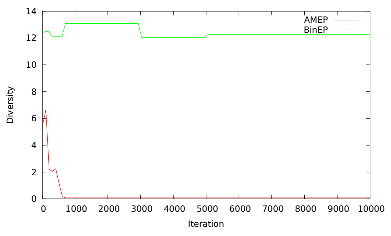
(b) f_4



(c) f_5



(d) f_7

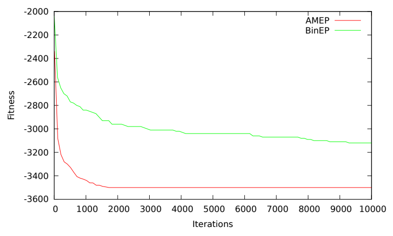


(e) f_{10}

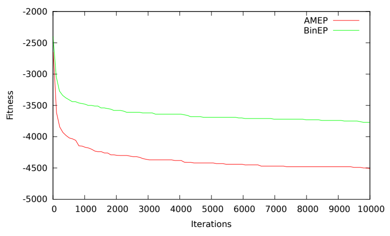
Figure 6.8: AMEP and BinEP diversity for selected binary-mapped problems

Table 6.5: Binary minimization function results for AMEP and BinEP, measured at a confidence of 95%. Bold p-values indicate statistical significance

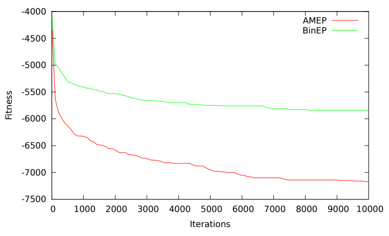
f	AMEP	BinEP	p-value	
			$H_1^{\neq}$	$H_1^{<}$
KC_4	-2000.0 ± 0.0	-2000.0 ± 0.0	1.0	1.0
KC_5	-3500.0 ± 0.0	-3120.0 ± 174.987684	$\mathbf{6.78 \times 10^{-12}}$	$\mathbf{3.39 \times 10^{-12}}$
KC_6	-4150.0 ± 229.467148	-3770.0 ± 187.818694	$\mathbf{2.47 \times 10^{-11}}$	$\mathbf{1.23 \times 10^{-11}}$
KC_7	-5550.0 ± 273.861278	-4720.0 ± 207.447542	$\mathbf{1.56 \times 10^{-11}}$	$\mathbf{7.82 \times 10^{-12}}$
KC_8	-7170.0 ± 490.003518	-5840.0 ± 241.546739	$\mathbf{1.75 \times 10^{-11}}$	$\mathbf{8.76 \times 10^{-12}}$
Q_4	0.0 ± 0.0	0.2 ± 0.610257	0.23895	0.118133
Q_5	0.0 ± 0.0	4.133333 ± 1.382983	$\mathbf{1.71 \times 10^{-12}}$	$\mathbf{8.56 \times 10^{-13}}$
Q_6	2.0666 ± 1.229895	13007.2 ± 15113.804346	$\mathbf{3.9 \times 10^{-12}}$	$\mathbf{1.95 \times 10^{-12}}$
Q_7	3.466666 ± 1.479359	30000.0 ± 0.0	$\mathbf{4.82 \times 10^{-13}}$	$\mathbf{2.41 \times 10^{-13}}$
Q_8	5.13333 ± 1.870521	30000.0 ± 0.0	$\mathbf{7.01 \times 10^{-13}}$	$\mathbf{3.5 \times 10^{-13}}$
Avg Rank	1.0	1.9		



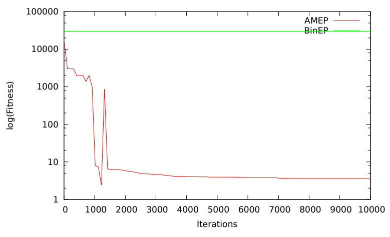
(a) KC_5



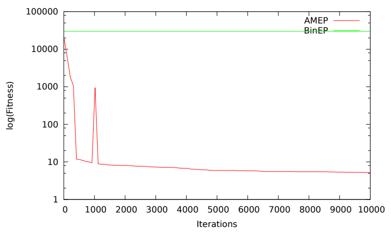
(b) KC_6



(c) KC_8



(d) Q_7



(e) Q_8

Figure 6.9: Binary minimization problem performances for AMEP and BinEP.

Maximization

As shown in Table 6.6, AMEP outperformed BinEP for six problems. For the remaining three problems, performance was the same for the two algorithms. The IPD results indicate that the AMEP tends to focus on greater fitnesses for both the best individual within the population, as well as improving the collective fitness of the population itself. This increase in fitness shows that the strategies cooperate more initially before always defecting. The standard deviations for the AMEP are, however, larger than those of BinEP for both IPD problems indicating an erratic behavior where learned strategies ranged from always defect to always cooperate. Overall, AMEP performs better than BinEP with an average rank of 1.0 compared to a rank of 1.9.

The maximization problem results are illustrated in Figure 6.10, for selected problems. The graphs show that AMEP performed better on larger dimensional problems, but that there is almost no difference between the algorithms.

Table 6.6: Binary maximization function results for AMEP and BinEP, measured at a confidence of 95%. Bold p-values indicate statistical significance

f	AMEP	BinEP	p-value	
			$H_1^{\neq}$	$H_1^{>}$
MDK_{10}	295.0 ± 0.0	295.0 ± 0.0	1.0	1.0
MDK_{20}	1024.0 ± 0.0	994.633333 ± 16.035485	$\mathbf{1.19 \times 10^{-12}}$	$\mathbf{5.93 \times 10^{-13}}$
KT_4	12.433333 ± 0.8172	11.6 ± 0.968468	$\mathbf{1.57 \times 10^{-4}}$	$\mathbf{7.89 \times 10^{-5}}$
KT_5	16.3 ± 0.915385	14.9 ± 0.803011	$\mathbf{2.74 \times 10^{-7}}$	$\mathbf{1.37 \times 10^{-7}}$
KT_6	32.73333 ± 0.520830	32.333333 ± 0.479463	**0.00695**	**0.0038**
KT_7	45.4 ± 0.932183	43.433333 ± 1.590561	$\mathbf{6.12 \times 10^{-7}}$	$\mathbf{3.06 \times 10^{-7}}$
KT_8	55.8 ± 6.315825	56.1 ± 1.213430	**0.0064**	**0.00319**
IPD_b	8533.333 ± 6494.928	7611.733 ± 373.912	0.6482	0.67612
IPD_c	$341333.333 \pm 259797.126$	304469.333 ± 14956.477	0.6502	0.67565
Avg Rank	1.0	1.9		

6.3.3 Overall outcome

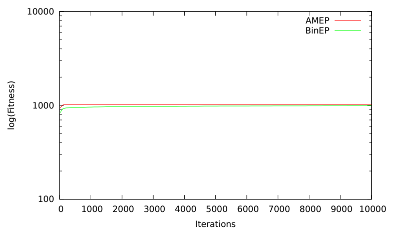
Over all problems, the performance and average rankings indicate that AMEP provided better performance, compared to BinEP. Therefore, based on the average rankings, where AMEP achieved the smaller ranking, AMEP performance was superior.

6.4 Differential evolution results

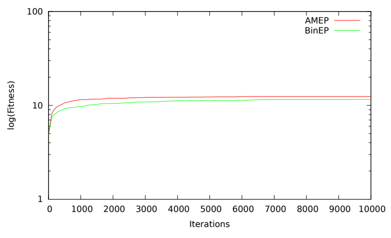
Chapter 3 proposed two binary versions of DE, namely BinDE and NormDE. This section performs two separate comparisons, one between the performance of AMDE and BinDE and the other between AMDE and NormDE. Results from the binary-mapped continuous-valued problems are presented in section 6.4.1, while section 6.4.2 covers the binary problem results.

6.4.1 Binary-mapped continuous-valued problems

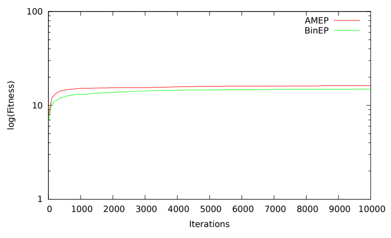
Table 6.7 presents the results of the DE algorithms on the binary-mapped continuous-valued problems. AMDE outperformed BinDE for eight of the problems, and for nine of the problems AMDE performed better than NormDE. For the other problems, the performances did not differ significantly as indicated by the p-values. For most of the



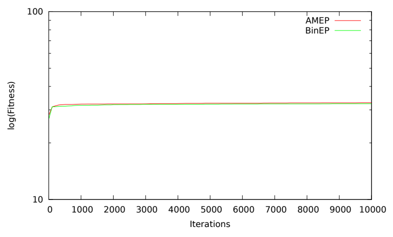
(a) MDK_{20}



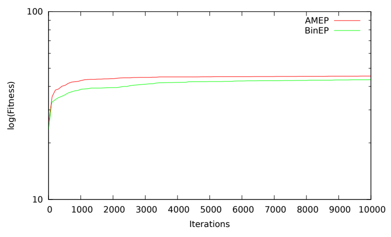
(b) KT_4



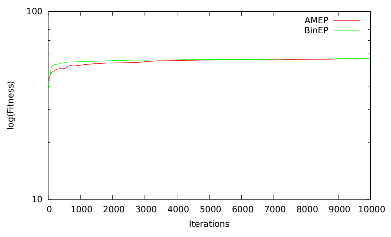
(c) KT_5



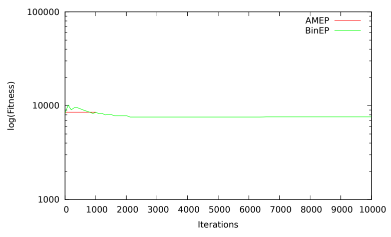
(d) KT_6



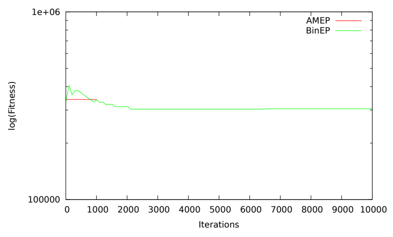
(e) KT_7



(f) KT_8



(g) IPD_b



(h) IPD_c

Figure 6.10: AMEP and BinEP performance for selected binary maximization problems.

problems where the AMDE provided a better accuracy, the mean fitness of AMDE showed a large difference to the competing binary algorithm. Such differences indicate that AMDE was able to achieve results that are more optimal than what was possible by the competing binary algorithms. The average ranks in table 6.7 also indicate that AMDE is superior. For both algorithm comparisons AMDE achieved an average rank of 1.1, whereas the average rank of BinDE and NormDE was 1.9.

Algorithm performances are illustrated in Figure 6.12 where it is noted that BinDE and NormDE were not able to refine solutions as well as AMDE, converging onto a solution too quickly. Problem f_8 was the only problem where AMDE performed worse. The population diversity for problems where BinDE and NormDE stopped refining solutions

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Table 6.7: Binary-mapped continuous-valued function results for AMDE compared to BinDE and NormDE. Measured at a confidence of 95%. Bold p-values indicate statistical significance.

f	AMDE	BinDE	p-value	
			$H_1^{\neq}$	$H_1^{<}$
f_1	0.00105 ± 0.001053	0.513073 ± 0.927609	$\mathbf{1.88 \times 10^{-11}}$	$\mathbf{9.40 \times 10^{-12}}$
f_2	0.000082 ± 0.000055	0.408522 ± 0.567259	$\mathbf{2.77 \times 10^{-10}}$	$\mathbf{1.38 \times 10^{-10}}$
f_3	0.0 ± 0.0	1.233333 ± 1.040004	$\mathbf{4.38 \times 10^{-9}}$	$\mathbf{2.19 \times 10^{-9}}$
f_4	$4.822\text{E-}6 \pm 0.000015$	0.048147 ± 0.088338	$\mathbf{1.07 \times 10^{-11}}$	$\mathbf{5.38 \times 10^{-12}}$
f_5	10.595878 ± 2.708785	32.335446 ± 89.376514	$\mathbf{9.11 \times 10^{-11}}$	$\mathbf{4.55 \times 10^{-11}}$
f_6	0.000001 ± 0.000003	0.013715 ± 0.034670	0.12497	0.063231
f_7	0.209604 ± 0.387201	3.017116 ± 4.947680	$\mathbf{2.75 \times 10^{-9}}$	$\mathbf{1.37 \times 10^{-9}}$
f_8	10.158624 ± 10.095957	4.173890 ± 3.188617	0.9074	0.547962
f_9	0.396522 ± 1.333000	97.108307 ± 45.402377	$\mathbf{1.55 \times 10^{-11}}$	$\mathbf{7.77 \times 10^{-12}}$
f_{10}	6.064350 ± 13.117947	34.008954 ± 8.009381	$\mathbf{9.60 \times 10^{-8}}$	$\mathbf{4.80 \times 10^{-8}}$
Avg Rank	1.1	1.9		

f	AMDE	NormDE	p-value	
			$H_1^{\neq}$	$H_1^{<}$
f_1	0.00105 ± 0.001053	0.378993 ± 0.001053	$\mathbf{2.83 \times 10^{-11}}$	$\mathbf{1.41 \times 10^{-11}}$
f_2	0.000082 ± 0.000055	0.382120 ± 0.650682	$\mathbf{3.57 \times 10^{-11}}$	$\mathbf{1.79 \times 10^{-11}}$
f_3	0.0 ± 0.0	1.133333 ± 0.937102	$\mathbf{1.32 \times 10^{-8}}$	$\mathbf{6.62 \times 10^{-9}}$
f_4	$4.822\text{E-}6 \pm 0.000015$	0.023030 ± 0.0550097	$\mathbf{1.33 \times 10^{-11}}$	$\mathbf{6.65 \times 10^{-12}}$
f_5	10.595878 ± 2.708785	14.269629 ± 2.284204	$\mathbf{2.52 \times 10^{-11}}$	$\mathbf{1.26 \times 10^{-11}}$
f_6	0.000001 ± 0.000003	0.022000 ± 0.029606	$\mathbf{5.28 \times 10^{-12}}$	$\mathbf{2.54 \times 10^{-12}}$
f_7	0.209604 ± 0.387201	1.546037 ± 1.382063	$\mathbf{2.28 \times 10^{-8}}$	$\mathbf{1.14 \times 10^{-8}}$
f_8	10.158624 ± 10.095957	3.868201 ± 2.070014	0.9083	0.549172
f_9	0.396522 ± 1.333000	109.868756 ± 69.280991	$\mathbf{1.55 \times 10^{-11}}$	$\mathbf{7.78 \times 10^{-12}}$
f_{10}	6.064350 ± 13.117947	32.940752 ± 7.649142	$\mathbf{2.87 \times 10^{-8}}$	$\mathbf{1.43 \times 10^{-8}}$
Avg Rank	1.1	1.9		

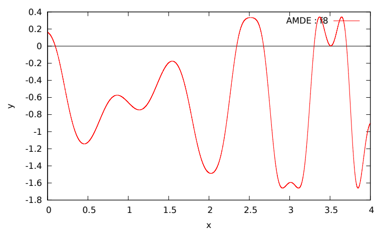
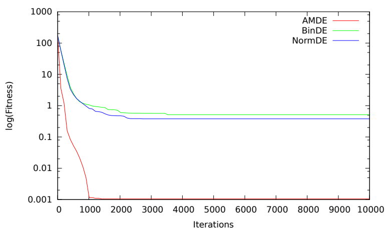


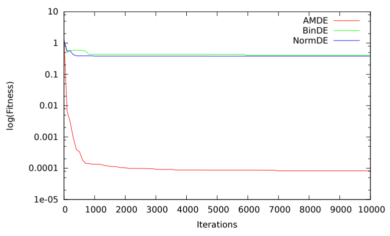
Figure 6.11: AMDE bit string generating function for f_8

is given in Figure 6.13. The plots show that the population diversity of both BinDE and NormDE remains unchanged, for f_2, from an early iteration, with a decreasing population diversity for problems f_6 and f_8. A solution was found for f_6, but f_8 indicates that the population diversity of BinDE and NormDE decreases, and then stagnates indicating that exploitation of the solution was not possible.

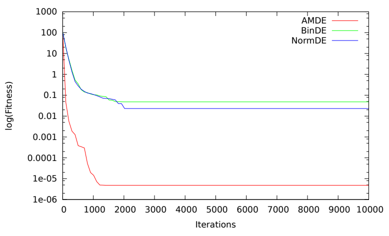
Figure 6.11 illustrates an evolved bit string generating function for problem f_8. The target solution for this problem is given by $\mathbf{x} = \mathbf{0}$, but several of the generated bit values are 1, specifically the MSB. The MSB has a large effect on the value of the evaluated fitness, therefore resulting in an sub-optimal bit string solution produced by the generating function.



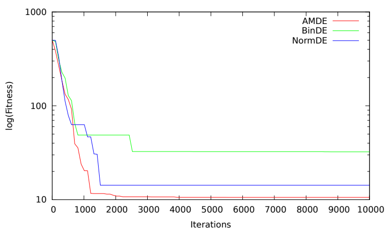
(a) f_1



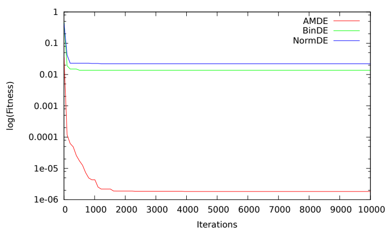
(b) f_2



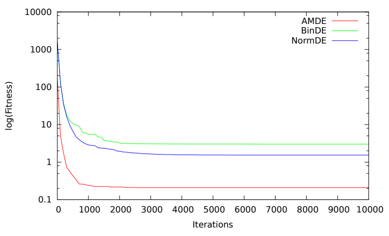
(c) f_4



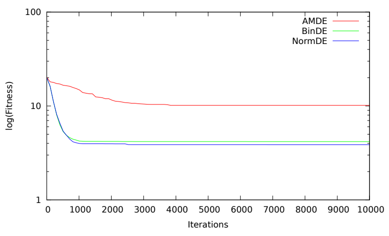
(d) f_5



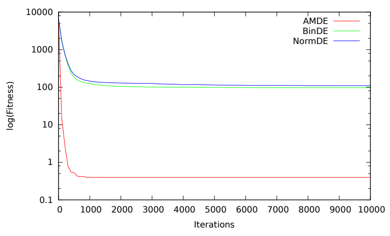
(e) f_6



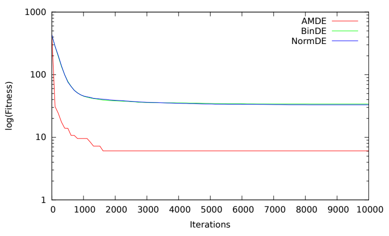
(f) f_7



(g) f_8

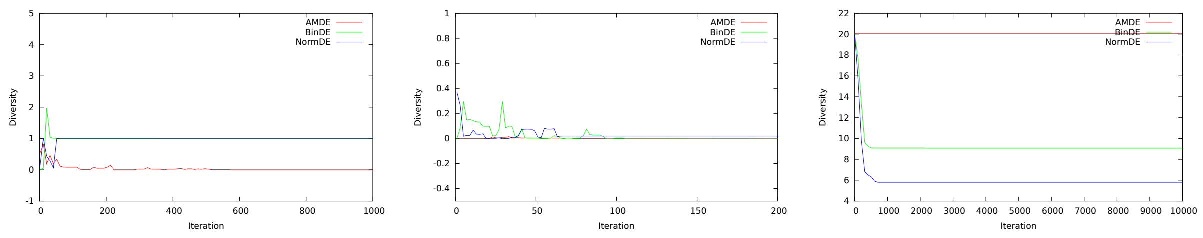


(h) f_9



(i) f_{10}

Figure 6.12: Performance plots of AMDE, BinDE and NormDE.



(a) f_2 (b) f_6 (c) f_8

Figure 6.13: Diversity of AMDE, BinDE and NormDE for selected binary-mapped problems

6.4.2 Binary-valued problems

Table 6.8 presents the results for the minimization problems, while maximization results are given in Table 6.9.

Minimization

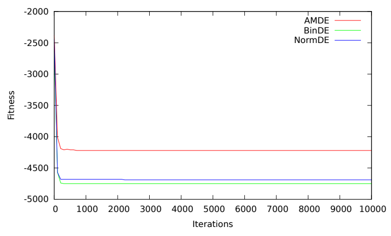
From the p-values presented in Table 6.8, AMDE provides a better performance than BinDE in six of the problems and in five of the problems compared to NormDE. When AMDE did outperform the binary algorithms, the difference between mean fitnesses was large, as illustrated in Figure 6.14. BinDE and NormDE provided no valid solutions for problems Q_7 and Q_8, and the remainder of the n-queens problems had AMDE provide solutions that were better than that of the binary algorithm variants, even though the solutions were achieved within 1000 iterations. The KC problems show that AMDE was not able to out-perform the binary algorithms. All algorithms achieved the target solution for problem KC_5, and all, except for BinDE, reached the target solution for KC_4.

The average ranks also indicate that AMDE performances were better with respective average ranks of 1.3 and 1.6 between AMDE and BinDE, and 1.4 and 1.6 between AMDE and NormDE.

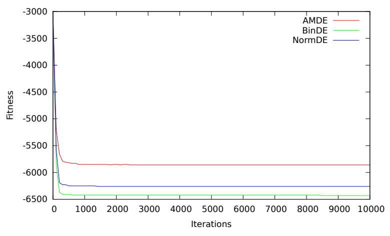
Maximization

Results for the binary maximization binary problems together with p-values, show that AMDE did not manage to achieve better accuracy in any of the nine problems. The

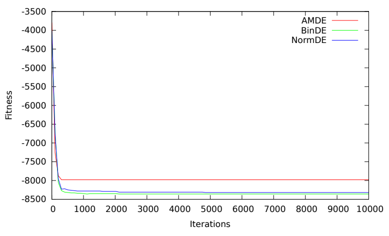
UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA



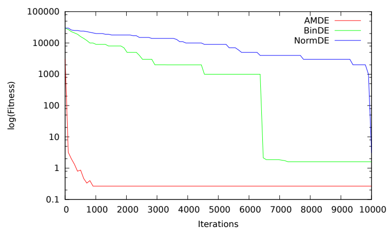
(a) KC_6



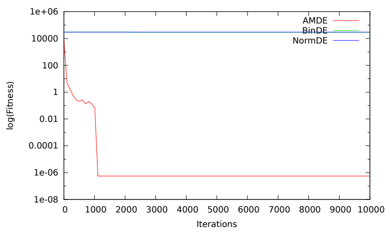
(b) KC_7



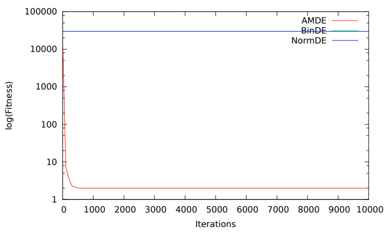
(c) KC_8



(d) Q_6



(e) Q_7



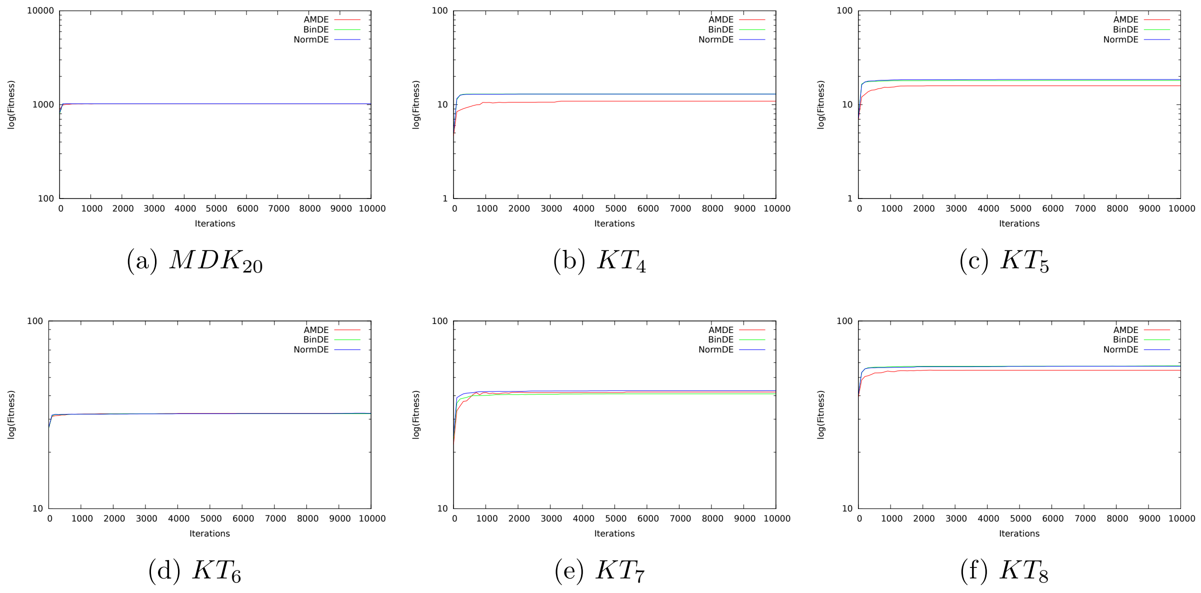
(f) Q_8

Figure 6.14: Binary minimization performances between AMDE, BinDE and NormDE.

binary algorithm variants provided better performance than AMDE, except for problems KT_6 and KT_7 between AMDE and BinDE, where AMDE provided a better mean fitness. Average ranks show that BinDE with an average rank of 1.2 performs better than AMDE which has an average rank of 1.7. Similarly, the average ranks are in favor of NormDE with a rank of 1.0 where AMDE has an average rank of 1.9.

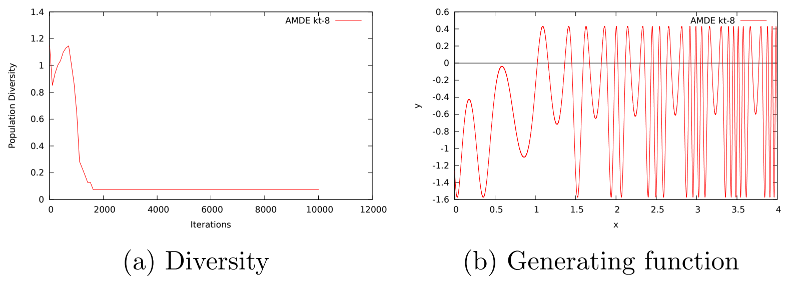
Figure 6.15 illustrates performance for the binary maximization problems. AMDE failed to outperform the binary algorithms. Figure 6.16(a) gives the diversity of the population for problem KT_8 and from this it can be seen that the population does converge onto a solution. The evolved bit string generating function, as illustrated in Figure 6.16(b), is not able to produce a bit string which is better than those generated by the binary algorithms. The diversity indicates that no further exploration of the search space occurs after approximately 2000 iterations. It is probable that the repair strategy within the KT problem prevents valid solutions from being evolved by AMDE for the KT problems, as several 1 bits will be produced from the generating function.

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA



(a) MDK_{20} (b) KT_4 (c) KT_5 (d) KT_6 (e) KT_7 (f) KT_8

Figure 6.15: AMDE, BinDE and NormDE performance for selected binary maximization problems.



(a) Diversity (b) Generating function

Figure 6.16: Population diversity and generating function of AMDE for problem KT_8 in continuous-valued four-tuple space.

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Table 6.8: Binary minimization function results for AMDE compared to BinDE and NormDE. Measured confidence of 95%. Bold p-values indicate statistical significance.

f	AMDE	BinDE	p-value	
			$H_1^{\neq}$	$H_1^{<}$
KC_4	-2000.0 ± 0.0	-1990.0 ± 54.772255	0.33371	0.498977
KC_5	-3500.0 ± 0.0	-3500.0 ± 0.0	1.0	1.0
KC_6	-4220.0 ± 76.112439	-4750.0 ± 138.339911	$\mathbf{1.31 \times 10^{-12}}$	0.999999
KC_7	-5860.0 ± 171.403939	-6430.0 ± 232.156308	$\mathbf{2.01 \times 10^{-10}}$	0.99999999
KC_8	-7980.0 ± 76.112439	-8360.0 ± 328.633534	$\mathbf{1.62 \times 10^{-7}}$	0.999999
Q_4	0.0 ± 0.0	0.133333 ± 0.507416	0.491296	0.247287
Q_5	0.0 ± 0.0	0.4 ± 0.813676	**0.0234**	**0.011627**
Q_6	0.266666 ± 0.691491	1.6 ± 0.813676	$\mathbf{2.99 \times 10^{-7}}$	$\mathbf{1.49 \times 10^{-7}}$
Q_7	0.0 ± 0.0	29000.0 ± 5477.225575	$\mathbf{1.16 \times 10^{-13}}$	$\mathbf{5.82 \times 10^{-14}}$
Q_8	2.0 ± 0.0	30000.0 ± 0.0	$\mathbf{1.68 \times 10^{-14}}$	$\mathbf{8.42 \times 10^{-15}}$
Avg Rank	1.3	1.6		

f	AMDE	NormDE	p-value	
			$H_1^{\neq}$	$H_1^{<}$
KC_4	-2000.0 ± 0.0	-2000.0 ± 0.0	1.0	1.0
KC_5	-3500.0 ± 0.0	-3500.0 ± 0.0	1.0	1.0
KC_6	-4220.0 ± 76.112439	-4690.0 ± 229.467148	$\mathbf{3.62 \times 10^{-10}}$	0.999999
KC_7	-5860.0 ± 171.403939	-6260.0 ± 277.426597	$\mathbf{1.02 \times 10^{-7}}$	0.999999
KC_8	-7980.0 ± 76.112439	-8320.0 ± 283.329952	$\mathbf{8.09 \times 10^{-8}}$	0.999999
Q_4	0.0 ± 0.0	0.066666 ± 0.365148	0.33371	0.166855
Q_5	0.0 ± 0.0	0.466666 ± 0.860366	**0.010657**	**0.00533**
Q_6	0.266666 ± 0.691491	3.133333 ± 3.510902	$\mathbf{1.00 \times 10^{-8}}$	$\mathbf{9.96 \times 10^{-9}}$
Q_7	0.0 ± 0.0	30000.0 ± 0.0	$\mathbf{1.68 \times 10^{-14}}$	$\mathbf{8.42 \times 10^{-15}}$
Q_8	2.0 ± 0.0	30000.0 ± 0.0	$\mathbf{1.68 \times 10^{-14}}$	$\mathbf{8.42 \times 10^{-15}}$
Avg Rank	1.4	1.6		

6.4.3 Overall outcome

From the average ranks, AMDE provided better accuracy for both the binary-mapped continuous-valued problems and the binary minimization problems. The binary maximization problems however, favored the BinDE and NormDE but mean fitnesses between AMDE and the non-AM algorithms are similar. Based on the average rankings, the AMDE performed better than its binary variants.

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Table 6.9: Binary maximization function results for AMDE compared to BinDE and NormDE. Measured at a confidence of 95%. Bold p-values indicate statistical significance.

f	AMDE	BinDE	p-value	
			$H_1^{\neq}$	$H_1^{>}$
MDK_{10}	294.966666 ± 0.182574	295.0 ± 0.0	0.33371	0.833144
MDK_{20}	1020.866666 ± 6.146113	1022.733333 ± 3.571277	0.14023	0.933481
KT_4	10.9 ± 1.863441	12.966666 ± 0.182574	$\mathbf{4.73 \times 10^{-8}}$	0.999999
KT_5	15.9 ± 1.268993	18.1 ± 1.061878	$\mathbf{3.73 \times 10^{-8}}$	0.999999
KT_6	32.166666 ± 0.461133	32.066666 ± 0.639683	0.33852	0.169597
KT_7	41.8 ± 4.77349	40.9 ± 3.294195	0.324329	0.163754
KT_8	54.633333 ± 1.751518	57.666666 ± 1.516196	$\mathbf{3.76 \times 10^{-8}}$	0.999999
IPD_b	8531.266 ± 6489.151	8654.133 ± 856.282	0.6615	0.67194
IPD_c	$341210.133 \pm 259555.558$	346127.266 ± 34259.65	0.66129	0.67158
Avg Rank	1.7	1.2		

f	AMDE	NormDE	p-value	
			$H_1^{\neq}$	$H_1^{>}$
MDK_{10}	294.944444 ± 0.182574	294.9 ± 0.402577	0.745066	0.372330
MDK_{20}	1020.866666 ± 6.146113	1022.1 ± 4.69298	0.293956	0.861960
KT_4	10.9 ± 1.863441	12.966666 ± 0.182574	$\mathbf{4.73 \times 10^{-8}}$	0.999999
KT_5	15.9 ± 1.268993	18.533333 ± 1.252124	$\mathbf{8.32 \times 10^{-9}}$	0.999999
KT_6	32.166666 ± 0.461133	32.2 ± 0.846901	0.811513	0.60065
KT_7	41.8 ± 4.773490	42.566666 ± 3.058998	0.94802	0.529857
KT_8	54.633333 ± 1.751518	57.4 ± 1.499425	$\mathbf{1.15 \times 10^{-7}}$	0.999999
IPD_b	8531.266 ± 6489.151	8803.4 ± 640.095	0.659825	0.672878
IPD_c	$341210.133 \pm 259555.558$	352096.466 ± 25602.444	0.6607	0.67219
Avg Rank	1.9	1.0		

6.5 Particle swarm optimization results

This section compares the performance accuracy of AMPSO and BinPSO. Section 6.5.1 discusses the set of binary-mapped continuous-valued benchmark problems, with results for the set of binary-valued problems provided in Section 6.5.2.

6.5.1 Binary-mapped continuous-valued problems

Table 6.10 provides the results of AMPSO and BinPSO on the binary-mapped continuous-valued problems. AMPSO provided better performance for nine of the problems, and displayed a small deviation in the results. Mean accuracy differences between the two algorithms indicate that BinPSO is at least 97.3% worse for several of the problems.

Table 6.10: Binary-mapped continuous-valued minimization function results for AMPSO and BinPSO, measured at a confidence of 95%. Bold p-values indicate statistical significance.

f	AMPSO	BinPSO	p-value	
			$H_1^{\neq}$	$H_1^{<}$
f_1	0.108380 ± 0.272725	99.142794 ± 7.171802	$\mathbf{1.92 \times 10^{-11}}$	$\mathbf{9.61 \times 10^{-12}}$
f_2	0.000169 ± 0.000499	0.000001 ± 0.000002	$\mathbf{1.39 \times 10^{-9}}$	0.999999
f_3	0.0 ± 0.0	0.166660 ± 0.379049	**0.021419**	**0.026096**
f_4	0.002691 ± 0.006866	33.887258 ± 5.490071	$\mathbf{2.38 \times 10^{-11}}$	$\mathbf{1.19 \times 10^{-11}}$
f_5	7.904640 ± 4.781322	$499.999999 \pm 0.00000004$	$\mathbf{2.88 \times 10^{-11}}$	$\mathbf{1.44 \times 10^{-11}}$
f_6	0.000411 ± 0.001760	0.002739 ± 0.004619	0.204482	0.898292
f_7	0.124504 ± 0.308436	$1338.154191 \pm 77.497687$	$\mathbf{1.77 \times 10^{-11}}$	$\mathbf{8.88 \times 10^{-12}}$
f_8	14.001633 ± 9.288794	19.879088 ± 0.029248	**0.02603**	0.9874
f_9	7.794557 ± 11.760161	$2434.387761 \pm 333.687736$	$\mathbf{2.8 \times 10^{-11}}$	$\mathbf{1.40 \times 10^{-11}}$
f_{10}	8.305339 ± 14.779981	318.328762 ± 14.476842	$\mathbf{1.43 \times 10^{-11}}$	$\mathbf{7.18 \times 10^{-12}}$
Avg Rank	1.1	1.9		

The average ranks in Table 6.10 show that the accuracy of AMPSO, having a rank of 1.1, is better than that of BinPSO, with a rank of 1.9. Figure 6.18 provides the performance plots for AMPSO and BinPSO for the binary-mapped continuous-valued problems. From the nine problems where AMPSO performed better than BinPSO, the initial solution was refined by AMPSO whereas the solution found by BinPSO did not deviate much from the initial found solution in seven of the problems. The diversity of BinPSO, illustrated in Figure 6.19 for selected problems, shows that little exploitation occurs and that swarm is predominately exploring the search space. Over time, the diversity of AMPSO decreases indicating exploitation and solution refinement, whereas diversity of BinPSO initially increases slightly and then oscillates around a diversity value. This results in very little refinement of the solution from iteration to iteration. It is probable that Hamming cliffs are the reason that solutions are not refined.

For problem f_2, where AMPSO provided worse results, the bit string generating function failed to produce a better bit string than the BinPSO. Figure 6.17 shows this bit string generating function for AMPSO. The generating function produces bit values for the f_2 problem and results in poor solutions, even though f_2 has a low dimensionality. The target value for problem f_2 is given as $\mathbf{x} = \mathbf{0}$, but the generating function will produce 1 bits and therefore will result in a poor solution to the problem. The generating

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

function should produce 0 bits, indicating that PSO did not produce good values for the AM four-tuple. The target bit string should be:

000000000000000000000000

but the generated bit string is:

100000000001100000110010

which has a Hamming distance of six from the target bit string.

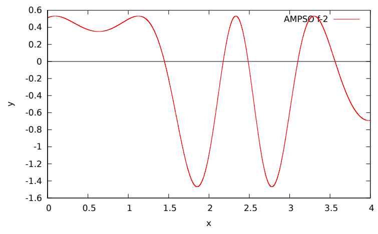


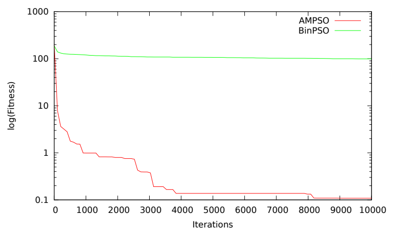
Figure 6.17: AMPSO generating function for binary-mapped problem f_2

6.5.2 Binary-valued problems

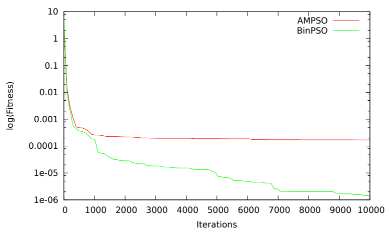
Results for minimization and maximization are given respectively in Table 6.11 and Table 6.12.

Minimization

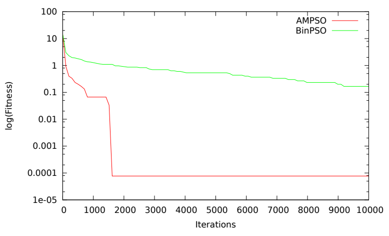
From the p-values in Table 6.11, AMPSO provided better performance for four of the problems, with another four problems achieving the same performance for AMPSO and BinPSO. For the four problems with the same performance, the problem dimension is small indicating that both algorithms did not struggle to find the solution. Compared to BinPSO, AMPSO provided a larger deviation for some problems indicating that different solutions were found in the problem space. Despite the larger deviations of AMPSO, for some problems the fitness obtained is still better than that of BinPSO. Overall the average rank is better for AMPSO with a value of 1.4, where BinPSO achieved a rank of 1.6.



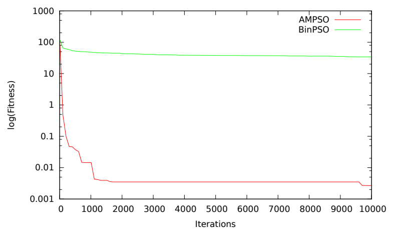
(a) f_1



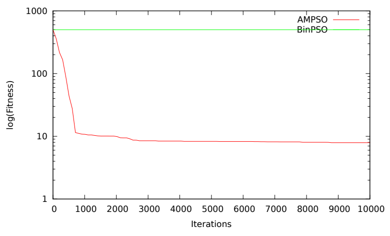
(b) f_2



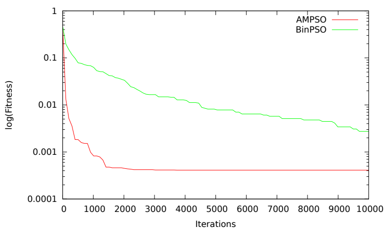
(c) f_3



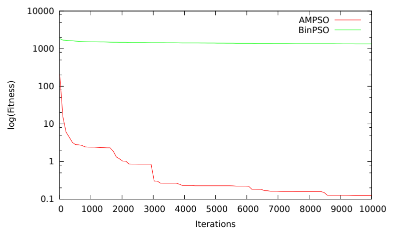
(d) f_4



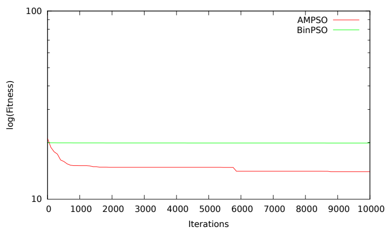
(e) f_5



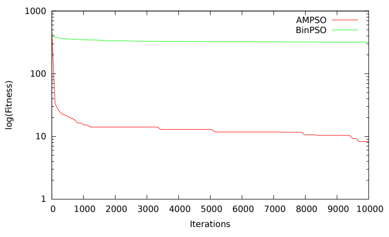
(f) f_6



(g) f_7

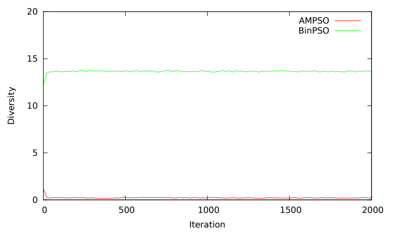


(h) f_8

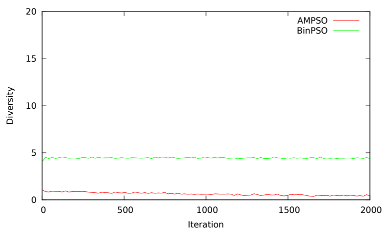


(i) f_{10}

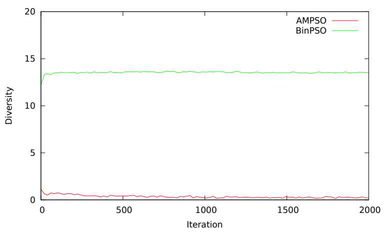
Figure 6.18: Performance plots of AMPSO and BinPSO for binary-mapped problems.



(a) f_1



(b) f_5



(c) f_7

Figure 6.19: AMPSO and BinPSO diversity for selected binary-mapped problems

Table 6.11: Binary minimization function results for AMPSO and BinPSO, measured at a confidence of 95%. Bold p-values indicate statistical significance.

f	AMPSO	BinPSO	p-value	
			$H_1^{\neq}$	$H_1^{<}$
KC_4	-2000.0 ± 0.0	-2000.0 ± 0.0	1.0	1.0
KC_5	-3500.0 ± 0.0	-3500.0 ± 0.0	1.0	1.0
KC_6	-4410.0 ± 263.072036	-4790.0 ± 54.772255	$\mathbf{2.95 \times 10^{-8}}$	0.999999
KC_7	-5970.0 ± 218.379612	-5980.0 ± 174.987684	0.568416	0.741656
KC_8	-8000.0 ± 0.0	-7290.0 ± 242.615053	$\mathbf{7.38 \times 10^{-13}}$	$\mathbf{3.69 \times 10^{-13}}$
Q_4	0.0 ± 0.0	0.0 ± 0.0	1.0	1.0
Q_5	0.0 ± 0.0	0.0 ± 0.0	1.0	1.0
Q_6	0.333333 ± 0.758098	2001.866666	$\mathbf{6.76 \times 10^{-8}}$	$\mathbf{3.38 \times 10^{-8}}$
Q_7	0.066666 ± 0.365148	30000.0 ± 0.0	$\mathbf{2.7 \times 10^{-14}}$	$\mathbf{1.35 \times 10^{-14}}$
Q_8	2.0 ± 0.0	30000.0 ± 0.0	$\mathbf{1.68 \times 10^{-14}}$	$\mathbf{8.42 \times 10^{-15}}$
Avg Rank	1.4	1.6		

Figure 6.21 illustrates the performance of AMPSO and BinPSO. These plots show that AMPSO achieved a solution faster than the BinPSO. Despite the faster convergence of the AMPSO, BinPSO was still able to refine its solutions as the algorithm iterations continued. Solution refinement was however, not possible in Q_7 and Q_8 where BinPSO provided invalid solutions. BinPSO outperformed AMPSO for problem KC_6, producing a better average fitness and lower standard deviation. Even though both AMPSO and BinPSO continued to improve the solution for KC_6 over the iterations, AMPSO could not achieve the results of BinPSO.

Figure 6.20 gives the bit string generating function which failed to produce a bit string to improve on BinPSO. This generating function has a small period, and is vertically shifted in the positive direction. Such a generating function will mostly generate 1 bit values, which is not what is required to solve the binary minimization problems.

Maximization

Table 6.12 provides results for the maximization problems. The results indicate that AMPSO managed to achieve a better performance in only two of the problems, with the remaining problems indicating almost no difference in algorithm performance. This similarity is confirmed by the p-values for most of the problems, except for the KT problems where BinPSO performed better than AMPSO.

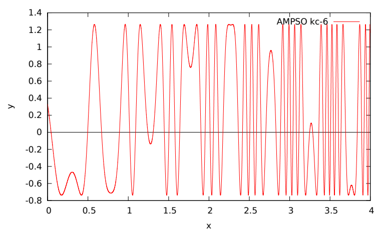


Figure 6.20: AMPSO bit string generating function for KC_6

Figure 6.22 illustrates the performances of AMPSO and BinPSO on selected binary maximization problems. For the two IPD problems, where AMPSO performed better than BinPSO, the generated strategies initially provided cooperation behavior, after which defection was predominant. For the problems where AMPSO provided poor performance, the AM four-tuples did not produce bit string generating functions that could match the performance of BinPSO. The ranks in Table 6.12 show that BinPSO performed better with an average rank of 1.44, while AMPSO achieved an average rank of 1.55.

6.5.3 Overall outcome

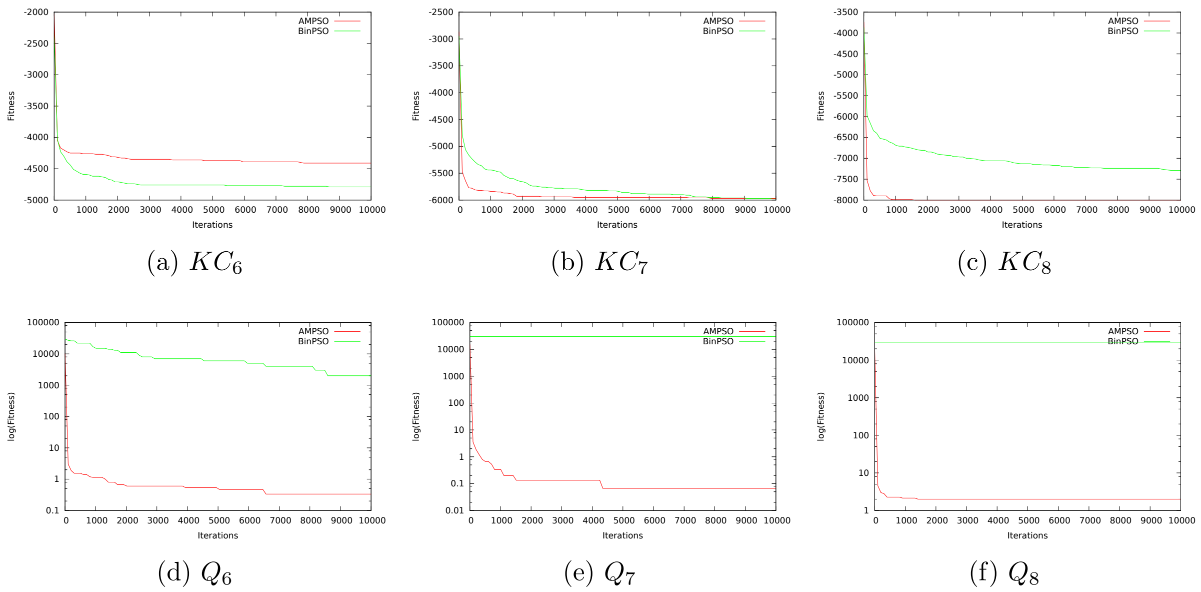
The benchmark results and rankings for AMPSO and BinPSO indicate that AMPSO provided, in general, better performance than BinPSO overall.

6.6 Artificial bee colony results

As with the DE, Chapter 3 proposed two binary versions of ABC, namely BinABC and NormABC. This section compares the performance of AMABC to BinABC and to NormABC. Results for the binary-mapped continuous-valued problems are presented in section 6.6.1, with the results on the binary problems provided in section 6.6.2.

6.6.1 Binary-mapped continuous-valued problems

Table 6.13 provides the results of the ABC algorithms on the binary-mapped continuous-valued problems. For five of the problems AMABC outperformed BinABC, and for the



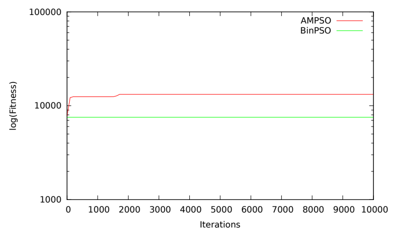
(a) KC_6 (b) KC_7 (c) KC_8

(d) Q_6 (e) Q_7 (f) Q_8

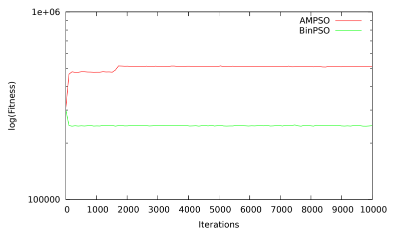
Figure 6.21: Binary minimization performances between AMPSO and BinPSO.

Table 6.12: Binary maximization function results for AMPSO and BinPSO, measured at a confidence of 95%. Bold p-values indicate statistical significance.

f	AMPSO	BinPSO	p-value	
			$H_1^{\neq}$	$H_1^{>}$
MDK_{10}	295.0 ± 0.0	295.0 ± 0.0	1.0	1.0
MDK_{20}	1024.0 ± 0.0	1024.0 ± 0.0	1.0	1.0
KT_4	12.833333 ± 0.379049	13.0 ± 0.0	0.0538	0.9892904
KT_5	17.4 ± 1.476248	18.733333 ± 1.080655	$\mathbf{1.90 \times 10^{-5}}$	0.999990
KT_6	32.866666 ± 0.571346	33.3 ± 0.794376	**0.011022**	0.994818
KT_7	45.866666 ± 0.507416	45.8 ± 0.610257	0.654333	0.500526
KT_8	57.466666 ± 0.860366	59.233333 ± 0.678910	$\mathbf{4.11 \times 10^{-9}}$	0.999999
IPD_b	13237.666 ± 3580.801	7550.033 ± 103.378	$\mathbf{6.45 \times 10^{-5}}$	$\mathbf{3.22 \times 10^{-5}}$
IPD_c	$510894.666 \pm 172595.778$	247814.1 ± 7001.326	$\mathbf{3.54 \times 10^{-5}}$	$\mathbf{1.77 \times 10^{-5}}$
Avg Rank	1.55	1.44		



(a) IPD_b



(b) IPD_c

Figure 6.22: AMPSO and BinPSO performance for selected binary maximization problems.

same five problems, AMABC outperformed NormABC as confirmed by the calculated p-values. The average ranks indicate that AMABC with an rank of 1.6 performed worse than both BinABC and NormABC which have a ranking of 1.4.

Figure 6.23 illustrates the algorithm performances and indicates that AMABC failed to refine solutions for five of the problems. These problems are all high dimensional problems, except for f_5 which has a dimension of two.

The diversity profiles of AMABC, as illustrated in Figure 6.24, show that the ant colony does not converge onto a single solution for some problems, and that the ant colony for f_5 and f_6 starts to diverge around iteration 400 and 200 respectively. Problem f_8 also resulted in divergent behavior but much earlier, except that although the diversity of BinABC and NormABC increased, it also dropped later eventually converging onto a solution. This divergent behavior also explains the large deviations in Table 6.13, as the solutions are found early, and that AMABC was outperformed by the binary algorithm variants.

Figure 6.25 provides the resultant generating functions for selected problems for which divergent behavior was obtained. The generating function for f_5 mainly creates 1 bit values, but was found at an early iteration before the ant colony diverged, therefore the generating function would not produce valid bit strings. Similarly, the generating functions for f_6 and f_8 were not refined by AMABC due to the divergent colony behavior.

6.6.2 Binary-valued problems

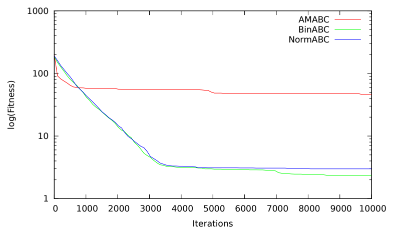
Table 6.14 presents the results for the binary minimization problems, with the binary maximization results provided in Table 6.15.

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

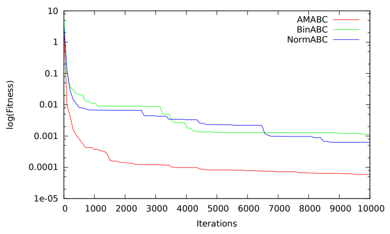
Table 6.13: Binary-mapped continuous-valued minimization function results for AMABC compared to BinABC and NormABC, measured at a confidence of 95%. Bold p-values indicate statistical significance.

f	AMABC	BinABC	p-value	
			$H_1^{\neq}$	$H_1^{<}$
f_1	46.019016 ± 27.032470	2.349176 ± 1.564701	$\mathbf{1.12 \times 10^{-12}}$	0.999999
f_2	0.000005 ± 0.000003	0.001125 ± 0.002240	$\mathbf{3.73 \times 10^{-9}}$	$\mathbf{1.86 \times 10^{-9}}$
f_3	0.0 ± 0.0	0.133333 ± 0.345745	**0.041773**	**0.020886**
f_4	2.859712 ± 3.751347	0.018678 ± 0.020928	$\mathbf{7.75 \times 10^{-9}}$	0.999999
f_5	$254.987408 \pm 248.199241$	11.907222 ± 1.032585	**0.00198**	0.999025
f_6	0.0 ± 0.0	0.008217 ± 0.013048	$\mathbf{5.66 \times 10^{-7}}$	$\mathbf{2.83 \times 10^{-7}}$
f_7	61.407034 ± 18.764513	6.054737 ± 6.276384	$\mathbf{1.13 \times 10^{-15}}$	0.999999
f_8	20.078535 ± 0.0	4.440892E-16 $\pm$ 0.0	$\mathbf{1.68 \times 10^{-14}}$	0.999999
f_9	$255.463737 \pm 227.799763$	231.130574 ± 55.833095	**0.02382**	**0.012189**
f_{10}	45.459317 ± 21.304660	80.660512 ± 10.029137	$\mathbf{3.86 \times 10^{-8}}$	$\mathbf{1.93 \times 10^{-8}}$
Avg Rank	1.6	1.4		

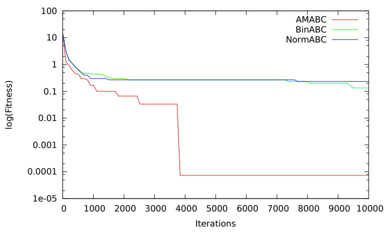
f	AMABC	NormABC	p-value	
			$H_1^{\neq}$	$H_1^{<}$
f_1	46.019016 ± 27.032470	2.987273 ± 1.458625	$\mathbf{3.63 \times 10^{-12}}$	0.999999
f_2	0.000005 ± 0.000003	0.000063 ± 0.001180	$\mathbf{4.42 \times 10^{-6}}$	$\mathbf{2.21 \times 10^{-6}}$
f_3	0.0 ± 0.0	0.233333 ± 0.430183	**0.010533**	**0.005563**
f_4	2.859712 ± 3.751347	0.021546 ± 0.024542	$\mathbf{1.84 \times 10^{-8}}$	0.999999
f_5	$254.987408 \pm 248.199241$	44.459309 ± 123.833984	**0.012025**	0.994625
f_6	0.0 ± 0.0	0.006505 ± 0.005034	$\mathbf{1.79 \times 10^{-7}}$	$\mathbf{8.95 \times 10^{-8}}$
f_7	61.407034 ± 18.764513	3.797366 ± 3.146056	$\mathbf{1.69 \times 10^{-17}}$	0.999999
f_8	20.078535 ± 0.0	4.440892E-16 $\pm$ 0.0	$\mathbf{1.68 \times 10^{-14}}$	0.999999
f_9	$255.463737 \pm 227.799763$	226.475861 ± 62.977697	0.069717	**0.034688**
f_{10}	45.459317 ± 21.304660	78.210589 ± 11.502809	$\mathbf{8.83 \times 10^{-8}}$	$\mathbf{4.41 \times 10^{-8}}$
Avg Rank	1.6	1.4		



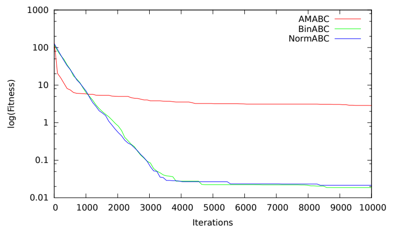
(a) f_1



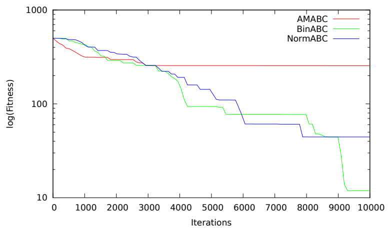
(b) f_2



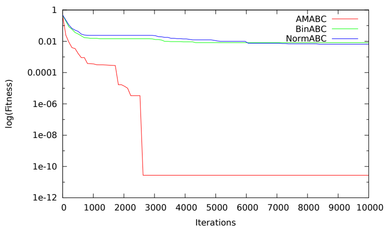
(c) f_3



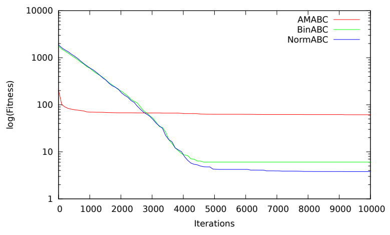
(d) f_4



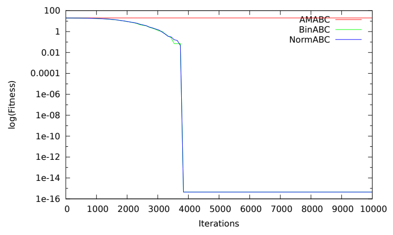
(e) f_5



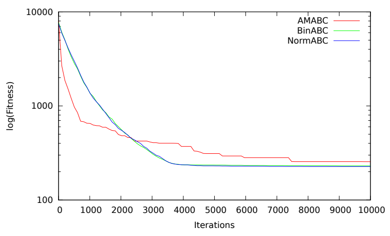
(f) f_6



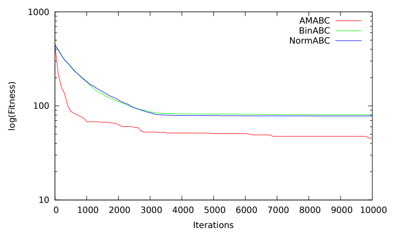
(g) f_7



(h) f_8

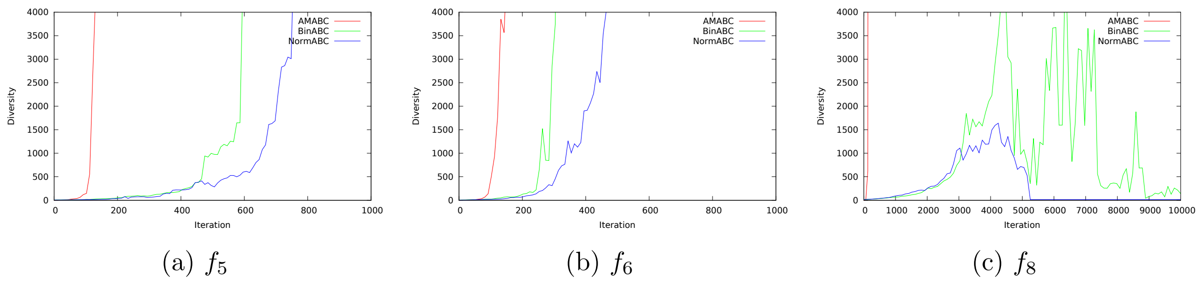


(i) f_9



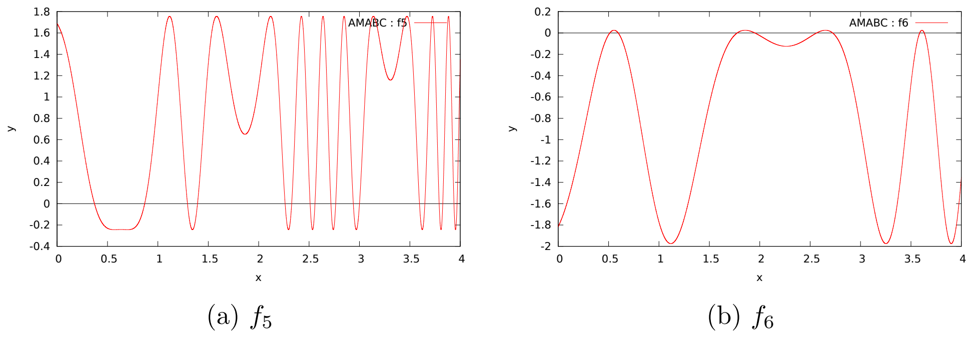
(j) f_{10}

Figure 6.23: Performances of AMABC, BinABC and NormABC on binary-mapped continuous problems

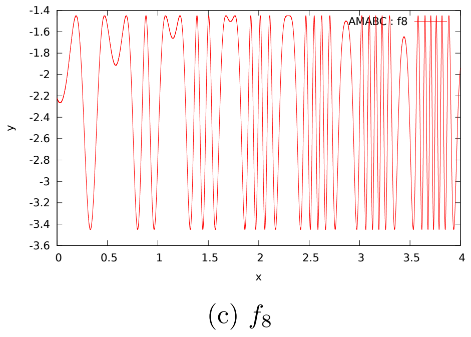


(a) f_5 (b) f_6 (c) f_8

Figure 6.24: Diversity of AMABC, BinABC and NormABC for selected binary-mapped problems



(a) f_5 (b) f_6



(c) f_8

Figure 6.25: Generating functions for selected divergent AMABC problems

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Minimization

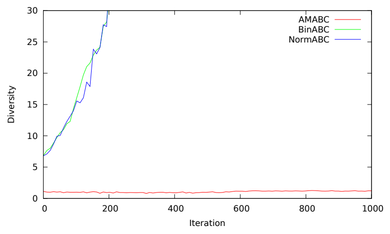
The results and p-values presented in Table 6.14 show that AMABC performed better than the binary algorithms for five of the ten problems. For the remaining problems, AMABC nearly achieved the same performance than the binary algorithms. The low dimensional KC problems were solved by all three algorithms. Despite the larger standard deviation for the larger dimensional problems, AMABC provided better fitnesses for several problems. Comparing the ranks of AMABC with BinABC, AMABC achieved a better rank of 1.4, compared to the 1.6 rank of BinABC. The same applies to the ranks of AMABC compared to NormABC.

Table 6.14: Binary minimization function results for AMABC compared to BinABC and NormABC, measured at a confidence of 95%. Bold p-values indicate statistical significance.

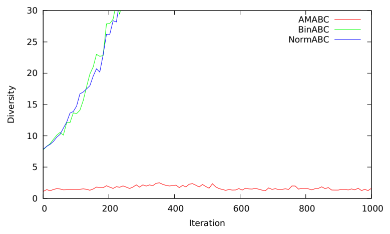
f	AMABC	BinABC	p-value	
			$H_1^{\neq}$	$H_1^{<}$
KC_4	-2000.0 ± 0.0	-2000.0 ± 0.0	1.0	1.0
KC_5	-3500.0 ± 0.0	-3500.0 ± 0.0	1.0	1.0
KC_6	-4660.0 ± 152.224879	-4750.0 ± 113.714706	**0.024452**	0.9976
KC_7	-5890.0 ± 442.056323	-6220.0 ± 191.905148	**0.000262**	0.999843
KC_8	-7650.0 ± 370.228261	-8170.0 ± 170.496233	$\mathbf{1.23 \times 10^{-8}}$	0.999999
Q_4	0.0 ± 0.0	2.533333 ± 1.041660	$\mathbf{1.38 \times 10^{-12}}$	$\mathbf{6.91 \times 10^{-13}}$
Q_5	0.2 ± 0.610257	5.533333 ± 1.136641	$\mathbf{1.16 \times 10^{-12}}$	$\mathbf{5.83 \times 10^{-13}}$
Q_6	2.533333 ± 1.166584	$12007.733333 \pm 14941.76433$	$\mathbf{1.15 \times 10^{-11}}$	$\mathbf{5.79 \times 10^{-12}}$
Q_7	4.2 ± 1.517711	30000.0 ± 0.0	$\mathbf{6.14 \times 10^{-13}}$	$\mathbf{3.07 \times 10^{-13}}$
Q_8	6.8 ± 2.265178	30000.0 ± 0.0	$\mathbf{6.3 \times 10^{-13}}$	$\mathbf{3.15 \times 10^{-13}}$
Avg Rank	1.4	1.6		

f	AMABC	NormABC	p-value	
			$H_1^{\neq}$	$H_1^{<}$
KC_4	-2000.0 ± 0.0	-2000.0 ± 0.0	1.0	1.0
KC_5	-3500.0 ± 0.0	-3500.0 ± 0.0	1.0	1.0
KC_6	-4660.0 ± 152.224879	-4740.0 ± 122.051430	**0.03050**	0.993223
KC_7	-5890.0 ± 442.056323	-6140.0 ± 165.258418	**0.001136**	0.9993
KC_8	-7650.0 ± 370.228261	-8210.0 ± 238.313036	$\mathbf{1.40 \times 10^{-8}}$	0.999999
Q_4	0.0 ± 0.0	2.733333 ± 1.112106	$\mathbf{1.86 \times 10^{-12}}$	$\mathbf{9.31 \times 10^{-13}}$
Q_5	0.2 ± 0.610257	5.533333 ± 1.357821	$\mathbf{1.62 \times 10^{-12}}$	$\mathbf{8.13 \times 10^{-13}}$
Q_6	2.533333 ± 1.166584	$17005.733333 \pm 15113.539659$	$\mathbf{8.11 \times 10^{-12}}$	$\mathbf{4.05 \times 10^{-12}}$
Q_7	4.2 ± 1.517711	30000.0 ± 0.0	$\mathbf{6.14 \times 10^{-13}}$	$\mathbf{3.07 \times 10^{-13}}$
Q_8	6.8 ± 2.265178	30000.0 ± 0.0	$\mathbf{6.30 \times 10^{-13}}$	$\mathbf{3.15 \times 10^{-13}}$
Avg Rank	1.4	1.6		

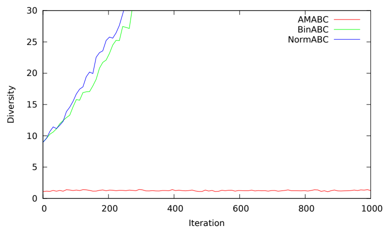
Figure 6.27 illustrates the performance of AMABC, BinABC and NormABC on the binary minimization problems. The graphs show that for all problems, except problems KC_6 to KC_8, AMABC achieved a solution faster than the binary algorithms. As illustrated in Figure 6.26, the diversity of the AMABC ant colony never converges. For the Q problems, the ant colony displays divergent behavior, except for Q_5 which diverges initially, only to converge later.



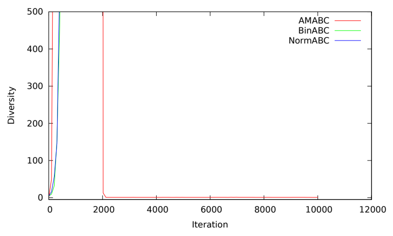
(a) KC_6



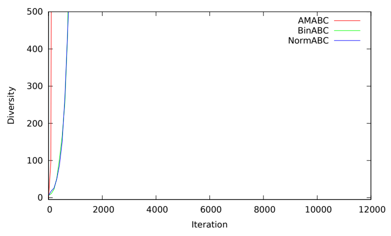
(b) KC_7



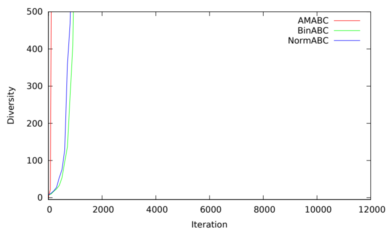
(c) KC_8



(d) Q_5



(e) Q_7



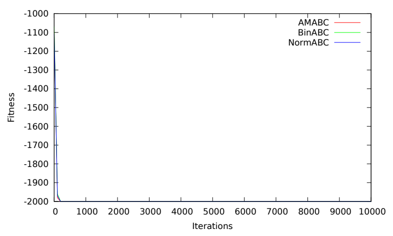
(f) Q_8

Figure 6.26: Diversity of AMABC, BinABC and NormABC for selected binary minimization problems

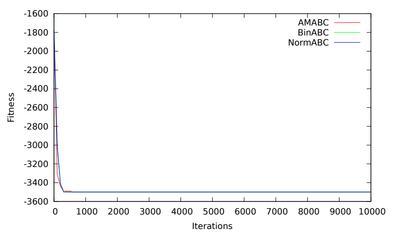
Maximization

From the calculated p-values provided in Table 6.15, AMABC outperformed both binary algorithms in five of the maximization problems. For the MDK problems, BinABC and NormABC could not reach the target solution whereas AMABC could. The larger dimensional KT problems provided similar results for all algorithms, including mean fitness and standard deviations. AMABC provided a better solution than the binary algorithms for the IPD problems. Overall, comparing the ranks of AMABC with BinABC, AMABC achieved a better rank of 1.2 compared to the BinABC rank of 1.7. The same applies to

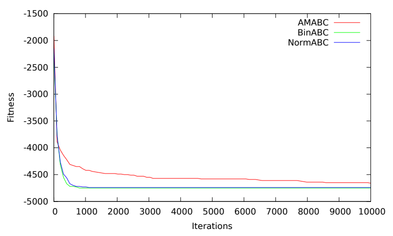
UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA



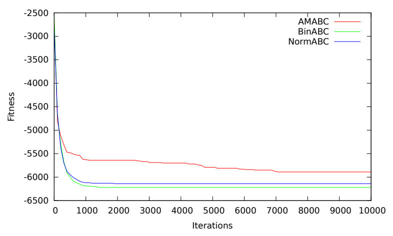
(a) KC_4



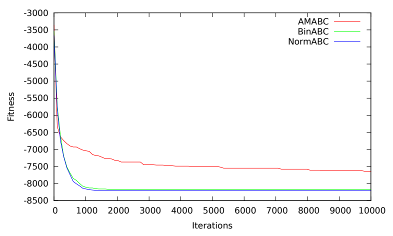
(b) KC_5



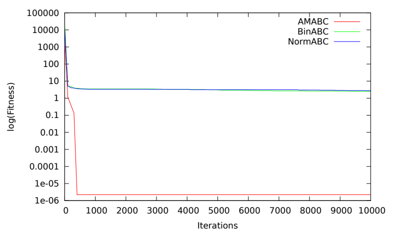
(c) KC_6



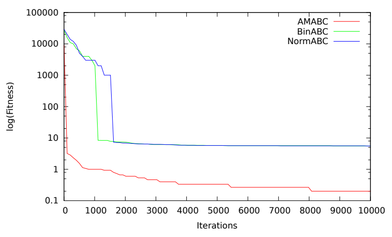
(d) KC_7



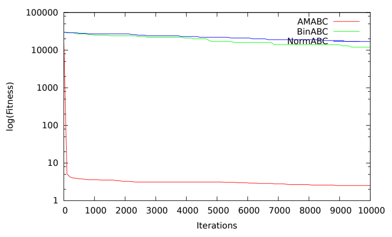
(e) KC_8



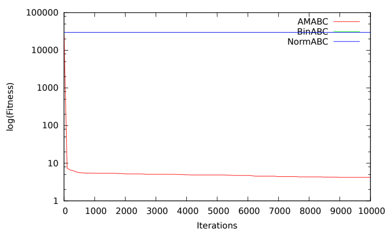
(f) Q_4



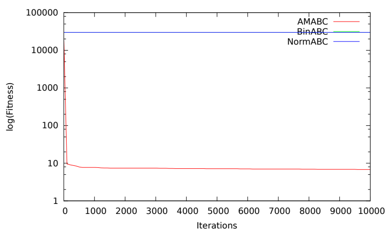
(g) Q_5



(h) Q_6



(i) Q_7



(j) Q_8

Figure 6.27: Binary minimization problem performances for AMABC, BinABC and NormABC

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

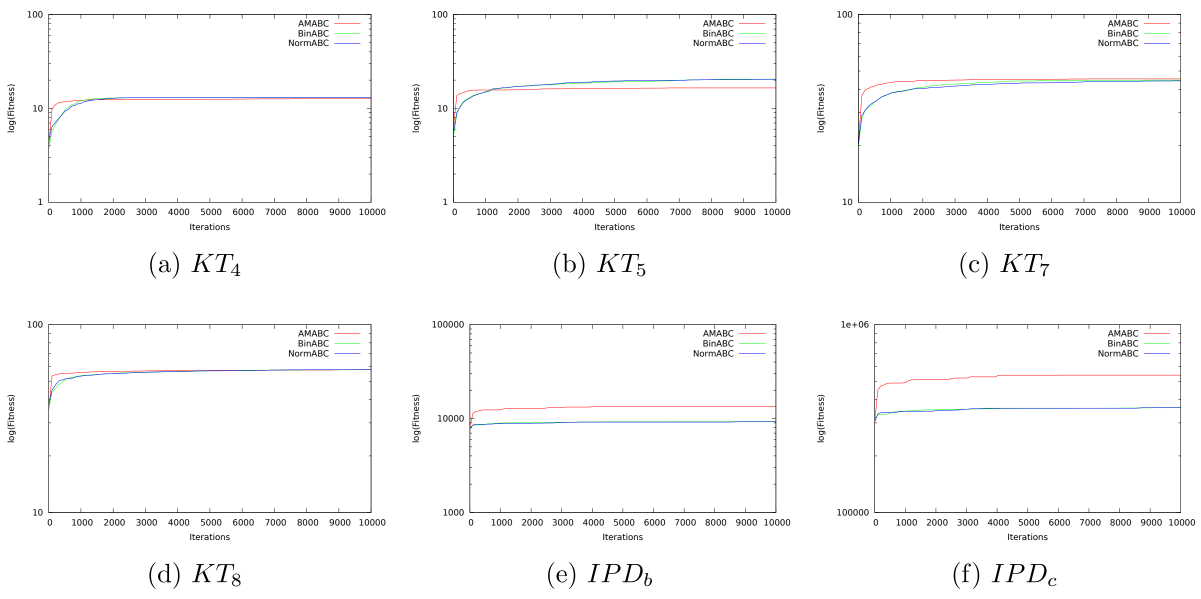
the ranks of AMABC compared to NormABC.

Table 6.15: Binary maximization function results for AMABC compared to BinABC and NormABC, measured at a confidence of 95%. Bold p-values indicate statistical significance.

f	AMABC	BinABC	p-value	
			$H_1^{\neq}$	$H_1^{>}$
MDK_{10}	295.0 ± 0.0	294.933333	0.492666	0.2460601
MDK_{20}	1024.0 ± 0.0	$1002.833333 \pm 15.848601$	$\mathbf{1.57 \times 10^{-11}}$	$\mathbf{7.88 \times 10^{-12}}$
KT_4	12.766666 ± 0.430183	13.0 ± 0.0	**0.01046**	0.997266
KT_5	16.533333 ± 0.776079	20.566666 ± 1.0063	$\mathbf{1.22 \times 10^{-11}}$	0.999999
KT_6	32.633333 ± 0.556053	32.333333 ± 0.546672	**0.034011**	**0.017175**
KT_7	45.4 ± 0.932183	44.733333 ± 1.311312	**0.033795**	**0.016847**
KT_8	57.6 ± 1.132589	57.666666 ± 1.124441	0.957731	0.479573
IPD_b	13526.133 ± 3383.968	9294.733 ± 340.297	$\mathbf{2.67 \times 10^{-4}}$	$\mathbf{1.25 \times 10^{-4}}$
IPD_c	$538664.466 \pm 139798.097$	361111.2 ± 12998.231	$\mathbf{1.66 \times 10^{-4}}$	$\mathbf{1.14 \times 10^{-4}}$
Avg Rank	1.2	1.7		
f	AMABC	NormABC	p-value	
			$H_1^{\neq}$	$H_1^{>}$
MDK_{10}	295.0 ± 0.0	294.933333 ± 0.253708	0.491489	0.245543
MDK_{20}	1024.0 ± 0.0	1006.3 ± 14.290459	$\mathbf{5.68 \times 10^{-9}}$	$\mathbf{2.84 \times 10^{-9}}$
KT_4	12.766666 ± 0.430183	13.0 ± 0.0	**0.010373**	0.997266
KT_5	16.533333 ± 0.776079	20.4 ± 1.037237	$\mathbf{1.23 \times 10^{-11}}$	0.999999
KT_6	32.633333 ± 0.556053	32.333333 ± 0.546672	**0.0343**	**0.01704**
KT_7	45.4 ± 0.932183	44.333333 ± 1.26854	**0.0008**	**0.0003**
KT_8	57.6 ± 1.132589	57.666666 ± 1.26854	0.590228	0.704171
IPD_b	13526.133 ± 3383.968	9247.0 ± 873.568	$\mathbf{1.66 \times 10^{-4}}$	$\mathbf{1.0 \times 10^{-4}}$
IPD_c	$538664.466 \pm 139798.097$	361728.666 ± 33495.297	$\mathbf{7.5 \times 10^{-5}}$	$\mathbf{1.26 \times 10^{-4}}$
Avg Rank	1.2	1.7		

Figure 6.28 illustrates the performance of AMABC, BinABC and NormABC for selected binary maximization problems. For most of the problems there was almost no difference between AMABC compared to BinABC, as well as AMABC compared to NormABC. The IPD problems, together with MDK_{20}, indicated that AMABC provided a better performance, compared to the binary algorithm variants.

Ant colony diversity for selected binary maximization problems is illustrated in Figure 6.29. These graphs shows that ant colonies of BinABC and NormABC show divergent behavior with AMABC diversity hovering around zero and indicating that AMABC could exploit found solutions, unlike BinABC and NormABC.



(a) KT_4 (b) KT_5 (c) KT_7

(d) KT_8 (e) IPD_b (f) IPD_c

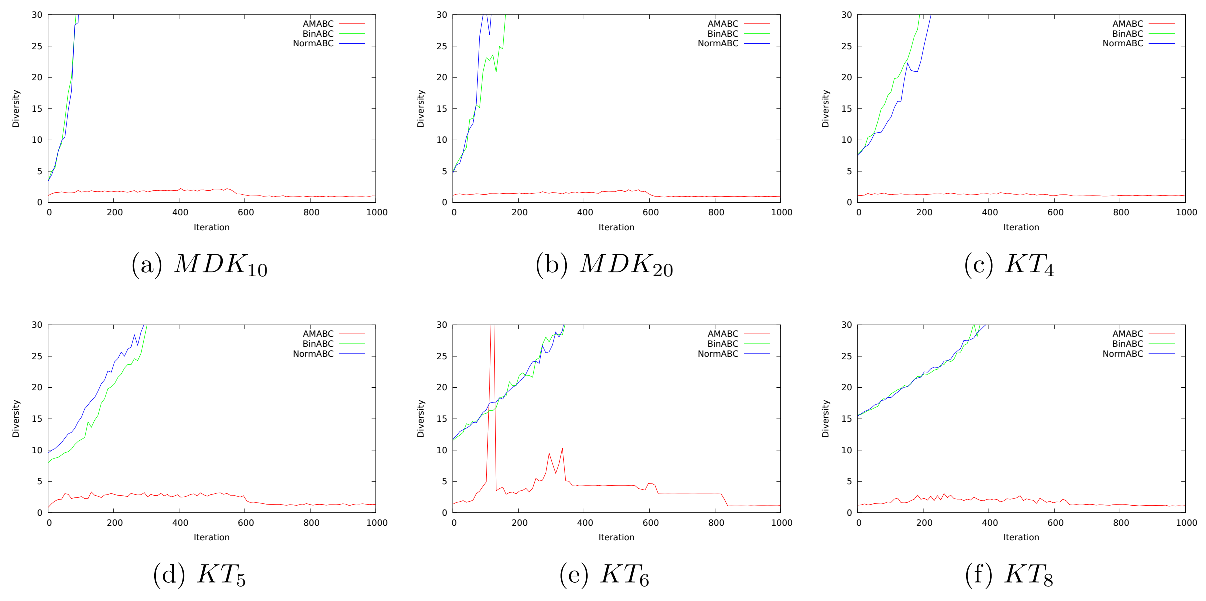
Figure 6.28: Binary maximization problem performances for AMABC, BinABC and NormABC.

6.6.3 Overall outcome

The average ranks indicate poor performance for AMABC for the binary-mapped continuous-valued problems, and better performances for the binary problems compared to the binary algorithm variants.

6.7 Comparison of angle modulation algorithms to binary algorithm variants

The main objective of this section is to evaluate the null hypothesis that the AM algorithms overall performed better than their binary counterparts. For the purpose of this comparison, the algorithms are ranked based on their performance for each individual problem. From these individual rankings, an overall rank is then calculated for AM and non-AM algorithms. Table 6.16 provides the ranks for the binary-mapped continuous-valued problems, while Tables 6.17 and 6.18 respectively rank the algorithms



(a) MDK_{10} (b) MDK_{20} (c) KT_4

(d) KT_5 (e) KT_6 (f) KT_8

Figure 6.29: ABC algorithm diversity graphs for selected binary maximization problems

for the binary minimization and maximization problems. Note that these tables label AM algorithms with an A and binary algorithms with a B. The symbol N indicates the normalized algorithms. Subscripts 1 to 5 to these labels respectively represent the algorithms GA, EP, DE, PSO and ABC respectively.

6.7.1 Continuous-valued problems

From the list of ranks per problem provided in Table 6.16, f_8 is the only problem for which the AM algorithms have a worse rank than the non-AM algorithms. For each algorithm over all problems, the rank can be used to determine a preference of algorithms for the binary-mapped continuous-valued problems. The resulting preference is:

$$\begin{aligned} AMGA = AMDE > AMPSO > NormDE > NormABC > ABC \\ > BinABC > BinDE > BinGA > AMEP > BinPSO > BinEP \end{aligned} \quad (6.1)$$

Considering a Mann-Whitney U test on the average ranks per problem, the null hypothesis that there is no significant difference in performance between AM and non-AM algorithms is rejected as a significant difference is observed with a resulting p-value of

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

0.0023. As a result the alternatve hypothesis that AM algorithms provide an improved performance over the non-AM algorithms is accepted, with AM algorithms improving on non-AM algorithm results for the problems considered. The overall rank of the AM algorithms is also better than the non-AM algorithm rank.

Table 6.16: Rankings for algorithms on continuous-valued benchmark problems. Algorithms labeled A are AM algorithms, with B referring to binary algorithms and N indicating the normalized algorithms. Subscripts 1–5 respectively represent GA, EP, DE, PSO and ABC.

Problem	A_1	A_2	A_3	A_4	A_5	Avg Rank	B_1	B_2	B_3	N_3	B_4	B_5	N_5	Avg Rank
f_1	2	10	1	3	9	5.0	8	11	4	6	12	5	6	7.43
f_2	7	3	6	9	2	5.4	8	10	12	4	1	11	4	7.14
f_3	1	6	1	1	1	2.0	1	11	12	9	8	7	9	8.14
f_4	1	9	2	3	8	4.6	10	11	7	5	12	4	5	7.71
f_5	3	8	2	1	9	4.6	10	11	5	6	11	4	6	7.57
f_6	4	6	3	5	1	3.8	1	10	12	8	7	11	8	8.14
f_7	2	9	3	1	10	5.0	8	11	4	5	12	7	5	7.43
f_8	4	11	6	8	12	8.2	7	9	5	1	10	1	1	4.86
f_9	2	10	1	3	7	4.6	9	12	4	5	11	6	8	7.86
f_{10}	1	6	2	3	5	3.4	10	12	4	7	11	9	7	8.57
Avg Rank	2.7	7.8	2.7	3.7	6.4	4.66	7.2	10.8	6.9	5.6	9.5	6.5	5.9	7.49
Preference	1.5	10	1.5	3	6	-	9	12	8	4	11	7	5	-

6.7.2 Binary-valued problems

Ranking results for the evaluated algorithms on both sets of binary-valued benchmarks problems follows. Tabulated results for the maximization problems follow the minimization results.

Minimization

Table 6.17 provides the ranking for algorithms on the considered problems. The rankings per problem show that AM algorithms perform worse on problems KC_6, KC_7 and KC_8. The overall algorithm rankings provide an algorithm preference for the binary minimization problems:

$$\begin{aligned} &AMGA > AMPSO > AMDE > BinDE = NormDE = BinPSO \\ &\quad > AMEP = AMABC = BinGA > BinABC > NormABC > BinEP \end{aligned} \tag{6.2}$$

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Problems KC_6 to KC_8 resulted in poor AM algorithm results. AMEP and AMABC additionally provided poor performance results for the Q problems. These poor results are due to the divergent behavior of the population or ant colony. A Mann-Whitney U test on the average ranks showed that the null hypothesis, that there is no significant difference in the performance of the AM and non-AM algorithms, is accepted based on the calculated p-value of 0.0708. The alternative hypothesis that AM algorithms performed better than the non-AM algorithms is therefore rejected. Furthermore, it is important to note that the overall ranking for AM algorithms is better than non-AM algorithms and that the first three preferred algorithms are AM algorithms.

Table 6.17: Rankings for algorithms on binary-valued minimization benchmark problems. Algorithms label A are AM algorithms, with B referring to binary algorithms and N indicating the normalized algorithms. Subscripts 1–5 respectively represent GA, EP, DE, PSO and ABC.

Problem	A_1	A_2	A_3	A_4	A_5	Avg Rank	B_1	B_2	B_3	N_3	B_4	B_5	N_5	Avg Rank
KC_4	1	1	1	1	1	1.0	1	1	12	1	1	1	1	2.57
KC_5	1	1	1	1	1	1.0	1	12	1	1	1	1	1	2.57
KC_6	9	11	10	8	7	9.0	4	12	2	6	1	2	4	4.43
KC_7	6	11	10	8	9	8.8	5	12	1	2	7	3	4	4.86
KC_8	5	11	7	5	9	7.4	8	12	1	2	10	4	3	5.71
Q_4	1	1	1	1	1	1.0	1	10	9	8	1	11	12	7.43
Q_5	1	1	1	1	6	2.0	9	10	7	8	1	11	11	8.14
Q_6	1	5	2	3	6	3.4	8	11	4	7	9	10	12	8.71
Q_7	1	4	1	3	5	2.8	7	8	6	8	8	8	8	7.57
Q_8	1	4	1	1	5	2.4	6	6	6	6	6	6	6	6.0
Avg Rank	2.7	5.0	3.5	3.2	5.0	3.88	5.0	9.4	4.9	4.9	4.5	5.7	6.2	5.8
Preference	1	8	3	2	8	-	8	12	5	5	5	10	11	-

Maximization

Table 6.18 presents average ranks for algorithms on the binary maximization problems. For the rankings per problem, the non-AM algorithms had a better performance for only problems KT_4, KT_5, KT_6, and KT_8. The rankings over all problems per algorithm produce the following algorithm preference:

$$\begin{aligned} &AMDE > AMGA > AMPSO > NormDE > NormABC > AMABC \\ &\quad > BinABC > BinDE > BinGA > AMEP > BinPSO > BinEP \end{aligned} \tag{6.3}$$

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Considering a Mann-Whitney U test on the average ranks, the null hypothesis that there is no significant difference between the AM and non-AM algorithms, is accepted based on the calculated p-value of 0.5. The alternatve hypothesis that the AM algorithms provide a significant improvement in performance over non-AM algorithms, is therefore rejected. The overall rankings for AM and non-AM algorithms are the same, but it should be noted that the algorithm preference indicates that the first three algorithms are AM algorithms.

Table 6.18: Rankings for algorithms on binary-valued maximization benchmark problems. Algorithms label A are AM algorithms, with B referring to binary algorithms and N indicating the normalized algorithms. Subscripts 1–5 respectively represent GA, EP, DE, PSO and ABC.

Problem	A_1	A_2	A_3	A_4	A_5	Avg Rank	B_1	B_2	B_3	N_3	B_4	B_5	N_5	Avg Rank
MDK_{10}	1	1	9	1	1	2.6	1	1	1	12	1	10	10	5.14
MDK_{20}	1	1	9	1	1	2.6	7	12	6	8	1	11	10	7.86
KT_4	10	8	12	6	7	8.6	9	11	5	4	1	1	1	4.57
KT_5	11	9	10	6	8	8.8	7	12	5	4	3	1	2	4.86
KT_6	10	4	10	3	5	6.4	2	8	12	9	1	6	6	6.29
KT_7	1	4	11	2	4	4.4	6	9	12	10	3	7	8	7.86
KT_8	8	11	12	7	6	8.8	2	10	3	9	1	3	3	4.43
IPD_b	12	7	8	2	1	6.0	11	9	6	5	10	3	4	6.86
IPD_c	12	7	8	2	1	6.0	10	9	6	5	11	4	3	6.86
Avg Rank	7.3	5.7	9.8	3.3	3.7	6.0	6.1	9	6.2	7.3	3.5	5.1	5.2	6.0
Preference	2	10	1	3	6	-	9	12	8	4	11	7	5	-

6.7.3 Summary of ranking results

The outcomes of the Mann-Whitney U tests for the AM algorithms indicate that the AM provides a significant difference over non-AM algorithms for the binary-mapped continuous-valued functions, and no significant difference for the binary functions.

However, note from equations (6.1), (6.2) and (6.3) that AM algorithms always achieved the first three positions in algorithm preference, with AMDE best for the binary maximization functions, AMGA best for binary minimization functions, and both AMDE and AMGA being best for the binary-mapped continuous-valued functions. An analysis of the poor performances of AMEP and AMABC is suggested for future work.

6.8 Scalability

The previous section provided results indicating that the AM methods are better, in general, than their non-AM counterparts. This section investigates the scalability of the AM approach, using PSO. Section 3.3 discussed some problems with binary representations, specifically the curse of dimensionality [86]. To quantify the scalability of AM, its ability to accurately match random bit strings of varying length is determined. More formally, the ability of the AM algorithm to solve the following optimization problem is evaluated:

$$\min f(x) = \sum_{i=1}^{n} (t_i - \text{AM}_i)^2 \tag{6.4}$$

where n is the length of the randomly generated bit string, t_i represents bit i within the random bit string and AM_i is the bit generated using AM. The smaller the value of the objective function, $f(x)$, the better the algorithm is at producing an arbitrary bit string, of varying lengths n.

Bit string lengths started with sizes of $n = 1$, increasing the size by a factor of two, up to a length of 2048. For each bit string length, a maximum of 50000 randomly generated bit strings were used. Figure 6.30 illustrates the accuracy in reproducing the randomly generated bit strings for the different bit sting sizes using equation (6.4). This scalability study used the AMPSO algorithm as PSO was the first algorithm to which AM was successfully applied, and was performed before AM was applied to other algorithms. The results indicate that AM is very scalable up to a length of 512, where an accuracy of 90% was still achieved. For bit strings of length 1024, the accuracy was 86%. For bit strings of length 2048, accuracy is at 68%. The deterioration in performance is significantly less than the increase in dimension (as illustrated in Figure 6.31).

It is thought that the scalability of AM can be improved by using two four-tuples sets: The first four-tuple set, representing the bit generating function's coefficients, is then used to produce the first $n/2$ bits, while the second four-tuple set is used to produce the last $n/2$ bits. While this increases the dimensionality of the AM algorithms, a dimension of eight is still significantly less than that of the corresponding binary representation with the added improved scalability (as illustrated in Figure 6.30). An impact analysis of including additional four-tuple sets is suggested as future work.

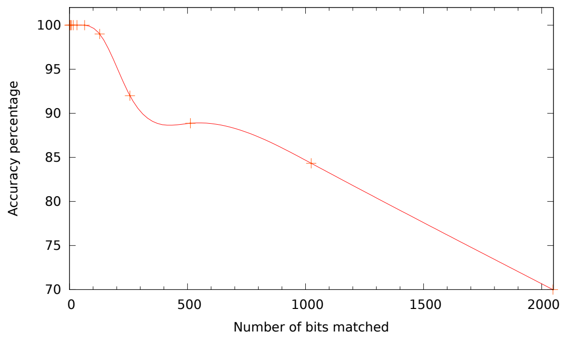


Figure 6.30: Accuracy of matched random bit strings

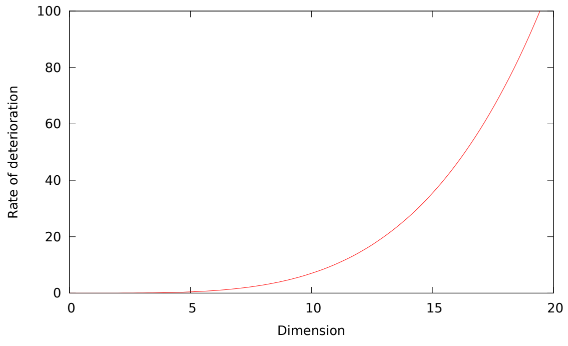


Figure 6.31: Rate of deterioration compared to an increase in dimension. Dimension values represent 2^n bits where n is the dimension value.

6.9 Summary

The aim of this chapter was to determine if AM is a viable approach to solve binary-valued optimization problems. Each AM algorithm and the associated binary algorithm variants were evaluated on three sets of problems, consisting of binary-mapped continuous-valued problems as well as binary minimization and maximization problems. An initial, but superficial, scalability analysis of AM was also conducted.

Compelling evidence has been provided to show that AM is a viable method to solve binary-valued optimization problems. It was shown that the three AM approaches, AMDE, AMGA and AMPSO, performed best. Based on the results, is was concluded that AMDE is the preferred AM approach to use.

Chapter 7

Conclusions

This chapter presents the conclusions of the work presented in this thesis. Section 7.1 presents the main findings of this study. Potential future research topics are given and enumerated in Section 7.2.

7.1 Summary of Conclusions

This thesis proposed angle modulation (AM) as an approach to solve binary-valued optimization problems in continuous-valued space. To solve a binary-valued optimization problems within a continuous space, a bit generating function is found within the continuous-valued space which is then used to generate a bit string solution to the binary-valued optimization problem. The consequence is then that algorithms developed for continuous-valued spaces can now be used to solve optimization problems within the binary-valued space, without the need to adapt the algorithm to be able to operate in binary-valued space.

Continuous-valued algorithms examined include the GA, EP, DE, PSO and ABC. Each continuous-valued algorithm was then compared to its non-AM, or binary algorithm version, with the DE and ABC also compared to normalization algorithm variants.

Three different sets of benchmarks functions were used to evaluate the performance and accuracy of the algorithms. The set of continuous-valued benchmark functions consisted of ten individual functions, each defined with its own domain and dimensionality. The suite of binary-valued minimization problems included two chess-based assignment problems of increasing complexity, namely the knight's tour (KT) and n-Queens (Q) problems. Binary-valued maximization problems included the two variations of the multidimensional knapsack problem (MDK), the knight's tour (KT) with increasing complexity and the iterated prisoner's dilemma (IPD).

For each set of algorithms, a pairwise comparison between the AM algorithm and the binary algorithm variant was then performed, with the candidate solutions and bit string evaluation differing between the algorithms. The main conclusions for the comparisons, for each algorithm, are

- **Genetic algorithm:** The continuous-valued benchmark problems indicated that the AMGA was able to outperform the BinGA on most of the benchmark functions, except for three problems, which have a low dimensionality. Binary-valued minimization problems displayed a similar trend, with the AMGA providing better mean results for the problems but not always giving a result that is statistically significant. Binary maximization problems, however, showed that AMGA was not able to achieve the results obtained by BinGA.

- **Evolutionary programming:** Pairwise comparisons between AMEP and BinEP indicated improved results for the AMEP on the set of continuous-valued optimization problems. Binary-valued problems continued the trend with the AMEP improving on the BinEP for most of the binary minimization and maximization problems.

- **Differential evolution:** Comparisons occurred between AMDE and two different binary optimization algorithms, namely the BinDE and NormDE.

  The comparison between AMDE and BinDE showed an improvement for AMDE on the set of continuous-valued optimization problems, with only two problems not yielding an improvement that was statistically significant. The binary-valued problems also provided improved results for the set of binary minimization problems, but only for larger dimensions of the Q problem. The remaining minimization problem results were not an improvement, with three problems indicating that BinDE achieved better results. AMDE failed to provide an improvement with the set of binary maximization problems.

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

AMDE and NormDE comparisons also indicated an improved set of results for the continuous-valued benchmark functions, with a single function failing to yield an improvement. The binary-valued minimization problems indicated similar results to the comparison between AMDE and BinDE, where the same benchmark functions achieved improved results. With the binary-valued maximization problems, AMDE failed to achieve a performance that improves on the performance of NormDE.

- **Particle swarm optimization:** PSO and BinPSO comparisons resulted in an improved accuracy for seven of the continuous-valued benchmark functions with the remainder showing an improved performance by BinPSO. For the binary minimization problems accuracies for AMPSO and BinPSO were similar with only four functions providing improved results. Binary maximization problem results indicated similar performances between AMPSO and BinPSO.

- **Artificial bee colony:** As with the comparisons for AMDE, pairwise comparisons for AMABC occurred with BinABC and NormABC.

  AMABC and BinABC results were similar with five continuous-valued functions providing a noticeable improvement in accuracy for AMABC. Binary-valued minimization problems showed that AMABC could improve on the performance of BinPSO for only the Q problems. For the maximization binary problems, AMABC was able to provide improved results, but only for binary-valued problems of larger dimensions.

  AMABC and NormABC results indicated similar trends, with the exception of three problems where AMABC provided better performance. For the binary minimization problems improvements to the Q problems were possible. Results for binary maximization problems mirrored the problems improved on with AMABC and BinABC, where AMABC provided better results with the larger dimensional problems.

The performance accuracy of all AM algorithms were then compared to all non-AM algorithms, for each suite of benchmark functions. The results indicated that the AM approach did provide improvements across all benchmark problems, especially for the sets of continuous-valued and binary-valued minimization problems, with AMDE,

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

AMPSO and AMGA showed to be the preferred algorithms. Comparisons for the binary maximization problems provided no clear distinction between classes of AM and non-AM algorithms, but it was observed that all AM algorithms outperformed the respective binary algorithm variants.

As the length of a bit string for a candidate solution increased, a reduction in accuracy was observed for most of the binary algorithms. The scalability analysis of the AM provided results showing the deterioration in accuracy for bit strings generated through the AM approach. While performance does deteriorate, it is at a significantly slower rate than the increase in dimensionality.

From all the provided results and discussions, this study finds that AM is a viable approach to solve binary-valued optimization problems through the use of continuous optimization algorithms.

7.2 Future work

Potential research that may follow from this thesis includes:

- This study used one mapping function, evolving the four coefficients of that function. Future research should investigate alternative mapping functions.
- This study focused on applying continuous algorithms using continuous-valued decision variables to solve binary problems. Applying AM to mixed problem space representations should be investigated.
- Empirically associate problem types with specific AM algorithms that are known to perform well on such problems.
- Application of AM to niching and algorithms which maintain multiple populations.
- Possibility of using a set of heterogeneous mapping functions for AM where the choice of (possibly multiple) mapping functions is also part of the candidate solution to an optimization problem.
- To analyze the behavior of AMEP and AMABC in more detail to understand the bad performance compared to other AM approaches.

- Analyze effects on performance and accuracy through the addition of additional four-tuple sets for very long bit strings.

Bibliography

[1] E.H. Armstrong. A method of reducing disturbances in radio signaling by a system of frequency modulation. *Proceedings of the IEEE*, 72(8):1042–1062, 1984, doi:10.1109/PROC.1984.12971.

[2] R. Axelrod. *The Evolution of Cooperation.* Basic Books, 1984.

[3] R. Axelrod. Tit-for-tat strategies. *Routledge Encyclopedia of International Political Economy*, 2001.

[4] T. Bäck. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms.* Oxford University Press, 1996.

[5] T. Bäck, U. Hammel, and H.-P. Schwefel. Evolutionary computation: Comments on the history and current state. *IEEE Transactions on Evolutionary Computation*, 1(1):3–17, 1997, doi:10.1109/4235.585888.

[6] J.E. Baker. Reducing bias and inefficiency in the selection algorithm. In *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*, pages 14–21. L. Erlbaum Associates Inc., 1987.

[7] B. Basturk and D. Karaboga. An artificial bee colony (ABC) algorithm for numeric function optimization. In *Proceedings of the IEEE Swarm Intelligence Symposium*, 2006.

[8] R. Bellman. *Adaptive process control: A guided tour.* Princeton University Press, 1961.

[9] R. Bellman. *Dynamic Programming.* Dover Publications, 2003.

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

[10] G. Bilchev and I.C. Parmee. The Ant Colony Metaphor for Searching Continuous Design Spaces. In T. Fogarty, editor, *Proceedings of the AISB Workshop on Evolutionary Computation*, volume 993, pages 25–39. Springer-Verlag, 1995.

[11] G. Bilchev and I.C. Parmee. Constrained Optimisation with an Ant Colony Search Model. In *Proceedings of Adaptive Computing in Engineering Design and Control*, pages 145–151, 1996.

[12] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence*. Oxford University Press, 1999.

[13] H.J. Bremermann. *Optimization through Evolution and Recombination*. Spartan Books, 1962.

[14] R. Brits, A.P. Engelbrecht, and F. van den Bergh. A niching particle swarm optimizer. In *Proceedings of the Forth Asia-Pacific Conference on Simulated Evolution and Learning (SEAL '02)*, volume 2, pages 692–696, 2002.

[15] J.A. Brown and D.A. Ashlock. Domination in iterated prisoner's dilemma. In *Proceedings of the Twenty-Forth Canadian Conference on Electrical and Computer Engineering.*, pages 001125–001128, 2011, doi:10.1109/CCECE.2011.6030637.

[16] H. Bu, S. Zheng, and J. Xia. Genetic algorithm based semi-feature selection method. In *International Joint Conference on Bioinformatics, Systems Biology and Intelligent Computing.*, pages 521–524, 2009, doi:10.1109/IJCBS.2009.38.

[17] Bukhari. Using genetic algorithms to develop strategies for the prisoners dilemma. *Asian Journal of Information Technology*, 8(5):866–871, 2006.

[18] A. Carlisle and G. Dozier. *An off-the-shelf PSO*, volume 1, pages 1–6. Indianapolis, IN: Purdue School of Engineering and Technology, IUPUI, 2001.

[19] G.E. Carlson. *Signal and Linear System Analysis*. John Wiley & Sons, 2nd edition, 1998.

[20] K. Chellapilla. Combining mutation operators in evolutionary programming. *IEEE Transactions on Evolutionary Computation*, 2(3):91–96, 1998, doi:10.1109/4235.735431.

[21] T. Chen, J. He, G. Sun, G. Chen, and X. Yao. A new approach for analyzing average time complexity of population-based evolutionary algorithms on unimodal problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 39(5):1092–1106, 2009, doi:10.1109/TSMCB.2008.2012167.

[22] W. Chen, J. Zhang, H.S.H. Chung, W. Zhong, W. Wu, and Y. Shi. A novel set-based particle swarm optimization method for discrete optimization problems. *IEEE Transactions on Evolutionary Computation*, 14(2):278–300, 2010, doi:10.1109/TEVC.2009.2030331.

[23] S.C. Chiam, C.K. Goh, and K.C. Tan. Issues of binary representation in evolutionary algorithms. In *Proceedings of the IEEE Conference on Cybernetics and Intelligent Systems*, volume 7–9, pages 1–8, 2006, doi:10.1109/ICCIS.2006.252329.

[24] D. Choi and S. Oh. A new mutation rule for evolutionary programming motivated from backpropagation learning. *IEEE Transactions on Evolutionary Computation*, 4(2):188–190, 2000, doi:10.1109/4235.850659.

[25] M. Clerc. Binary Particle Swarm Optimisers: toolbox, derivations, and mathematical insights. Technical report, 2005.

[26] M. Clerc and J. Kennedy. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1):58–73, 2002, doi:10.1109/4235.985692.

[27] T. Cloete, A.P. Engelbrecht, and G. Pamparà. CIlib: A collaborative framework for Computational Intelligence algorithms - Part II. In *Proceedings of the IEEE International Joint Conference on Neural Networks. IJCNN 2008.*, pages 1764–1773, 2008, doi:10.1109/IJCNN.2008.4634037.

[28] C.A. Coello Coello, E. Luna, and A. Aguirre. Use of particle swarm optimization to design combinational logic circuits. In AAndy Tyrrell, Pauline Haddow, and Jim Torresen, editors, *Evolvable Systems: From Biology to Hardware*, volume 2606 of *Lecture Notes in Computer Science*, pages 398–409. Springer Berlin / Heidelberg, 2003, doi:10.1007/3-540-36553-2_36.

[29] C.A. Coello Coello, E.H. Luna, and A.H. Aguirre. A comparative study of encodings to design combinational logic circuits using particle swarm optimization. In *Proceedings of the NASA/DoD Conference on Evolvable Hardware*, pages 71–78, 2004, doi:10.1109/EH.2004.1310811.

[30] P.A. Corning. The re-emergence of emergence: A venerable concept in search of a theory. *Complexity*, 7(6):18–30, 2002, doi:10.1002/cplx.10043.

[31] P. Cull. Tours of graphs, digraphs, and sequential machines. *IEEE Transactions on Computers*, C-29(1):50–54, 1980, doi:10.1109/TC.1980.1675456.

[32] O. Dahl, E. W. Dijkstra, and C. A. R. Hoare. *Structured Programming*. Academic Press, 1972.

[33] C. R. Darwin. *On the Origin of Species by Means of Natural Selection or Preservation of Favoured Races in the Struggle for Life*. 1859.

[34] K. de Jong. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan, 1975.

[35] K. Deb and H. Beyer. Self-adaptive genetic algorithms with simulated binary crossover. *Evolutionary Computation*, 9:197–221, 2001, doi:10.1162/106365601750190406.

[36] K. Deb, D. Joshi, and A. Anand. Real-coded evolutionary algorithms with parent-centric recombination. In *Proceedings of Congress on Evolutionary Computation.*, volume 1, pages 61–66, 2002, doi:10.1109/CEC.2002.1006210.

[37] S. Dirakkhunakon and Y. Suansook. Simulated annealing with iterative improvement. In *2009 International Conference on Signal Processing Systems*, pages 302–306, 2009, doi:10.1109/ICSPS.2009.61.

[38] M. Dorigo. *Ant Colony Optimization*. MIT Press, 2004.

[39] M. Dresher. *The mathematics of games of strategy: theory and applications*. Dover books on advanced mathematics. Dover, 1981.

[40] R. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *Proceedings of the Sixth International Symposium on Micro Machine and Human Science. MHS '95.*, pages 39–43, 1995, doi:10.1109/MHS.1995.494215.

[41] R. Eberhart, P. Simpson, and R. Dobbins. *Computational intelligence PC tools.* Academic Press Professional, Inc., 1996.

[42] R.C. Eberhart and Y. Shi. Particle swarm optimization: developments, applications and resources. In *Proceedings of the Congress on Evolutionary Computation.*, volume 1, pages 81–86, 2001, doi:10.1109/CEC.2001.934374.

[43] A. P. Engelbrecht. *Fundamentals of Computational Swarm Intelligence.* John Wiley & Sons, 2005.

[44] A.P. Engelbrecht and G. Pamparà. Binary differential evolution strategies. In *Proceedings of the IEEE Congress on Evolutionary Computation.*, pages 1942–1947, 2007, doi:10.1109/CEC.2007.4424711.

[45] L. J. Eshelman and J. D. Schaffer. Real-coded genetic algorithms and interval-schemata. In Darrell L. Whitley, editor, *Foundation of Genetic Algorithms 2*, pages 187–202. Morgan Kaufmann., 1993.

[46] D.C. Fisher. On the n x n Knight Cover Problem. *Ars Combinatoria*, 69:255–274, 2003.

[47] L.J. Fogal, A.J. Owens, and M.J. Walsh. *Artificial Intelligence through Simulated Evolution.* John Wiley, 1966.

[48] D. B. Fogel, editor. *Evolutionary Computation (The Fossil Record).* Wiley-IEEE Press, 1998.

[49] D.B. Fogel, L.F. Fogel, and V.W. Porto. Evolutionary Programming for Training Neural Networks. In *Proceedings of the IEEE International Joint Conference on Neural Networks*, volume 1, pages 601–605, 1990.

[50] D.B. Fogel, L.J. Fogel, and J.W. Atmar. Meta-evolutionary programming. In *Conference Record of the Twenty-Fifth Asilomar Conference on Signals, Systems and Computers*, volume 1, pages 540–545, 1991, doi:10.1109/ACSSC.1991.186507.

[51] L.J. Fogel. Autonomous automata. *Industrial Research*, 4:14–19, 1962.

[52] D.E. Foster and S.W. Seeley. Automatic tuning, simplified circuits, and design practice. *Proceedings of the Institute of Radio Engineers*, 25(3):289–313, 1937, doi:10.1109/JRPROC.1937.228940.

[53] N. Franken. PSO-Based Coevolutionary Game Learning. Master's thesis, Department of Computer Science, University of Pretoria, South Africa, 2004.

[54] N. Franken and A.P. Engelbrecht. Investigating binary pso parameter influence on the knights cover problem. In *The 2005 IEEE Congress on Evolutionary Computation.*, volume 1, pages 282–289, 2005, doi:10.1109/CEC.2005.1554696.

[55] N. Franken and A.P. Engelbrecht. Particle swarm optimization approaches to coevolve strategies for the iterated prisoner's dilemma. *IEEE Transactions on Evolutionary Computation*, 9(6):562–579, 2005, doi:10.1109/TEVC.2005.856202.

[56] A. S. Fraser. Simulation of Genetic Systems by Automatic Digital Computers I: Introduction. *Australian Journal of Biological Science*, 10:484–491, 1957.

[57] A. S. Fraser. Simulation of Genetic Systems by Automatic Digital Computers II: Effects of Linkage on Rates of Advance Under Selection. *Australian Journal of Biological Science*, 10:492–499, 1957.

[58] P. Gancarski and A. Blansche. Darwinian, lamarckian, and baldwinian (co)evolutionary approaches for feature weighting in -means-based algorithms. *IEEE Transactions on Evolutionary Computation*, 12(5):617–629, 2008, doi:10.1109/TEVC.2008.920670.

[59] M. Glomba, T. Filak, and H. Kwasnicka. Discovering effective strategies for the iterated prisoner's dilemma using genetic algorithms. In *Proceedings of the Fifth International Conference on Intelligent Systems Design and Applications.*, pages 356–361, 2005, doi:10.1109/ISDA.2005.38.

[60] J. Golbeck. Evolving strategies for the prisoner's dilemma. Technical report, Computer Science Department, University of Maryland, College Park.

[61] D. Goldberg and K. Deb. *A comparative analysis of selection scheme used in genetic algorithms*. Foundations of Genetic Algorithms. Morgan Kaufmann Publishers, 1991.

[62] W. Gong, Z. Cai, X Lu, and S. Jiang. A new mutation operator based on the t probability distribution in evolutionary programming. In *Proceedings of the Fifth IEEE International Conference on Cognitive Informatics. ICCI 2006.*, volume 2, pages 675–679, 2006, doi:10.1109/COGINF.2006.365569.

[63] V.S. Gordon and T.J. Slocum. The knight's tour - evolutionary vs. depth-first search. In *Proceedings of the Congress on Evolutionary Computation.*, volume 2, pages 1435–1440, 2004, doi:10.1109/CEC.2004.1331065.

[64] J. Gosling, B. Joy, G.L. Steele, and G. Bracha. *The Java Language Specification*. Addison-Wesley, 3 edition, 2005.

[65] X. Guo, D. He, and G. Liu. An algorithm based on chaotic genetic algorithm for 0-1 knapsack problem. In *International Conference on Biomedical Engineering and Computer Science*, pages 1–3, 2010, doi:10.1109/ICBECS.2010.5462446.

[66] R.W. Hamming. Error Detecting and Error Codes. Technical Report Vol. XXVI, No. 2, The Bell System Technical Journal, 1950.

[67] J. Hartmanis. Computers and intractability: A guide to the theory of np-completeness (michael r. garey and david s. johnson). *SIAM Review*, 24(1):90–91, 1982, doi:10.1137/1024022.

[68] K. Hattori, Y. Narita, Y. Kashimori, and T. Kambara. Self-organized critical behavior of fish school and emergence of group intelligence. In *Proceedings on the Sixth international Conference on Neural Information Processing. ICONIP '99.*, volume 2, pages 465–470, 1999, doi:10.1109/ICONIP.1999.845639.

[69] R. Hinterding. Gaussian mutation and self-adaption for numeric genetic algorithms. In *IEEE International Conference on Evolutionary Computation*, volume 1, page 384, 1995, doi:10.1109/ICEC.1995.489178.

[70] J.H. Holland. *Adaption in Natural and Artificial Systems.* University of Michigan Press, 1975.

[71] G. Hughes. On the mean accuracy of statistical pattern recognizers. *IEEE Transactions on Information Theory*, 14(1):55–63, 1968.

[72] H. Ishibuchi and N. Namikawa. Evolution of cooperative behavior in the iterated prisoner's dilemma under random pairing in game playing. In *Proceedings of the IEEE Congress on Evolutionary Computation.*, volume 3, pages 2637–2644, 2005, doi:10.1109/CEC.2005.1555025.

[73] H. Ishibuchi and N. Namikawa. Evolution of iterated prisoner's dilemma game strategies in structured demes under random pairing in game playing. *IEEE Transactions on Evolutionary Computation*, 9(6):552–561, 2005, doi:10.1109/TEVC.2005.856198.

[74] C. Jiang and C. Wang. Improved evolutionary programming with dynamic mutation and metropolis criteria for multi-objective reactive power optimisation. *IEEE Proceedings on Generation, Transmission and Distribution*, 152(2):291–294, 2005, doi:10.1049/ip-gtd:20045007.

[75] T. Jones. Crossover, Macromutation, and Population-based Search. In L. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 73–80, 1995.

[76] R. Joshi and A.C. Sanderson. Minimal representation multisensor fusion using differential evolution. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 29(1):63–76, 1999, doi:10.1109/3468.736361.

[77] D. Karaboga. An idea based on honey bee swarm for numerical optimization. Technical Report TR-06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

[78] D. Karaboga and B. Basturk. *Artificial Bee Colony (ABC) Optimization Algorithm for Solving Constrained Optimization Problems*, volume 4529 of *Lecture Notes in Computer Science*, pages 789–798. Springer Berlin, 2007, doi:10.1007/978-3-540-72950-1.

[79] D. Karaboga and B. Basturk. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. *Journal of Global Optimization*, 39(3):459–471, 2007, doi:10.1007/s10898-007-9149-x.

[80] D. Karaboga and B. Basturk. On the performance of artificial bee colony (abc) algorithm. *Applied Soft Computing*, 8(1):687–697, 2008, doi:10.1016/j.asoc.2007.05.007.

[81] J. Kennedy. Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance. In *Proceedings of the Congress on Evolutionary Computation.*, volume 3, pages 3 vol. (xxxvii+2348), 1999, doi:10.1109/CEC.1999.785509.

[82] J. Kennedy and R.C. Eberhart. Particle Swarm Optimisation. In *Proceedings of the International Conference on Neural Networks*, pages 1942–1948, 1995.

[83] J. Kennedy and R.C. Eberhart. A discrete binary version of the particle swarm algorithm. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics.*, volume 5, pages 4104–4108, 1997, doi:10.1109/ICSMC.1997.637339.

[84] J. Kennedy and R. Mendes. Population structure and particle swarm performance. In *Proceedings of the Congress on Evolutionary Computation.*, volume 2, pages 1671–1676, 2002, doi:10.1109/CEC.2002.1004493.

[85] J. Kim, H. Chae, J. Jeon, and S. Lee. Identification and control of systems with friction using accelerated evolutionary programming. *IEEE Control Systems*, 16(4):38–47, 1996, doi:10.1109/37.526914.

[86] K. Koppen. The curse of dimensionality. In *Fifth Online World Conference on Soft Computing in Industrial Applications (WSC5).September 4-18*, 2000.

[87] S. Koziel and Z. Michalewicz. Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization. *Evolutionary Computation*, 7(1):19–44, 1999, doi:10.1162/evco.1999.7.1.19.

[88] R.A. Krohling and L. dos Santos Coelho. PSO-E: Particle Swarm with Exponential Distribution. In *IEEE Congress on Evolutionary Computation.*, pages 1428–1433, 2006, doi:10.1109/CEC.2006.1688476.

[89] U. Kutschera. A comparative analysis of the darwin-wallace papers and the development of the concept of natural selection. *Theory in Biosciences*, 122:343–359, 2003, doi:10.1007/s12064-003-0063-6.

[90] P.L. Lanzi. Fast feature selection with genetic algorithms: a filter approach. In *IEEE International Conference on Evolutionary Computation.*, pages 537–540, 1997, doi:10.1109/ICEC.1997.592369.

[91] E.C. Laskari, K.E. Parsopoulos, and M.N. Vrahatis. Particle swarm optimization for integer programming. In *Proceedings of Congress on Evolutionary Computation.*, volume 2, pages 1582–1587, 2002, doi:10.1109/CEC.2002.1004478.

[92] C. Lee and X. Yao. Evolutionary programming using mutations based on the levy probability distribution. *IEEE Transactions on Evolutionary Computation*, 8(1):1–13, 2004, doi:10.1109/TEVC.2003.816583.

[93] Y. Li, T-J. Wu, and D.J. Hill. An Accelerated Ant Colony Algorithm for Complex Nonlinear System Optimization. In *Proceedings of the IEEE International Symposium on Intelligent Control*, pages 709–713, 2003.

[94] K. Liang, X. Yao, and C. Newton. Lamarckian evolution in global optimization. In *Twenty-sixth Annual Conference of the IEEE Industrial Electronics Society*, volume 4, pages 2975–2980, 2000, doi:10.1109/IECON.2000.972471.

[95] M. Locatelli. A note on the griewank test function. *Journal of Global Optimization*, 25(2):169–174, 2003, doi:10.1023/A:1021956306041.

[96] H. Long, W. Xu, J. Sun, and W. Ji. Multiple sequence alignment based on a binary particle swarm optimization algorithm. In *Fifth International Conference on Natural Computation.*, volume 3, pages 265–269, 2009, doi:10.1109/ICNC.2009.238.

[97] E.H. Luna, C.A. Coello Coello, and A.H. Aguirre. On the use of a population-based particle swarm optimizer to design combinational logic circuits. In *Proceed-*

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

ings of the NASA/DoD Conference on Evolvable Hardware, pages 183–190, 2004, doi:10.1109/EH.2004.1310829.

[98] H.B. Mann and D.R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *Annals of Mathematical Statistics*, 1947.

[99] I. Martinjak and M. Golub. Comparison of heuristic algorithms for the n-queen problem. In *Proceedings of the Twenty-ninth International Conference on Information Technology Interfaces. ITI 2007.*, pages 759–764, 2007, doi:10.1109/ITI.2007.4283867.

[100] Z Mason. Programming with stigmergy: Using swarms for construction. In *In Artificial Life VI: Proceedings of the Eighth International Conference on Artificial Life*, pages 371–374. MIT Press, 2002.

[101] M. Matsumoto and T. Nishimura. Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. Model. Comput. Simul.*, 8:3–30, 1998, doi:10.1145/272991.272995.

[102] R. Mendes, P. Cortez, M. Rocha, and J. Neves. Particle Swarms for Feedforward Neural Network Training. In *Proceedings of the IEEE International Joint Conference on Neural Networks*, pages 1895–1899, 2002.

[103] Z. Michalewicz. *Genetic algorithms + data structures = evolution programs (3rd ed.)*. Springer-Verlag, 1996.

[104] H. Mhlenbein and D. Schlierkamp-Voosen. Predictive models for the breeder genetic algorithm – i. continuous parameter optimization. *Evolutionary Computation*, 1:25–49, 1993.

[105] H. Narihisa, T. Taniguchi, M. Thuda, and K. Katayama. Efficiency of parallel exponential evolutionary programming. In *International Conference Workshops on Parallel Processing. ICPP 2005 Workshops.*, pages 588–595, 2005, doi:10.1109/ICPPW.2005.29.

[106] C.M. Neethling. Using SetPSO to determine RNA secondary structure. Master's thesis, University of Pretoria, 2008.

[107] M.G.H. Omran, A.P. Engelbrecht, and A. Salman. Barebones particle swarm for integer programming problems. In *IEEE Swarm Intelligence Symposium*, pages 170–175, 2007, doi:10.1109/SIS.2007.368042.

[108] M.G.H Omran, A.P. Engelbrecht, and A. Salman. Empirical Analysis of Self-Adaptive Differential Evolution. *European Journal of Operational Research (in press)*, 2007.

[109] D. Orvosh and L. Davis. Using a genetic algorithm to optimize problems with feasibility constraints. In *Proceedings of the First IEEE Conference on Evolutionary Computation.*, volume 2, pages 548–553, 1994, doi:10.1109/ICEC.1994.350001.

[110] G. Pamparà and A.P. Engelbrecht. Binary artificial bee colony optimization. In *IEEE Symposium on Swarm Intelligence*, pages 1–8, 2011, doi:10.1109/SIS.2011.5952562.

[111] G. Pamparà, A.P. Engelbrecht, and T. Cloete. CIlib: A collaborative framework for Computational Intelligence algorithms - Part I. In *Proceedings of the IEEE International Joint Conference on Neural Networks. IJCNN 2008. (IEEE World Congress on Computational Intelligence)*, pages 1750–1757, 2008, doi:10.1109/IJCNN.2008.4634035.

[112] G. Pamparà, A.P. Engelbrecht, and N. Franken. Binary differential evolution. In *Proceedings of the IEEE Congress on Evolutionary Computation.*, pages 1873–1879, 2006, doi:10.1109/CEC.2006.1688535.

[113] G. Pamparà, N. Franken, and A.P. Engelbrecht. Combining particle swarm optimisation with angle modulation to solve binary problems. In *Proceedings of the IEEE Congress on Evolutionary Computation.*, volume 1, pages 89–96, 2005, doi:10.1109/CEC.2005.1554671.

[114] T. Pankiw. The honey bee foraging behavior syndrome: quantifying the response threshold model of division of labor. In *Proceedings of the IEEE Swarm Intelligence Symposium. SIS 2005.*, pages 1–6, 2005, doi:10.1109/SIS.2005.1501595.

[115] I. Parberry. An efficient algorithm for the knight's tour problem. *Discrete Applied Mathematics*, 73(3):251–260, 1997, doi:10.1016/S0166-218X(96)00010-8.

[116] K.E. Parsopoulos, E.C. Laskari, and M.N. Vrahatis. Particle identification by light scattering through evolutionary algorithms. In *Proceedings of the First International Conference for Mathematics & Informatics for Industry (MII 2003)*, pages 97–108, 2003.

[117] R. Pasupath. On Choosing Parameters in Retrospective-Approximation Algorithms for Stochastic Root Finding. *Operational Research*, 58:889–901, 2010, doi:10.1287/opre.1090.0773.

[118] E.S. Peer. A Serendipitous Software Framework for Facilitating Collaboration in Computational Intelligence. Master's thesis, University of Pretoria, 2005.

[119] E.S. Peer, F. van den Bergh, and A.P. Engelbrecht. Using neighbourhoods with the guaranteed convergence pso. In *Proceedings of the IEEE Swarm Intelligence Symposium. SIS '03.*, pages 235–242, 2003, doi:10.1109/SIS.2003.1202274.

[120] C. Peng, L. Jian, and L. Zhiming. Solving 0-1 knapsack problems by a discrete binary version of differential evolution. In *Second International Symposium on Intelligent Information Technology Application.*, volume 2, pages 513–516, 2008, doi:10.1109/IITA.2008.538.

[121] M.R. Peterson, M.L. Raymer, and G.B. Lamont. Balanced accuracy for feature subset selection with genetic algorithms. In *IEEE Congress on Evolutionary Computation.*, volume 3, pages 2506–2513, 2005, doi:10.1109/CEC.2005.1555008.

[122] D. Pisinger. Where are all the hard knapsack problems? *Computers & Operations Research*, 32(9):2271–2284, 2005, doi:10.1016/j.cor.2004.03.002.

[123] W. Poundstone. *Prisoner's Dilemma.* Doubleday, 1993. Based On Work By-Neumann, John Von.

[124] K.V. Price. Differential evolution: a fast and simple numerical optimizer. In *Biennial Conference of the North American Fuzzy Information Processing Society. NAFIPS.*, pages 524–527, 1996, doi:10.1109/NAFIPS.1996.534790.

[125] J. Puchinger, G.R. Raidl, and U. Pferschy. The multidimensional knapsack problem: Structure and algorithms. Technical report, 2007.

[126] G.R. Raidl. An improved genetic algorithm for the multiconstrained 0-1 knapsack problem. In *Evolutionary Computation Proceedings of IEEE World Congress on Computational Intelligence.*, pages 207–211, 1998, doi:10.1109/ICEC.1998.699502.

[127] J. Reed, R. Toombs, and N.A. Barricelli. Simulation of biological evolution and machine learning : I. selection of self-reproducing numeric patterns by data processing machines, effects of hereditary control, mutation type and crossing. *Journal of Theoretical Biology*, 17(3):319–342, 1967, doi:10.1016/0022-5193(67)90097-5.

[128] C. Reis, J.A.T. Machado, A.M.S. Galhano, and J.B. Cunha. Circuit synthesis using particle swarm optimization. In *IEEE International Conference on Computational Cybernetics*, pages 1–6, 2006, doi:10.1109/ICCCYB.2006.305723.

[129] Z. Ren, M. Pham, W. Li, and C.S. Koh. A new global optimization algorithm for mixed-integer-discrete-continuous variables based on particles swarm optimization. In *Fourteenth Biennial IEEE Conference on Electromagnetic Field Computation.*, pages 1–10, 2010, doi:10.1109/CEFC.2010.5481403.

[130] AT&T Research. On-line encyclopedia of integer sequences: A006075. `http://www.research.att.com`. [Available On-line].

[131] C.W. Reynolds. Flocks, herds and schools: A distributed behavioral model. In *SIGGRAPH '87: Proceedings of the Fourteenth annual conference on Computer graphics and interactive techniques*, pages 25–34. ACM, 1987, doi:10.1145/37401.37406.

[132] The scientific consensus around evolution is overwhelming. Appendix: Frequently Asked Questions (php). Science and creationism: A view from the national academy of sciences (second ed.), 1999. Washington, DC: The National Academy of Sciences. 1999. p. 28. ISBN: ISBN-0-309-06406-6. Retrieved September 24, 2009.

[133] T.D. Seeley. *The Wisdom of the Hive: The Social Physiology of Honey Bee Colonies*. Harvard University Press, 1 edition, 1995.

[134] Y. Shang and Y. Qiu. A Note on the Extended Rosenbrock Function. *Evolutionary Computation*, 14:119–126, 2006.

[135] Y. Shao, H. Xu, and W. Yin. Solve zero-one knapsack problem by greedy genetic algorithm. In *International Workshop on Intelligent Systems and Applications*, pages 1–4, 2009, doi:10.1109/IWISA.2009.5073116.

[136] Y. Shi and R. Eberhart. A modified particle swarm optimizer. In *Proceedings of the IEEE International Congress on Computational Intelligence.*, pages 69–73, 1998, doi:10.1109/ICEC.1998.699146.

[137] Y. Shi and R. Eberhart. Parameter selection in particle swarm optimization. In V. Porto, N. Saravanan, D. Waagen, and A. Eiben, editors, *Evolutionary Programming VII*, volume 1447 of *Lecture Notes in Computer Science*, pages 591–600. Springer Berlin / Heidelberg, 1998, doi:10.1007/BFb0040810.

[138] W. Spears. C function code. Binary to continuous valued decision variable. `http://www.aic.nrl.navy.mil/~spears/functs.dejong.html`. [Online; accessed 18-May-2006].

[139] W. Spears. DeJong Test Functions From William M. Spears. `http://www.cs.uwyo.edu/~wspears/functs.dejong.html`, 1999.

[140] M.R. Spiegel. Theory and problems of probability and statistics. In *Schaum's Outline Series in Mathematics*, 1975.

[141] R. Storn. On the usage of differential evolution for function optimization. In *Biennial Conference of the North American Fuzzy Information Processing Society*, pages 519–523, 1996, doi:10.1109/NAFIPS.1996.534789.

[142] R. Storn and K. Price. Differential evolution - A Simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, International Computer Science Institute, 1995.

[143] R. Storn and K. Price. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11:341–359, 1997, doi:10.1023/A:1008202821328.

[144] R.A. Sturm and T.N. Frudakis. Eye colour: portals into pigmentation genes and ancestry. *Trends in Genetics*, 20(8):327–332, 2004, doi:10.1016/j.tig.2004.06.010.

[145] P.N. Suganthan, N. Hansen, J.J. Liang, K. Deb, Y.P. Chen, A. Auger, and S. Tiwari. Problem definitions and evaluation criteria for the CEC 2005 special session on real parameter optimization. Technical report, Nanyang Technological University, 2005.

[146] Y. Tekol and A. Acan. Ants can play prisoner's dilemma. In *The 2003 Congress on Evolutionary Computation.*, volume 2, pages 1348–1354, 2003, doi:10.1109/CEC.2003.1299825.

[147] D. Teodorović, P. Lučić, G. Markovic, and M. Dell' Orco. Bee colony optimization: Principles and applications. In *Eighth Seminar on Neural Network Applications in Electrical Engineering*, pages 151–156, 2006, doi:10.1109/NEUREL.2006.341200.

[148] G. Theraulaz and E. Bonabeau. Modeling the collective building of complex architectures in social insects with lattice swarms. *Journal of Theoretical Biology*, 1995.

[149] I.C. Trelea. The particle swarm optimization algorithm: convergence analysis and parameter selection. *Information Processing Letters*, 85(6):317–325, 2003, doi:10.1016/S0020-0190(02)00447-7.

[150] A.M. Turky and M.S. Ahmad. Using genetic algorithm for solving n-queens problem. In *International Symposium in Information Technology*, volume 2, pages 745–747, 2010, doi:10.1109/ITSIM.2010.5561604.

[151] Computational Intelligence Research Group @ UP. CIlib: Computational Intelligence library. `http://www.cilib.net/`, 2003–2009.

[152] F. van den Bergh. *An Analysis of Particle Swarm Optimizers.* PhD thesis, Department of Computer Science, University of Pretoria, South Africa, 2002.

[153] F. van den Bergh and A.P. Engelbrecht. Effects of swarm size on cooperative particle swarm optimisers. In *Proceedings of GECCO 2001*, 2001.

[154] F. van den Bergh and A.P. Engelbrecht. A study of particle swarm optimization particle trajectories. *Information Sciences*, 176(8):937–971, 2006, doi:10.1016/j.ins.2005.02.003.

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

[155] H. Voigt and H. Muhlenbein. Gene pool recombination and utilization of covariances for the breeder genetic algorithm. In *IEEE International Conference on Evolutionary Computation.*, volume 1, page 172, 1995, doi:10.1109/ICEC.1995.489139.

[156] J. Wang and Y. Zhang. Using evolutionary algorithm based on hybrid probability distribution to solve tsp with area partition. In *International Conference on Environmental Science and Information Application Technology (ESIAT).*, volume 4, pages 337–340, 2010, doi:10.1109/ESIAT.2010.5567406.

[157] D. Whitley, V.S. Gordon, and K. Mathias. Lamarckian evolution, the Baldwin effect and function optimization. In *Parallel Problem Solving from Nature*, volume 866 of *Lecture Notes in Computer Science*, pages 5–15. Springer Berlin / Heidelberg, 1994, doi:10.1007/3-540-58484-6_245.

[158] A.H. Wright. Genetic algorithms for real parameter optimization. In *Foundations of Genetic Algorithms*, pages 205–218. Morgan Kaufmann, 1991.

[159] X. Yao, Y. Liu, and G. Lin. Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation*, 3(2):82–102, 1999, doi:10.1109/4235.771163.

[160] S. Yoshi, K. Suzuki, and Y. Kakazu. Lamarckian ga with genetic supervision. In *Proceedings of the IEEE International Conference on Evolutionary Computation*, volume 1, page 367, 1995, doi:10.1109/ICEC.1995.489175.

[161] G. Zhang and Y. Wei. An improved particle swarm optimization algorithm for solving 0-1 knapsack problem. In *International Conference on Machine Learning and Cybernetics*, volume 2, pages 915–918, 2008, doi:10.1109/ICMLC.2008.4620535.

[162] J. Zhao, T. Huang, F. Pang, and Y. Liu. Genetic algorithm based on greedy strategy in the 0-1 knapsack problem. In *Third International Conference on Genetic and Evolutionary Computing.*, pages 105–107, 2009, doi:10.1109/WGEC.2009.43.

[163] P. Zhao, P. Zhao, and X. Zhang. A new ant colony optimization for the knapsack problem. In *Seventh International Conference on Computer-Aided Industrial Design and Conceptual Design*, pages 1–3, 2006, doi:10.1109/CAIDCD.2006.329439.

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

[164] X.H. Zhi, X.L. Xing, Q.X. Wang, L.H. Zhang, X.W. Yang, C.G. Zhou, and Y.C. Liang. A discrete pso method for generalized tsp problem. In *Proceedings of International Conference on Machine Learning and Cybernetics.*, volume 4, pages 2378–2383, 2004, doi:10.1109/ICMLC.2004.1382200.

Appendix A

List of symbols

c	Capacity of knapsack.
f	Objective function.
IPD_b	Personal best IPD.
IPD_c	Collective IPD.
KC_4	Knights cover of 4×4 chessboard.
KC_5	Knights cover of 5×5 chessboard.
KC_6	Knights cover of 6×6 chessboard.
KC_7	Knights cover of 7×7 chessboard.
KC_8	Knights cover of 8×8 chessboard.
KT_4	Knights tour on 4×4 chessboard.
KT_5	Knights tour on 5×5 chessboard.
KT_6	Knights tour on 6×6 chessboard.
KT_7	Knights tour on 7×7 chessboard.
KT_8	Knights tour on 8×8 chessboard.
MDK_{10}	Multi-dimensional Knapsack problem of dimension 10.

MDK_{20}	Multi-dimensional Knapsack problem of dimension 20.
$n_{\mathcal{N}_i}$	Neighborhood size for the neighborhood $\mathcal{N}_i$.
n_e	Number of employed bees.
n_o	Number of unemployed bees.
n_s	Size of population.
n_v	Number of difference vectors.
n_x	Length of individual.
n_t	Size of tournament.
P	Population within a EA.
p_c	Probability of crossover.
p_i	Profit value associated with object i.
p_m	probability of mutation.
Q_4	n-Queens problem on 4×4 chessboard.
Q_5	n-Queens problem on 5×5 chessboard.
Q_6	n-Queens problem on 6×6 chessboard.
Q_7	n-Queens problem on 7×7 chessboard.
Q_8	n-Queens problem on 8×8 chessboard.
$U(0,1)$	Uniform random number between 0 and 1.
w_i	Weight value associated with object i.
$\widetilde{x}$	Mutated offspring individual.

Appendix B

List of acronyms

ABC	artificial bee colony.
ACO	ant colony optimization.
AM	angle modulation.
AMABC	angle modulated artificial bee colony.
AMDE	angle modulated differential evolution.
AMEP	angle modulated evolutionary programming.
AMGA	angle modulated genetic algorithm.
AMPSO	angle modulated particle swarm optimization.
BinABC	binary artificial bee colony.
BinDE	binary differential evolution.
BinEP	binary evolutionary programming.
BinGA	binary genetic algorithm.
BinPSO	binary particle swarm optimization.
bit	binary digit.
BLX-α	blend crossover.
CEP	classic evolutionary programming.
CGA	canonical genetic algorithm.
CIlib	computational intelligence library.

DE	differential evolution.
EA	evolutionary algorithm.
EC	evolutionary computation.
EP	evolutionary programming.
ES	evolutionary strategies.
FEP	fast evolutionary programming.
FM	frequency modulation.
FR	fuzzy recombination.
FSM	finite state machine.
GA	genetic algorithm.
IPD	iterated prisoner's dilemma.
KC	knight's coverage.
KT	knight's tour.
LSB	least significant bit.
MDK	multidimensional knapsack problem.
MSB	most significant bit.
NormABC	normalization artificial bee colony.
NormDE	normalization differential evolution.
PBA	population-based algorithm.
PD	prisoner's dilemma.
PSO	particle swarm optimization.
Q	n-Queens.

SBX	simulated binary crossover.
SI	swarm intelligence.
TSP	traveling salesman problem.

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Appendix C

Derived Publications

This appendix lists the publications which have been derived from this thesis:

- G. Pamparà, N. Franken, and A.P. Engelbrecht. Combining particle swarm optimisation with angle modulation to solve binary problems. In *Proceedings of the IEEE Congress on Evolutionary Computation.*, volume 1, pages 89–96, 2005, doi:10.1109/CEC.2005.1554671

- G. Pamparà, A.P. Engelbrecht, and N. Franken. Binary differential evolution. In *Proceedings of the IEEE Congress on Evolutionary Computation.*, pages 1873–1879, 2006, doi:10.1109/CEC.2006.1688535

- A.P. Engelbrecht and G. Pamparà. Binary differential evolution strategies. In *Proceedings of the IEEE Congress on Evolutionary Computation.*, pages 1942–1947, 2007, doi:10.1109/CEC.2007.4424711

- G. Pamparà and A.P. Engelbrecht. Binary artificial bee colony optimization. In *IEEE Symposium on Swarm Intelligence*, pages 1–8, 2011, doi:10.1109/SIS.2011.5952562

Appendix D

Binary encoding of continuous values

Kennedy and Eberhart [83] proposed a set of binary benchmark functions to test the accuracy of the BinPSO. The benchmarks were continuous-valued optimization problems, with binary strings used as the candidate solution representation. A conversion process to convert the binary candidate solution into a continuous-valued solution was required. Spears [138] defined a conversion process between a binary string and continuous-valued space, and this process was used by Kenedy and Eberhart to convert a binary particle position into a continuous-valued one. This thesis also uses the encoding of Spears.

The encoding process, as suggested by Spears, caters for a defined level of precision, requiring larger bit string representations. The conversion process sub-divides the binary candidate solution into smaller bit strings which represent the value for a continuous-valued decision variable. The bit string is converted directly into a continuous value, but without a fractional part. The fraction is determined by shifting the position of the decimal point. The shift of decimal point requires division of the continuous value by a factor of 10^p, where p defines the number of decimals to maintain. Equation (D.1) calculates the continuous-value from a given bit string:

$$f_{decode}(x_b) = \frac{B(x) - x_{min}}{10^p} \tag{D.1}$$

where x_b is a binary string, B is a function that converts a binary string into a continuous value by evaluating bit values, x_{min} is the search space lower bound and p is a natural number representing the number of decimals to maintain.

The number of bits required to represent a continuous-valued decision variable x in binary, with a given precision, is calculated as:

$$f_{bits}(x) = \lceil \log_2((|x_{min}| + |x_{max}|) \times 10^p) \rceil \tag{D.2}$$

Appendix E

Simulation specifications

The specification files for all simulations run to produce the results for this thesis are presented within this appendix. Simulation files are separated, based on the algorithms that effectively ran the simulation experiments and the benchmark function suite. All simulation specifications are provided in the sections that follow.

Section E.6 lists additional Java sources have been listed. These source files are required in addition to the main CIlib source tree for the provided simulation specifications. The simulation specifications require CIlib, version 0.7.3.1 [27, 111]. For convenience, the electronic version of this thesis embeds the additional Java sources and simulation specifications.

E.1 Genetic algorithm

E.1.1 Continuous simulations

Angle modulated genetic algorithm

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE simulator [
 <!ATTLIST algorithm id ID #IMPLIED>
<!ATTLIST problem id ID #IMPLIED>
<!ATTLIST measurements id ID #IMPLIED>
]>
<simulator>
  <algorithms>
    <algorithm id="ga" class="ec.EC">
      <iterationStrategy class="ec.iterationstrategies.GeneticAlgorithmIterationStrategy">
        <mutationStrategy class="entity.operators.mutation.GaussianMutationStrategy">
    <mutationProbability class="controlparameter.ConstantControlParameter" parameter="0.7"/>
  </mutationStrategy>
  <boundaryConstraint class="problem.boundaryconstraint.ClampingBoundaryConstraint"/>
      </iterationStrategy>
      <addStoppingCondition class="stoppingcondition.MaximumIterations" maximumIterations="10000"/>
    </algorithm>
  </algorithms>
  <problems>
    <problem id="spherical" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2" domain="R
           (-1.0,1.0)^4">
        <function class="functions.continuous.unconstrained.Spherical" domain="R(-5.12,5.12)^30"/>
      </function>
    </problem>
    <problem id="rosenbrock2d" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="3" domain="R
           (-1.0,1.0)^4">
        <function class="functions.continuous.unconstrained.Rosenbrock" domain="R(-2.048,2.048)^2"/>
      </function>
    </problem>
    <problem id="step" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2" domain="R
           (-1.0,1.0)^4">
        <function class="functions.continuous.Step" domain="R(-5.12,5.12)^5"/>
      </function>
    </problem>
    <problem id="quartic" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2" domain="R
           (-1.0,1.0)^4">
        <function class="functions.continuous.Quartic" domain="R(-1.28,1.28)^30"/>
      </function>
    </problem>
    <problem id="foxholes" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2" domain="R
           (-1.0,1.0)^4">
        <function class="functions.continuous.Foxholes" domain="R(-65536.0,65536.0)^2"/>
      </function>
    </problem>
    <problem id="schaffer6" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2" domain="R
           (-1.0,1.0)^4">
        <function class="functions.continuous.Schaffer6" domain="R(-100.0,100.0)^2"/>
      </function>
    </problem>
    <problem id="griewank" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2" domain="R
           (-1.0,1.0)^4">
        <function class="functions.continuous.unconstrained.Griewank" domain="R(-300.0,300.0)^30"/>
      </function>
    </problem>
    <problem id="ackley" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2" domain="R
           (-1.0,1.0)^4">
        <function class="functions.continuous.unconstrained.Ackley" domain="R(-30.0,30.0)^30"/>
      </function>
    </problem>
    <problem id="rosenbrock" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="3" domain="R
           (-1.0,1.0)^4">
        <function class="functions.continuous.unconstrained.Rosenbrock" domain="R(-2.048,2.048)^30"/>
      </function>
    </problem>
    <problem id="rastrigin" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2" domain="R
           (-1.0,1.0)^4">
        <function class="functions.continuous.unconstrained.Rastrigin" domain="R(-5.12,5.12)^30"/>
      </function>
    </problem>
  </problems>
  <measurements id="measurements" class="simulator.MeasurementSuite" samples="30" resolution="1">
    <addMeasurement class="measurement.single.Fitness"/>
  </measurements>
  <simulations>
    <simulation>
      <algorithm idref="ga"/>
      <problem idref="spherical"/>
      <measurements idref="measurements" file="data/masters/ga/amga-spherical.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="ga"/>
      <problem idref="rosenbrock2d"/>
      <measurements idref="measurements" file="data/masters/ga/amga-rosenbrock2d.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="ga"/>
      <problem idref="step"/>
      <measurements idref="measurements" file="data/masters/ga/amga-step.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="ga"/>
      <problem idref="quartic"/>
      <measurements idref="measurements" file="data/masters/ga/amga-quartic.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="ga"/>
      <problem idref="foxholes"/>
      <measurements idref="measurements" file="data/masters/ga/amga-foxholes.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="ga"/>
      <problem idref="schaffer6"/>
      <measurements idref="measurements" file="data/masters/ga/amga-schaffer6.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="ga"/>
      <problem idref="griewank"/>
      <measurements idref="measurements" file="data/masters/ga/amga-griewank.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="ga"/>
      <problem idref="ackley"/>
      <measurements idref="measurements" file="data/masters/ga/amga-ackley.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="ga"/>
      <problem idref="rosenbrock"/>
      <measurements idref="measurements" file="data/masters/ga/amga-rosenbrock.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="ga"/>
      <problem idref="rastrigin"/>
      <measurements idref="measurements" file="data/masters/ga/amga-rastrigin.txt"/>
    </simulation>
  </simulations>
</simulator>
```

Binary genetic algorithm

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE simulator [
<!ATTLIST algorithm id ID #IMPLIED>
```

```xml
<!ATTLIST problem id ID #IMPLIED>
<!ATTLIST measurements id ID #IMPLIED>
]>
<simulator>
  <algorithms>
    <algorithm id="binga" class="ec.EC">
      <initialisationStrategy class="algorithm.initialisation.ClonedPopulationInitialisationStrategy" entityNumber="40">
        <entityType class="ec.Individual" />
      </initialisationStrategy>
      <iterationStrategy class="ec.iterationstrategies.GeneticAlgorithmIterationStrategy">
  <boundaryConstraint class="problem.boundaryconstraint.ClampingBoundaryConstraint"/>
  <mutationStrategy class="entity.operators.mutation.RandomMutationStrategy">
    <mutationProbability
     class="controlparameter.ConstantControlParameter"
     parameter="0.7"/>
  </mutationStrategy>
      </iterationStrategy>
      <addStoppingCondition class="stoppingcondition.MaximumIterations" maximumIterations="10000"/>
    </algorithm>
  </algorithms>
  <problems>
    <problem id="spherical" class="problem.FunctionMinimisationProblem">
      <function class="functions.discrete.BinaryAdapter" precision="2" bitsPerDimension="10" domain="B^300">
        <function class="functions.continuous.unconstrained.Spherical" domain="R(-5.12,5.12)^30"/>
      </function>
    </problem>
    <problem id="rosenbrock2d" class="problem.FunctionMinimisationProblem">
      <function class="functions.discrete.BinaryAdapter" precision="3" bitsPerDimension="12" domain="B^24">
        <function class="functions.continuous.unconstrained.Rosenbrock" domain="R(-2.048,2.048)^2"/>
      </function>
    </problem>
    <problem id="step" class="problem.FunctionMinimisationProblem">
      <function class="functions.discrete.BinaryAdapter" precision="2" bitsPerDimension="10" domain="B^50">
        <function class="functions.continuous.Step" domain="R(-5.12,5.12)^5"/>
      </function>
    </problem>
    <problem id="quartic" class="problem.FunctionMinimisationProblem">
      <function class="functions.discrete.BinaryAdapter" precision="2" bitsPerDimension="8" domain="B^240">
        <function class="functions.continuous.Quartic" domain="R(-1.28,1.28)^30"/>
      </function>
    </problem>
    <problem id="foxholes" class="problem.FunctionMinimisationProblem">
      <function class="functions.discrete.BinaryAdapter" precision="0" bitsPerDimension="17" domain="B^34">
        <function class="functions.continuous.Foxholes" domain="R(-65536.0,65536.0)^2"/>
      </function>
    </problem>
    <problem id="schaffer6" class="problem.FunctionMinimisationProblem">
      <function class="functions.discrete.BinaryAdapter" precision="0" bitsPerDimension="8" domain="B^16">
        <function class="functions.continuous.Schaffer6" domain="R(-100.0,100.0)^2"/>
      </function>
    </problem>
   <problem id="griewank" class="problem.FunctionMinimisationProblem">
      <function class="functions.discrete.BinaryAdapter" precision="0" bitsPerDimension="10" domain="B^300">
        <function class="functions.continuous.unconstrained.Griewank" domain="R(-300.0,300.0)^30"/>
      </function>
    </problem>
    <problem id="ackley" class="problem.FunctionMinimisationProblem">
      <function class="functions.discrete.BinaryAdapter" precision="0" bitsPerDimension="6" domain="B^180">
```

```
        <function class="functions.continuous.unconstrained.Ackley" domain="R(-30.0,30.0)^30"/>
      </function>
    </problem>
    <problem id="rosenbrock" class="problem.FunctionMinimisationProblem">
      <function class="functions.discrete.BinaryAdapter" precision="3" bitsPerDimension="12" domain="B
          ^360">
        <function class="functions.continuous.unconstrained.Rosenbrock" domain="R(-2.048,2.048)^30"/>
      </function>
    </problem>
    <problem id="rastrigin" class="problem.FunctionMinimisationProblem">
      <function class="functions.discrete.BinaryAdapter" precision="2" bitsPerDimension="10" domain="B
          ^300">
        <function class="functions.continuous.unconstrained.Rastrigin" domain="R(-5.12,5.12)^30"/>
      </function>
    </problem>
  </problems>
  <measurements id="measurements" class="simulator.MeasurementSuite" samples="1" resolution="1">
    <addMeasurement class="measurement.single.Fitness"/>
  </measurements>
  <simulations>
  <simulation>
      <algorithm idref="binga"/>
      <problem idref="spherical"/>
      <measurements idref="measurements" file="data/masters/ga/binga/binga-spherical.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binga"/>
      <problem idref="rosenbrock2d"/>
      <measurements idref="measurements" file="data/masters/ga/binga/binga-rosenbrock2d.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binga"/>
      <problem idref="step"/>
      <measurements idref="measurements" file="data/masters/ga/binga/binga-step.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binga"/>
      <problem idref="quartic"/>
      <measurements idref="measurements" file="data/masters/ga/binga/binga-quartic.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binga"/>
      <problem idref="foxholes"/>
      <measurements idref="measurements" file="data/masters/ga/binga/binga-foxholes.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binga"/>
      <problem idref="schaffer6"/>
      <measurements idref="measurements" file="data/masters/ga/binga/binga-schaffer6.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binga"/>
      <problem idref="griewank"/>
      <measurements idref="measurements" file="data/masters/ga/binga/binga-griewank.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binga"/>
      <problem idref="ackley"/>
      <measurements idref="measurements" file="data/masters/ga/binga/binga-ackley.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binga"/>
      <problem idref="rosenbrock"/>
      <measurements idref="measurements" file="data/masters/ga/binga/binga-rosenbrock.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binga"/>
```

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

```
      <problem idref="rastrigin"/>
      <measurements idref="measurements" file="data/masters/ga/binga/binga-rastrigin.txt"/>
    </simulation>
  </simulations>
</simulator>
```

E.1.2 Binary simulations

Angle modulated genetic algorithm

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE simulator [
<!ATTLIST algorithm id ID #IMPLIED>
<!ATTLIST problem id ID #IMPLIED>
<!ATTLIST measurements id ID #IMPLIED>
]>
<simulator>
  <algorithms>
    <algorithm id="ga" class="ec.EC">
      <iterationStrategy class="ec.iterationstrategies.GeneticAlgorithmIterationStrategy">
        <mutationStrategy class="entity.operators.mutation.GaussianMutationStrategy">
    <mutationProbability class="controlparameter.ConstantControlParameter" parameter="0.7"/>
  </mutationStrategy>
  <boundaryConstraint class="problem.boundaryconstraint.ClampingBoundaryConstraint"/>
      </iterationStrategy>
      <addStoppingCondition class="stoppingcondition.MaximumIterations" maximumIterations="10000"/>
    </algorithm>
  </algorithms>
  <problems>
      <problem id="mdk-10" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.KnapSack" domain="B^10">
    <capacity value="269" />
    <numberOfObjects value="10" />
    <weight value="95,4,60,32,23,72,80,62,64,46" />
    <value value="55,10,47,5,4,50,8,61,85,87" />
  </function>
      </function>
    </problem>
    <problem id="mdk-20" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.KnapSack" domain="B^20">
    <capacity value="878" />
    <numberOfObjects value="20" />
    <weight value="92,4,43,83,84,68,92,82,6,44,32,18,56,83,25,96,70,48,14,58" />
    <value value="44,46,90,72,91,40,75,35,8,54,78,40,77,15,61,17,75,29,75,63" />
  </function>
      </function>
    </problem>
    <problem id="kc-4" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.KnightsCoverage" boardSize="4" />
      </function>
    </problem>
    <problem id="kc-5" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.KnightsCoverage" boardSize="5" />
      </function>
    </problem>
    <problem id="kc-6" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.KnightsCoverage" boardSize="6" />
      </function>
    </problem>
    <problem id="kc-7" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.KnightsCoverage" boardSize="7" />
      </function>
    </problem>
    <problem id="kc-8" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.KnightsCoverage" boardSize="8" />
      </function>
    </problem>
    <problem id="q-4" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.Queens" boardSize="4" />
      </function>
    </problem>
    <problem id="q-5" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.Queens" boardSize="5" />
      </function>
    </problem>
    <problem id="q-6" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.Queens" boardSize="6" />
      </function>
    </problem>
    <problem id="q-7" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.Queens" boardSize="7" />
      </function>
    </problem>
    <problem id="q-8" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.Queens" boardSize="8" />
      </function>
    </problem>
    <problem id="kt-4" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.RepairingKnightsTour" boardSize="4" />
      </function>
    </problem>
    <problem id="kt-5" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.RepairingKnightsTour" boardSize="5" />
      </function>
    </problem>
    <problem id="kt-6" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.RepairingKnightsTour" boardSize="6" />
      </function>
    </problem>
    <problem id="kt-7" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.RepairingKnightsTour" boardSize="7" />
      </function>
    </problem>
    <problem id="kt-8" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.RepairingKnightsTour" boardSize="8" />
      </function>
    </problem>
    <problem id="ipd" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.IteratedPrisonersDilemma" domain="B^64" />
      </function>
    </problem>
```

```
  </problems>
  <measurements id="measurements" class="simulator.MeasurementSuite" samples="30" resolution="1">
    <addMeasurement class="measurement.single.Fitness"/>
    <addMeasurement class="measurement.single.CollectiveFitness" />
   <!--<addMeasurement class="measurement.single.Solution" />-->
  </measurements>
  <simulations>
    <simulation>
      <algorithm idref="ga"/>
      <problem idref="mdk-10"/>
      <measurements idref="measurements" file="data/masters/binary/ga/amga/amga-mdk-10.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="ga"/>
      <problem idref="mdk-20"/>
      <measurements idref="measurements" file="data/masters/binary/ga/amga/amga-mdk-20.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="ga"/>
      <problem idref="kc-4"/>
      <measurements idref="measurements" file="data/masters/binary/ga/amga/amga-kc-4.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="ga"/>
      <problem idref="kc-5"/>
      <measurements idref="measurements" file="data/masters/binary/ga/amga/amga-kc-5.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="ga"/>
      <problem idref="kc-6"/>
      <measurements idref="measurements" file="data/masters/binary/ga/amga/amga-kc-6.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="ga"/>
      <problem idref="kc-7"/>
      <measurements idref="measurements" file="data/masters/binary/ga/amga/amga-kc-7.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="ga"/>
      <problem idref="kc-8"/>
      <measurements idref="measurements" file="data/masters/binary/ga/amga/amga-kc-8.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="ga"/>
      <problem idref="q-4"/>
      <measurements idref="measurements" file="data/masters/binary/ga/amga/amga-q-4.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="ga"/>
      <problem idref="q-5"/>
      <measurements idref="measurements" file="data/masters/binary/ga/amga/amga-q-5.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="ga"/>
      <problem idref="q-6"/>
      <measurements idref="measurements" file="data/masters/binary/ga/amga/amga-q-6.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="ga"/>
      <problem idref="q-7"/>
      <measurements idref="measurements" file="data/masters/binary/ga/amga/amga-q-7.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="ga"/>
      <problem idref="q-8"/>
      <measurements idref="measurements" file="data/masters/binary/ga/amga/amga-q-8.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="ga" />
      <problem idref="kt-4" />
      <measurements idref="measurements" file="data/masters/binary/ga/amga/amga-kt-4.txt" />
    </simulation>
    <simulation>
      <algorithm idref="ga" />
      <problem idref="kt-5" />
      <measurements idref="measurements" file="data/masters/binary/ga/amga/amga-kt-5.txt" />
    </simulation>
    <simulation>
      <algorithm idref="ga" />
      <problem idref="kt-6" />
      <measurements idref="measurements" file="data/masters/binary/ga/amga/amga-kt-6.txt" />
    </simulation>
    <simulation>
      <algorithm idref="ga" />
      <problem idref="kt-7" />
      <measurements idref="measurements" file="data/masters/binary/ga/amga/amga-kt-7.txt" />
    </simulation>
    <simulation>
      <algorithm idref="ga" />
      <problem idref="kt-8" />
      <measurements idref="measurements" file="data/masters/binary/ga/amga/amga-kt-8.txt" />
    </simulation>
    <simulation>
      <algorithm idref="ga" />
      <problem idref="ipd" />
      <measurements idref="measurements" file="data/masters/binary/ga/amga/amga-ipd.txt" />
    </simulation>
  </simulations>
</simulator>
```

Binary genetic algorithm

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE simulator [
 <!ATTLIST algorithm id ID #IMPLIED>
<!ATTLIST problem id ID #IMPLIED>
<!ATTLIST measurements id ID #IMPLIED>
]>
<simulator>
  <algorithms>
    <algorithm id="binga" class="ec.EC">
      <initialisationStrategy class="algorithm.initialisation.ClonedPopulationInitialisationStrategy"
           entityNumber="40">
        <entityType class="ec.Individual" />
      </initialisationStrategy>
      <iterationStrategy class="ec.iterationstrategies.GeneticAlgorithmIterationStrategy">
  <boundaryConstraint class="problem.boundaryconstraint.ClampingBoundaryConstraint"/>
  <mutationStrategy class="entity.operators.mutation.RandomMutationStrategy">
    <mutationProbability
     class="controlparameter.ConstantControlParameter"
     parameter="0.7"/>
  </mutationStrategy>
      </iterationStrategy>
      <addStoppingCondition class="stoppingcondition.MaximumIterations" maximumIterations="10000"/>
    </algorithm>
  </algorithms>
  <problems>
      <problem id="mdk-10" class="problem.FunctionMaximisationProblem">
        <function class="functions.discrete.KnapSack" domain="B^10">
    <capacity value="269" />
    <numberOfObjects value="10" />
    <weight value="95,4,60,32,23,72,80,62,64,46" />
    <value value="55,10,47,5,4,50,8,61,85,87" />
  </function>
    </problem>
    <problem id="mdk-20" class="problem.FunctionMaximisationProblem">
        <function class="functions.discrete.KnapSack" domain="B^20">
    <capacity value="878" />
    <numberOfObjects value="20" />
    <weight value="92,4,43,83,84,68,92,82,6,44,32,18,56,83,25,96,70,48,14,58" />
    <value value="44,46,90,72,91,40,75,35,8,54,78,40,77,15,61,17,75,29,75,63" />
  </function>
    </problem>
    <problem id="kc-4" class="problem.FunctionMinimisationProblem">
        <function class="functions.discrete.KnightsCoverage" boardSize="4" />
    </problem>
    <problem id="kc-5" class="problem.FunctionMinimisationProblem">
        <function class="functions.discrete.KnightsCoverage" boardSize="5" />
    </problem>
    <problem id="kc-6" class="problem.FunctionMinimisationProblem">
        <function class="functions.discrete.KnightsCoverage" boardSize="6" />
    </problem>
    <problem id="kc-7" class="problem.FunctionMinimisationProblem">
        <function class="functions.discrete.KnightsCoverage" boardSize="7" />
    </problem>
    <problem id="kc-8" class="problem.FunctionMinimisationProblem">
        <function class="functions.discrete.KnightsCoverage" boardSize="8" />
    </problem>
    <problem id="q-4" class="problem.FunctionMinimisationProblem">
        <function class="functions.discrete.Queens" boardSize="4" />
    </problem>
    <problem id="q-5" class="problem.FunctionMinimisationProblem">
        <function class="functions.discrete.Queens" boardSize="5" />
    </problem>
    <problem id="q-6" class="problem.FunctionMinimisationProblem">
        <function class="functions.discrete.Queens" boardSize="6" />
    </problem>
    <problem id="q-7" class="problem.FunctionMinimisationProblem">
        <function class="functions.discrete.Queens" boardSize="7" />
    </problem>
    <problem id="q-8" class="problem.FunctionMinimisationProblem">
        <function class="functions.discrete.Queens" boardSize="8" />
    </problem>
    <problem id="kt-4" class="problem.FunctionMaximisationProblem">
        <function class="functions.discrete.RepairingKnightsTour" boardSize="4" />
    </problem>
    <problem id="kt-5" class="problem.FunctionMaximisationProblem">
        <function class="functions.discrete.RepairingKnightsTour" boardSize="5" />
    </problem>
    <problem id="kt-6" class="problem.FunctionMaximisationProblem">
        <function class="functions.discrete.RepairingKnightsTour" boardSize="6" />
    </problem>
    <problem id="kt-7" class="problem.FunctionMaximisationProblem">
        <function class="functions.discrete.RepairingKnightsTour" boardSize="7" />
    </problem>
    <problem id="kt-8" class="problem.FunctionMaximisationProblem">
        <function class="functions.discrete.RepairingKnightsTour" boardSize="8" />
    </problem>
    <problem id="ipd" class="problem.FunctionMaximisationProblem">
        <function class="functions.discrete.IteratedPrisonersDilemma" domain="B^64" />
    </problem>
  </problems>
  <measurements id="measurements" class="simulator.MeasurementSuite" samples="30" resolution="1">
    <addMeasurement class="measurement.single.Fitness"/>
    <addMeasurement class="measurement.single.CollectiveFitness" />
  </measurements>
  <simulations>
```

```
    <simulation>
      <algorithm idref="binga"/>
      <problem idref="mdk-10"/>
      <measurements idref="measurements" file="data/masters/binary/ga/binga/binga-mdk-10.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binga"/>
      <problem idref="mdk-20"/>
      <measurements idref="measurements" file="data/masters/binary/ga/binga/binga-mdk-20.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binga"/>
      <problem idref="kc-4"/>
      <measurements idref="measurements" file="data/masters/binary/ga/binga/binga-kc-4.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binga"/>
      <problem idref="kc-5"/>
      <measurements idref="measurements" file="data/masters/binary/ga/binga/binga-kc-5.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binga"/>
      <problem idref="kc-6"/>
      <measurements idref="measurements" file="data/masters/binary/ga/binga/binga-kc-6.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binga"/>
      <problem idref="kc-7"/>
      <measurements idref="measurements" file="data/masters/binary/ga/binga/binga-kc-7.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binga"/>
      <problem idref="kc-8"/>
      <measurements idref="measurements" file="data/masters/binary/ga/binga/binga-kc-8.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binga"/>
      <problem idref="q-4"/>
      <measurements idref="measurements" file="data/masters/binary/ga/binga/binga-q-4.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binga"/>
      <problem idref="q-5"/>
      <measurements idref="measurements" file="data/masters/binary/ga/binga/binga-q-5.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binga"/>
      <problem idref="q-6"/>
      <measurements idref="measurements" file="data/masters/binary/ga/binga/binga-q-6.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binga"/>
      <problem idref="q-7"/>
      <measurements idref="measurements" file="data/masters/binary/ga/binga/binga-q-7.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binga"/>
      <problem idref="q-8"/>
      <measurements idref="measurements" file="data/masters/binary/ga/binga/binga-q-8.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binga" />
      <problem idref="kt-4" />
      <measurements idref="measurements" file="data/masters/binary/ga/binga/binga-kt-4.txt" />
    </simulation>
        <simulation>
      <algorithm idref="binga" />
      <problem idref="kt-5" />
      <measurements idref="measurements" file="data/masters/binary/ga/binga/binga-kt-5.txt" />
    </simulation>
        <simulation>
      <algorithm idref="binga" />
      <problem idref="kt-6" />
      <measurements idref="measurements" file="data/masters/binary/ga/binga/binga-kt-6.txt" />
    </simulation>
        <simulation>
      <algorithm idref="binga" />
      <problem idref="kt-7" />
      <measurements idref="measurements" file="data/masters/binary/ga/binga/binga-kt-7.txt" />
    </simulation>
        <simulation>
      <algorithm idref="binga" />
      <problem idref="kt-8" />
      <measurements idref="measurements" file="data/masters/binary/ga/binga/binga-kt-8.txt" />
    </simulation>
    <simulation>
      <algorithm idref="binga" />
      <problem idref="ipd" />
      <measurements idref="measurements" file="data/masters/binary/ga/binga/binga-ipd.txt" />
    </simulation>
  </simulations>
</simulator>
```

E.2 Evolutionary programming

E.2.1 Continuous simulations

Angle modulated evolutionary programming

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE simulator [
 <!ATTLIST algorithm id ID #IMPLIED>
<!ATTLIST problem id ID #IMPLIED>
<!ATTLIST measurements id ID #IMPLIED>
]>
<simulator>
  <algorithms>
    <algorithm id="ep" class="ec.EC">
      <iterationStrategy class="ec.iterationstrategies.EvolutionaryProgrammingIterationStrategy">
        <mutationStrategy class="entity.operators.mutation.SelfAdaptiveMutationStrategy"/>
      </iterationStrategy>
      <initialisationStrategy class="algorithm.initialisation.ClonedPopulationInitialisationStrategy"
           entityNumber="40">
        <entityType class="ec.Individual"/>
      </initialisationStrategy>
      <strategyParameterInitialization class="entity.initialization.ConstantInitializationStrategy"
           constant="3.0"/>
      <addStoppingCondition class="stoppingcondition.MaximumIterations" maximumIterations="10000"/>
    </algorithm>
  </algorithms>
  <problems>
    <problem id="spherical" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2" domain="R
           (-1.0,1.0)^4">
        <function class="functions.continuous.unconstrained.Spherical" domain="R(-5.12,5.12)^30"/>
      </function>
    </problem>
    <problem id="rosenbrock2d" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="3" domain="R(-1.0,1.0)^4">
        <function class="functions.continuous.unconstrained.Rosenbrock" domain="R(-2.048,2.048)^2"/>
      </function>
    </problem>
    <problem id="step" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2" domain="R(-1.0,1.0)^4">
        <function class="functions.continuous.Step" domain="R(-5.12,5.12)^5"/>
      </function>
    </problem>
    <problem id="quartic" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2" domain="R(-1.0,1.0)^4">
        <function class="functions.continuous.Quartic" domain="R(-1.28,1.28)^30"/>
      </function>
    </problem>
    <problem id="foxholes" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2" domain="R(-1.0,1.0)^4">
        <function class="functions.continuous.Foxholes" domain="R(-65536.0,65536.0)^2"/>
      </function>
    </problem>
    <problem id="schaffer6" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2" domain="R(-1.0,1.0)^4">
        <function class="functions.continuous.Schaffer6" domain="R(-100.0,100.0)^2"/>
      </function>
    </problem>
   <problem id="griewank" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2" domain="R(-1.0,1.0)^4">
        <function class="functions.continuous.unconstrained.Griewank" domain="R(-300.0,300.0)^30"/>
      </function>
    </problem>
    <problem id="ackley" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2" domain="R(-1.0,1.0)^4">
        <function class="functions.continuous.unconstrained.Ackley" domain="R(-30.0,30.0)^30"/>
      </function>
    </problem>
    <problem id="rosenbrock" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="3" domain="R(-1.0,1.0)^4">
        <function class="functions.continuous.unconstrained.Rosenbrock" domain="R(-2.048,2.048)^30"/>
      </function>
    </problem>
    <problem id="rastrigin" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2" domain="R(-1.0,1.0)^4">
        <function class="functions.continuous.unconstrained.Rastrigin" domain="R(-5.12,5.12)^30"/>
      </function>
    </problem>
  </problems>
  <measurements id="measurements" class="simulator.MeasurementSuite" samples="1" resolution="10">
    <addMeasurement class="measurement.single.Fitness"/>
    <!--          <addMeasurement class="measurement.single.Solution" />-->
  </measurements>
  <simulations>
  <simulation>
      <algorithm idref="ep"/>
      <problem idref="spherical"/>
      <measurements idref="measurements" file="data/masters/ep/amep/amep-spherical.txt"/>
    </simulation>
    <simulation>
```

```
      <algorithm idref="ep"/>
      <problem idref="rosenbrock2d"/>
      <measurements idref="measurements" file="data/masters/ep/amep/amep-rosenbrock2d.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="ep"/>
      <problem idref="step"/>
      <measurements idref="measurements" file="data/masters/ep/amep/amep-step.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="ep"/>
      <problem idref="quartic"/>
      <measurements idref="measurements" file="data/masters/ep/amep/amep-quartic.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="ep"/>
      <problem idref="foxholes"/>
      <measurements idref="measurements" file="data/masters/ep/amep/amep-foxholes.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="ep"/>
      <problem idref="schaffer6"/>
      <measurements idref="measurements" file="data/masters/ep/amep/amep-schaffer6.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="ep"/>
      <problem idref="griewank"/>
      <measurements idref="measurements" file="data/masters/ep/amep/amep-griewank.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="ep"/>
      <problem idref="ackley"/>
      <measurements idref="measurements" file="data/masters/ep/amep/amep-ackley.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="ep"/>
      <problem idref="rosenbrock"/>
      <measurements idref="measurements" file="data/masters/ep/amep/amep-rosenbrock.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="ep"/>
      <problem idref="rastrigin"/>
      <measurements idref="measurements" file="data/masters/ep/amep/amep-rastrigin.txt"/>
    </simulation>
  </simulations>
</simulator>
```

Binary evolutionary programming

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE simulator [
 <!ATTLIST algorithm id ID #IMPLIED>
<!ATTLIST problem id ID #IMPLIED>
<!ATTLIST measurements id ID #IMPLIED>
]>
<simulator>
  <algorithms>
    <algorithm id="ep" class="ec.EC">
      <iterationStrategy class="ec.iterationstrategies.EvolutionaryProgrammingIterationStrategy">
        <mutationStrategy class="entity.operators.mutation.RandomMutationStrategy">
            <mutationProbability class="controlparameter.ConstantControlParameter" parameter="0.5" />
        </mutationStrategy>
      </iterationStrategy>
      <initialisationStrategy class="algorithm.initialisation.ClonedPopulationInitialisationStrategy" entityNumber="40">
        <entityType class="ec.Individual"/>
      </initialisationStrategy>
      <!--<strategyParameterInitialization class="entity.initialization.ConstantInitializationStrategy" constant="3.0"/>-->
      <addStoppingCondition class="stoppingcondition.MaximumIterations" maximumIterations="10000"/>
    </algorithm>
  </algorithms>
  <problems>
    <problem id="spherical" class="problem.FunctionMinimisationProblem">
      <function class="functions.discrete.BinaryAdapter" precision="2" bitsPerDimension="10" domain="B^300">
        <function class="functions.continuous.unconstrained.Spherical" domain="R(-5.12,5.12)^30"/>
      </function>
    </problem>
    <problem id="rosenbrock2d" class="problem.FunctionMinimisationProblem">
      <function class="functions.discrete.BinaryAdapter" precision="3" bitsPerDimension="12" domain="B^24">
        <function class="functions.continuous.unconstrained.Rosenbrock" domain="R(-2.048,2.048)^2"/>
      </function>
    </problem>
    <problem id="step" class="problem.FunctionMinimisationProblem">
      <function class="functions.discrete.BinaryAdapter" precision="2" bitsPerDimension="10" domain="B^50">
        <function class="functions.continuous.Step" domain="R(-5.12,5.12)^5"/>
      </function>
    </problem>
    <problem id="quartic" class="problem.FunctionMinimisationProblem">
      <function class="functions.discrete.BinaryAdapter" precision="2" bitsPerDimension="8" domain="B^240">
        <function class="functions.continuous.Quartic" domain="R(-1.28,1.28)^30"/>
      </function>
    </problem>
    <problem id="foxholes" class="problem.FunctionMinimisationProblem">
      <function class="functions.discrete.BinaryAdapter" precision="0" bitsPerDimension="17" domain="B^34">
        <function class="functions.continuous.Foxholes" domain="R(-65536.0,65536.0)^2"/>
      </function>
    </problem>
    <problem id="schaffer6" class="problem.FunctionMinimisationProblem">
      <function class="functions.discrete.BinaryAdapter" precision="0" bitsPerDimension="8" domain="B^16">
        <function class="functions.continuous.Schaffer6" domain="R(-100.0,100.0)^2"/>
      </function>
    </problem>
     <problem id="griewank" class="problem.FunctionMinimisationProblem">
      <function class="functions.discrete.BinaryAdapter" precision="0" bitsPerDimension="10" domain="B^300">
        <function class="functions.continuous.unconstrained.Griewank" domain="R(-300.0,300.0)^30"/>
      </function>
    </problem>
    <problem id="ackley" class="problem.FunctionMinimisationProblem">
      <function class="functions.discrete.BinaryAdapter" precision="0" bitsPerDimension="6" domain="B^180">
        <function class="functions.continuous.unconstrained.Ackley" domain="R(-30.0,30.0)^30"/>
      </function>
    </problem>
    <problem id="rosenbrock" class="problem.FunctionMinimisationProblem">
      <function class="functions.discrete.BinaryAdapter" precision="3" bitsPerDimension="12" domain="B^360">
        <function class="functions.continuous.unconstrained.Rosenbrock" domain="R(-2.048,2.048)^30"/>
      </function>
    </problem>
    <problem id="rastrigin" class="problem.FunctionMinimisationProblem">
      <function class="functions.discrete.BinaryAdapter" precision="2" bitsPerDimension="10" domain="B^300">
```

```
        <function class="functions.continuous.unconstrained.Rastrigin" domain="R(-5.12,5.12)^30"/>
      </function>
    </problem>
  </problems>
  <measurements id="measurements" class="simulator.MeasurementSuite" samples="1" resolution="1">
    <addMeasurement class="measurement.single.Fitness"/>
    <!--          <addMeasurement class="measurement.single.Solution" />-->
  </measurements>
  <simulations>
  <simulation>
      <algorithm idref="ep"/>
      <problem idref="spherical"/>
      <measurements idref="measurements" file="data/masters/ep/binep/binep-spherical.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="ep"/>
      <problem idref="rosenbrock2d"/>
      <measurements idref="measurements" file="data/masters/ep/binep/binep-rosenbrock2d.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="ep"/>
      <problem idref="step"/>
      <measurements idref="measurements" file="data/masters/ep/binep/binep-step.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="ep"/>
      <problem idref="quartic"/>
      <measurements idref="measurements" file="data/masters/ep/binep/binep-quartic.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="ep"/>
      <problem idref="foxholes"/>
      <measurements idref="measurements" file="data/masters/ep/binep/binep-foxholes.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="ep"/>
      <problem idref="schaffer6"/>
      <measurements idref="measurements" file="data/masters/ep/binep/binep-schaffer6.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="ep"/>
      <problem idref="griewank"/>
      <measurements idref="measurements" file="data/masters/ep/binep/binep-griewank.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="ep"/>
      <problem idref="ackley"/>
      <measurements idref="measurements" file="data/masters/ep/binep/binep-ackley.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="ep"/>
      <problem idref="rosenbrock"/>
      <measurements idref="measurements" file="data/masters/ep/binep/binep-rosenbrock.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="ep"/>
      <problem idref="rastrigin"/>
      <measurements idref="measurements" file="data/masters/ep/binep/binep-rastrigin.txt"/>
    </simulation>
  </simulations>
</simulator>
```

E.2.2 Binary simulations

Angle modulated evolutionary programming

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE simulator [
<!ATTLIST algorithm id ID #IMPLIED>
<!ATTLIST problem id ID #IMPLIED>
<!ATTLIST measurements id ID #IMPLIED>
]>
<simulator>
  <algorithms>
    <algorithm id="ep" class="ec.EC">
      <iterationStrategy class="ec.iterationstrategies.EvolutionaryProgrammingIterationStrategy">
        <mutationStrategy class="entity.operators.mutation.SelfAdaptiveMutationStrategy"/>
      </iterationStrategy>
      <initialisationStrategy class="algorithm.initialisation.ClonedPopulationInitialisationStrategy"
           entityNumber="40">
        <entityType class="ec.Individual"/>
      </initialisationStrategy>
      <strategyParameterInitialization class="entity.initialization.ConstantInitializationStrategy"
           constant="3.0"/>
      <addStoppingCondition class="stoppingcondition.MaximumIterations" maximumIterations="10000"/>
    </algorithm>
  </algorithms>
  <problems>
      <problem id="mdk-10" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.KnapSack" domain="B^10">
    <capacity value="269" />
    <numberOfObjects value="10" />
    <weight value="95,4,60,32,23,72,80,62,64,46" />
    <value value="55,10,47,5,4,50,8,61,85,87" />
  </function>
      </function>
    </problem>
    <problem id="mdk-20" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.KnapSack" domain="B^20">
    <capacity value="878" />
    <numberOfObjects value="20" />
    <weight value="92,4,43,83,84,68,92,82,6,44,32,18,56,83,25,96,70,48,14,58" />
    <value value="44,46,90,72,91,40,75,35,8,54,78,40,77,15,61,17,75,29,75,63" />
  </function>
      </function>
    </problem>
    <problem id="kc-4" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.KnightsCoverage" boardSize="4" />
      </function>
    </problem>
    <problem id="kc-5" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.KnightsCoverage" boardSize="5" />
      </function>
    </problem>
    <problem id="kc-6" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.KnightsCoverage" boardSize="6" />
      </function>
    </problem>
    <problem id="kc-7" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.KnightsCoverage" boardSize="7" />
      </function>
```

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

```
    </problem>
    <problem id="kc-8" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.KnightsCoverage" boardSize="8" />
      </function>
    </problem>
    <problem id="q-4" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.Queens" boardSize="4" />
      </function>
    </problem>
    <problem id="q-5" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.Queens" boardSize="5" />
      </function>
    </problem>
    <problem id="q-6" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.Queens" boardSize="6" />
      </function>
    </problem>
    <problem id="q-7" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.Queens" boardSize="7" />
      </function>
    </problem>
    <problem id="q-8" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.Queens" boardSize="8" />
      </function>
    </problem>
    <problem id="kt-4" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.RepairingKnightsTour" boardSize="4" />
      </function>
    </problem>
    <problem id="kt-5" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.RepairingKnightsTour" boardSize="5" />
      </function>
    </problem>
    <problem id="kt-6" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.RepairingKnightsTour" boardSize="6" />
      </function>
    </problem>
    <problem id="kt-7" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.RepairingKnightsTour" boardSize="7" />
      </function>
    </problem>
    <problem id="kt-8" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.RepairingKnightsTour" boardSize="8" />
      </function>
    </problem>
    <problem id="ipd" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.IteratedPrisonersDilemma" domain="B^64" />
      </function>
    </problem>
  </problems>
  <measurements id="measurements" class="simulator.MeasurementSuite" samples="30" resolution="10">
    <addMeasurement class="measurement.single.Fitness"/>
    <addMeasurement class="measurement.single.CollectiveFitness" />
    <!--          <addMeasurement class="measurement.single.Solution" />-->
  </measurements>
```

```
<simulations>
  <simulation>
    <algorithm idref="ep"/>
    <problem idref="mdk-10"/>
    <measurements idref="measurements" file="data/masters/binary/ep/amep/amep-mdk-10.txt"/>
  </simulation>
  <simulation>
    <algorithm idref="ep"/>
    <problem idref="mdk-20"/>
    <measurements idref="measurements" file="data/masters/binary/ep/amep/amep-mdk-20.txt"/>
  </simulation>
  <simulation>
    <algorithm idref="ep"/>
    <problem idref="kc-4"/>
    <measurements idref="measurements" file="data/masters/binary/ep/amep/amep-kc-4.txt"/>
  </simulation>
  <simulation>
    <algorithm idref="ep"/>
    <problem idref="kc-5"/>
    <measurements idref="measurements" file="data/masters/binary/ep/amep/amep-kc-5.txt"/>
  </simulation>
  <simulation>
    <algorithm idref="ep"/>
    <problem idref="kc-6"/>
    <measurements idref="measurements" file="data/masters/binary/ep/amep/amep-kc-6.txt"/>
  </simulation>
  <simulation>
    <algorithm idref="ep"/>
    <problem idref="kc-7"/>
    <measurements idref="measurements" file="data/masters/binary/ep/amep/amep-kc-7.txt"/>
  </simulation>
  <simulation>
    <algorithm idref="ep"/>
    <problem idref="kc-8"/>
    <measurements idref="measurements" file="data/masters/binary/ep/amep/amep-kc-8.txt"/>
  </simulation>
  <simulation>
    <algorithm idref="ep"/>
    <problem idref="q-4"/>
    <measurements idref="measurements" file="data/masters/binary/ep/amep/amep-q-4.txt"/>
  </simulation>
  <simulation>
    <algorithm idref="ep"/>
    <problem idref="q-5"/>
    <measurements idref="measurements" file="data/masters/binary/ep/amep/amep-q-5.txt"/>
  </simulation>
  <simulation>
    <algorithm idref="ep"/>
    <problem idref="q-6"/>
    <measurements idref="measurements" file="data/masters/binary/ep/amep/amep-q-6.txt"/>
  </simulation>
  <simulation>
    <algorithm idref="ep"/>
    <problem idref="q-7"/>
    <measurements idref="measurements" file="data/masters/binary/ep/amep/amep-q-7.txt"/>
  </simulation>
  <simulation>
    <algorithm idref="ep"/>
    <problem idref="q-8"/>
    <measurements idref="measurements" file="data/masters/binary/ep/amep/amep-q-8.txt"/>
  </simulation>
  <simulation>
    <algorithm idref="ep" />
    <problem idref="kt-4" />
    <measurements idref="measurements" file="data/masters/binary/ep/amep/amep-kt-4.txt" />
  </simulation>
  <simulation>
```

```
      <algorithm idref="ep" />
      <problem idref="kt-5" />
      <measurements idref="measurements" file="data/masters/binary/ep/amep/amep-kt-5.txt" />
    </simulation>
    <simulation>
      <algorithm idref="ep" />
      <problem idref="kt-6" />
      <measurements idref="measurements" file="data/masters/binary/ep/amep/amep-kt-6.txt" />
    </simulation>
    <simulation>
      <algorithm idref="ep" />
      <problem idref="kt-7" />
      <measurements idref="measurements" file="data/masters/binary/ep/amep/amep-kt-7.txt" />
    </simulation>
    <simulation>
      <algorithm idref="ep" />
      <problem idref="kt-8" />
      <measurements idref="measurements" file="data/masters/binary/ep/amep/amep-kt-8.txt" />
    </simulation>
    <simulation>
      <algorithm idref="ep" />
      <problem idref="ipd" />
      <measurements idref="measurements" file="data/masters/binary/ep/amep/amep-ipd.txt" />
    </simulation>
  </simulations>
</simulator>
```

Binary evolutionary programming

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE simulator [
<!ATTLIST algorithm id ID #IMPLIED>
<!ATTLIST problem id ID #IMPLIED>
<!ATTLIST measurements id ID #IMPLIED>
]>
<simulator>
  <algorithms>
    <algorithm id="binep" class="ec.EC">
      <iterationStrategy class="ec.iterationstrategies.EvolutionaryProgrammingIterationStrategy">
        <mutationStrategy class="entity.operators.mutation.RandomMutationStrategy">
            <mutationProbability class="controlparameter.ConstantControlParameter" parameter="0.5" />
        </mutationStrategy>
      </iterationStrategy>
      <initialisationStrategy class="algorithm.initialisation.ClonedPopulationInitialisationStrategy"
          entityNumber="40">
        <entityType class="ec.Individual"/>
      </initialisationStrategy>
      <!--<strategyParameterInitialization class="entity.initialization.ConstantInitializationStrategy
          " constant="3.0"/>-->
      <addStoppingCondition class="stoppingcondition.MaximumIterations" maximumIterations="10000"/>
    </algorithm>
  </algorithms>
  <problems>
    <problem id="mdk-10" class="problem.FunctionMaximisationProblem">
        <function class="functions.discrete.KnapSack" domain="B^10">
          <capacity value="269" />
          <numberOfObjects value="10" />
          <weight value="95,4,60,32,23,72,80,62,64,46" />
          <value value="55,10,47,5,4,50,8,61,85,87" />
        </function>
    </problem>
    <problem id="mdk-20" class="problem.FunctionMaximisationProblem">
        <function class="functions.discrete.KnapSack" domain="B^20">
    <capacity value="878" />
    <numberOfObjects value="20" />
    <weight value="92,4,43,83,84,68,92,82,6,44,32,18,56,83,25,96,70,48,14,58" />
    <value value="44,46,90,72,91,40,75,35,8,54,78,40,77,15,61,17,75,29,75,63" />
  </function>
    </problem>
    <problem id="kc-4" class="problem.FunctionMinimisationProblem">
        <function class="functions.discrete.KnightsCoverage" boardSize="4" />
    </problem>
    <problem id="kc-5" class="problem.FunctionMinimisationProblem">
        <function class="functions.discrete.KnightsCoverage" boardSize="5" />
    </problem>
    <problem id="kc-6" class="problem.FunctionMinimisationProblem">
        <function class="functions.discrete.KnightsCoverage" boardSize="6" />
    </problem>
    <problem id="kc-7" class="problem.FunctionMinimisationProblem">
        <function class="functions.discrete.KnightsCoverage" boardSize="7" />
    </problem>
    <problem id="kc-8" class="problem.FunctionMinimisationProblem">
        <function class="functions.discrete.KnightsCoverage" boardSize="8" />
    </problem>
    <problem id="q-4" class="problem.FunctionMinimisationProblem">
        <function class="functions.discrete.Queens" boardSize="4" />
    </problem>
    <problem id="q-5" class="problem.FunctionMinimisationProblem">
        <function class="functions.discrete.Queens" boardSize="5" />
    </problem>
    <problem id="q-6" class="problem.FunctionMinimisationProblem">
        <function class="functions.discrete.Queens" boardSize="6" />
    </problem>
    <problem id="q-7" class="problem.FunctionMinimisationProblem">
        <function class="functions.discrete.Queens" boardSize="7" />
    </problem>
    <problem id="q-8" class="problem.FunctionMinimisationProblem">
        <function class="functions.discrete.Queens" boardSize="8" />
    </problem>
    <problem id="kt-4" class="problem.FunctionMaximisationProblem">
        <function class="functions.discrete.RepairingKnightsTour" boardSize="4" />
    </problem>
    <problem id="kt-5" class="problem.FunctionMaximisationProblem">
        <function class="functions.discrete.RepairingKnightsTour" boardSize="5" />
    </problem>
    <problem id="kt-6" class="problem.FunctionMaximisationProblem">
        <function class="functions.discrete.RepairingKnightsTour" boardSize="6" />
    </problem>
    <problem id="kt-7" class="problem.FunctionMaximisationProblem">
        <function class="functions.discrete.RepairingKnightsTour" boardSize="7" />
    </problem>
    <problem id="kt-8" class="problem.FunctionMaximisationProblem">
        <function class="functions.discrete.RepairingKnightsTour" boardSize="8" />
    </problem>
    <problem id="ipd" class="problem.FunctionMaximisationProblem">
        <function class="functions.discrete.IteratedPrisonersDilemma" domain="B^64" />
    </problem>
  </problems>
  <measurements id="measurements" class="simulator.MeasurementSuite" samples="30" resolution="1">
    <addMeasurement class="measurement.single.Fitness"/>
    <addMeasurement class="measurement.single.CollectiveFitness" />
    <!--        <addMeasurement class="measurement.single.Solution" />-->
  </measurements>
  <simulations>
    <simulation>
      <algorithm idref="binep"/>
      <problem idref="mdk-10"/>
      <measurements idref="measurements" file="data/masters/binary/ep/binep/binep-mdk-10.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binep"/>
      <problem idref="mdk-20"/>
      <measurements idref="measurements" file="data/masters/binary/ep/binep/binep-mdk-20.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binep"/>
      <problem idref="kc-4"/>
      <measurements idref="measurements" file="data/masters/binary/ep/binep/binep-kc-4.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binep"/>
      <problem idref="kc-5"/>
      <measurements idref="measurements" file="data/masters/binary/ep/binep/binep-kc-5.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binep"/>
      <problem idref="kc-6"/>
      <measurements idref="measurements" file="data/masters/binary/ep/binep/binep-kc-6.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binep"/>
      <problem idref="kc-7"/>
      <measurements idref="measurements" file="data/masters/binary/ep/binep/binep-kc-7.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binep"/>
      <problem idref="kc-8"/>
      <measurements idref="measurements" file="data/masters/binary/ep/binep/binep-kc-8.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binep"/>
      <problem idref="q-4"/>
      <measurements idref="measurements" file="data/masters/binary/ep/binep/binep-q-4.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binep"/>
      <problem idref="q-5"/>
      <measurements idref="measurements" file="data/masters/binary/ep/binep/binep-q-5.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binep"/>
      <problem idref="q-6"/>
      <measurements idref="measurements" file="data/masters/binary/ep/binep/binep-q-6.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binep"/>
      <problem idref="q-7"/>
      <measurements idref="measurements" file="data/masters/binary/ep/binep/binep-q-7.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binep"/>
      <problem idref="q-8"/>
      <measurements idref="measurements" file="data/masters/binary/ep/binep/binep-q-8.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binep" />
      <problem idref="kt-4" />
      <measurements idref="measurements" file="data/masters/binary/ep/binep/binep-kt-4.txt" />
    </simulation>
        <simulation>
      <algorithm idref="binep" />
      <problem idref="kt-5" />
      <measurements idref="measurements" file="data/masters/binary/ep/binep/binep-kt-5.txt" />
    </simulation>
        <simulation>
      <algorithm idref="binep" />
      <problem idref="kt-6" />
```

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

```
      <measurements idref="measurements" file="data/masters/binary/ep/binep/binep-kt-6.txt" />
    </simulation>
        <simulation>
      <algorithm idref="binep" />
      <problem idref="kt-7" />
      <measurements idref="measurements" file="data/masters/binary/ep/binep/binep-kt-7.txt" />
    </simulation>
    <simulation>
      <algorithm idref="binep" />
      <problem idref="kt-8" />
      <measurements idref="measurements" file="data/masters/binary/ep/binep/binep-kt-8.txt" />
    </simulation>
    <simulation>
      <algorithm idref="binep" />
      <problem idref="ipd" />
      <measurements idref="measurements" file="data/masters/binary/ep/binep/binep-ipd.txt" />
    </simulation>
  </simulations>
</simulator>
```

E.3 Differential evolution

E.3.1 Continuous simulations

Angle modulated differential evolution

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE simulator [
 <!ATTLIST algorithm id ID #IMPLIED>
<!ATTLIST problem id ID #IMPLIED>
<!ATTLIST measurements id ID #IMPLIED>
]>
<simulator>
  <algorithms>
    <algorithm id="de" class="ec.EC">
      <initialisationStrategy
       class="algorithm.initialisation.ClonedPopulationInitialisationStrategy">
        <entityType class="ec.Individual" />
  <entityNumber value="40"/>
      </initialisationStrategy>
      <iterationStrategy
       class="ec.iterationstrategies.DifferentialEvolutionIterationStrategy">
  <!--<trialVectorCreationStrategy
   class="entity.operators.creation.RandCreationStrategy">
    <scaleParameter
     class="controlparameter.ConstantControlParameter"
     parameter="1.0" />
  </trialVectorCreationStrategy>
  <crossoverStrategy
   class="entity.operators.crossover.DifferentialEvolutionBinomialCrossover">
    <crossoverProbability
     class="controlparameter.ConstantControlParameter"
     parameter="0.25"/>
  </crossoverStrategy>-->
  <boundaryConstraint
   class="problem.boundaryconstraint.ClampingBoundaryConstraint"/>
      </iterationStrategy>
      <addStoppingCondition class="stoppingcondition.MaximumIterations" maximumIterations="10000"/>
    </algorithm>
```

```
</algorithms>
<problems>
  <problem id="spherical" class="problem.FunctionMinimisationProblem">
    <function class="functions.continuous.decorators.AngleModulation" precision="2" domain="R
        (-1.0,1.0)^4">
      <function class="functions.continuous.unconstrained.Spherical" domain="R(-5.12,5.12)^30"/>
    </function>
  </problem>
  <problem id="rosenbrock2d" class="problem.FunctionMinimisationProblem">
    <function class="functions.continuous.decorators.AngleModulation" precision="3" domain="R
        (-1.0,1.0)^4">
      <function class="functions.continuous.unconstrained.Rosenbrock" domain="R(-2.048,2.048)^2"/>
    </function>
  </problem>
  <problem id="step" class="problem.FunctionMinimisationProblem">
    <function class="functions.continuous.decorators.AngleModulation" precision="2" domain="R
        (-1.0,1.0)^4">
      <function class="functions.continuous.unconstrained.Step" domain="R(-5.12,5.12)^5"/>
    </function>
  </problem>
  <problem id="quartic" class="problem.FunctionMinimisationProblem">
    <function class="functions.continuous.decorators.AngleModulation" precision="2" domain="R
        (-1.0,1.0)^4">
      <function class="functions.continuous.Quartic" domain="R(-1.28,1.28)^30"/>
    </function>
  </problem>
  <problem id="foxholes" class="problem.FunctionMinimisationProblem">
    <function class="functions.continuous.decorators.AngleModulation" precision="2" domain="R
        (-1.0,1.0)^4">
      <function class="functions.continuous.Foxholes" domain="R(-65536.0,65536.0)^2"/>
    </function>
  </problem>
  <problem id="schaffer6" class="problem.FunctionMinimisationProblem">
    <function class="functions.continuous.decorators.AngleModulation" precision="2" domain="R
        (-1.0,1.0)^4">
      <function class="functions.continuous.unconstrained.Schaffer6" domain="R(-100.0,100.0)^2"/>
    </function>
  </problem>
 <problem id="griewank" class="problem.FunctionMinimisationProblem">
    <function class="functions.continuous.decorators.AngleModulation" precision="2" domain="R
        (-1.0,1.0)^4">
      <function class="functions.continuous.unconstrained.Griewank" domain="R(-300.0,300.0)^30"/>
    </function>
  </problem>
  <problem id="ackley" class="problem.FunctionMinimisationProblem">
    <function class="functions.continuous.decorators.AngleModulation" precision="2" domain="R
        (-1.0,1.0)^4">
      <function class="functions.continuous.unconstrained.Ackley" domain="R(-30.0,30.0)^30"/>
    </function>
  </problem>
  <problem id="rosenbrock" class="problem.FunctionMinimisationProblem">
    <function class="functions.continuous.decorators.AngleModulation" precision="3" domain="R
        (-1.0,1.0)^4">
      <function class="functions.continuous.unconstrained.Rosenbrock" domain="R(-2.048,2.048)^30"/>
    </function>
  </problem>
  <problem id="rastrigin" class="problem.FunctionMinimisationProblem">
    <function class="functions.continuous.decorators.AngleModulation" precision="2" domain="R
        (-1.0,1.0)^4">
      <function class="functions.continuous.unconstrained.Rastrigin" domain="R(-5.12,5.12)^30"/>
    </function>
  </problem>
</problems>
<measurements id="measurements" class="simulator.MeasurementSuite" samples="1" resolution="1">
  <addMeasurement class="measurement.single.Fitness"/>
<!-- <addMeasurement class="measurement.single.Solution"/>-->
</measurements>
```

```
  <simulations>
  <simulation>
      <algorithm idref="de"/>
      <problem idref="spherical"/>
      <measurements idref="measurements" file="data/masters/de/amde/amde-spherical.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="de"/>
      <problem idref="rosenbrock2d"/>
      <measurements idref="measurements" file="data/masters/de/amde/amde-rosenbrock2d.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="de"/>
      <problem idref="step"/>
      <measurements idref="measurements" file="data/masters/de/amde/amde-step.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="de"/>
      <problem idref="quartic"/>
      <measurements idref="measurements" file="data/masters/de/amde/amde-quartic.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="de"/>
      <problem idref="foxholes"/>
      <measurements idref="measurements" file="data/masters/de/amde/amde-foxholes.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="de"/>
      <problem idref="schaffer6"/>
      <measurements idref="measurements" file="data/masters/de/amde/amde-schaffer6.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="de"/>
      <problem idref="griewank"/>
      <measurements idref="measurements" file="data/masters/de/amde/amde-griewank.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="de"/>
      <problem idref="ackley"/>
      <measurements idref="measurements" file="data/masters/de/amde/amde-ackley.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="de"/>
      <problem idref="rosenbrock"/>
      <measurements idref="measurements" file="data/masters/de/amde/amde-rosenbrock.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="de"/>
      <problem idref="rastrigin"/>
      <measurements idref="measurements" file="data/masters/de/amde/amde-rastrigin.txt"/>
    </simulation>
  </simulations>
</simulator>
```

Binary differential evolution

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE simulator [
 <!ATTLIST algorithm id ID #IMPLIED>
<!ATTLIST problem id ID #IMPLIED>
<!ATTLIST measurements id ID #IMPLIED>
]>
<simulator>
```

```
<algorithms>
  <algorithm id="de" class="ec.EC">
    <initialisationStrategy class="algorithm.initialisation.ClonedPopulationInitialisationStrategy">
      <entityType class="ec.Individual">
  <fitnessCalculator class="util.calculator.BinaryConvertedFitnessCalculator" />
</entityType>
<entityNumber value="40"/>
    </initialisationStrategy>
    <iterationStrategy class="ec.iterationstrategies.DifferentialEvolutionIterationStrategy">
<!--<trialVectorCreationStrategy class="entity.operators.creation.RandCreationStrategy">
  <scaleParameter class="controlparameter.ConstantControlParameter" parameter="1.0" />
</trialVectorCreationStrategy>
<crossoverStrategy class="entity.operators.crossover.DifferentialEvolutionBinomialCrossover">
  <crossoverProbability class="controlparameter.ConstantControlParameter" parameter="0.25"/>
</crossoverStrategy>-->
<boundaryConstraint class="problem.boundaryconstraint.ClampingBoundaryConstraint"/>
    </iterationStrategy>
    <addStoppingCondition class="stoppingcondition.MaximumIterations" maximumIterations="10000"/>
  </algorithm>
</algorithms>
<problems>
  <problem id="spherical" class="problem.FunctionMinimisationProblem">
    <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^300">
      <function class="functions.discrete.BinaryAdapter" precision="2" bitsPerDimension="10" domain=
          "B^300">
        <function class="functions.continuous.unconstrained.Spherical" domain="R(-5.12,5.12)^30"/>
      </function>
    </function>
  </problem>
  <problem id="rosenbrock2d" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^24">
    <function class="functions.discrete.BinaryAdapter" precision="3" bitsPerDimension="12" domain="B
        ^24">
      <function class="functions.continuous.unconstrained.Rosenbrock" domain="R(-2.048,2.048)^2"/>
    </function>
    </function>
  </problem>
  <problem id="step" class="problem.FunctionMinimisationProblem">
    <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^50">
      <function class="functions.discrete.BinaryAdapter" precision="2" bitsPerDimension="10" domain=
          "B^50">
        <function class="functions.continuous.unconstrained.Step" domain="R(-5.12,5.12)^5"/>
      </function>
    </function>
  </problem>
  <problem id="quartic" class="problem.FunctionMinimisationProblem">
    <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^240">
      <function class="functions.discrete.BinaryAdapter" precision="2" bitsPerDimension="8" domain="
          B^240">
        <function class="functions.continuous.Quartic" domain="R(-1.28,1.28)^30"/>
      </function>
    </function>
  </problem>
  <problem id="foxholes" class="problem.FunctionMinimisationProblem">
    <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^34">
    <function class="functions.discrete.BinaryAdapter" precision="0" bitsPerDimension="17" domain="B
        ^34">
      <function class="functions.continuous.Foxholes" domain="R(-65536.0,65536.0)^2"/>
    </function>
    </function>
  </problem>
  <problem id="schaffer6" class="problem.FunctionMinimisationProblem">
    <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^16">
    <function class="functions.discrete.BinaryAdapter" precision="0" bitsPerDimension="8" domain="B
        ^16">
      <function class="functions.continuous.unconstrained.Schaffer6" domain="R(-100.0,100.0)^2"/>
    </function>
    </function>
  </problem>
 <problem id="griewank" class="problem.FunctionMinimisationProblem">
     <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^300">
    <function class="functions.discrete.BinaryAdapter" precision="0" bitsPerDimension="10" domain="B^300">
      <function class="functions.continuous.unconstrained.Griewank" domain="R(-300.0,300.0)^30"/>
    </function>
    </function>
  </problem>
  <problem id="ackley" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^180">
    <function class="functions.discrete.BinaryAdapter" precision="0" bitsPerDimension="6" domain="B^180">
      <function class="functions.continuous.unconstrained.Ackley" domain="R(-30.0,30.0)^30"/>
    </function>
    </function>
  </problem>
  <problem id="rosenbrock" class="problem.FunctionMinimisationProblem">
    <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^360">
    <function class="functions.discrete.BinaryAdapter" precision="3" bitsPerDimension="12" domain="B^360">
      <function class="functions.continuous.unconstrained.Rosenbrock" domain="R(-2.048,2.048)^30"/>
    </function>
    </function>
  </problem>
  <problem id="rastrigin" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^300">
    <function class="functions.discrete.BinaryAdapter" precision="2" bitsPerDimension="10" domain="B^300">
      <function class="functions.continuous.unconstrained.Rastrigin" domain="R(-5.12,5.12)^30"/>
    </function>
    </function>
  </problem>
</problems>
<measurements id="measurements" class="simulator.MeasurementSuite" samples="1" resolution="1">
  <addMeasurement class="measurement.single.Fitness"/>
<!-- <addMeasurement class="measurement.single.Solution"/>-->
</measurements>
<simulations>
<simulation>
    <algorithm idref="de"/>
    <problem idref="spherical"/>
    <measurements idref="measurements" file="data/masters/de/binde/binde-spherical.txt"/>
  </simulation>
  <simulation>
    <algorithm idref="de"/>
    <problem idref="rosenbrock2d"/>
    <measurements idref="measurements" file="data/masters/de/binde/binde-rosenbrock2d.txt"/>
  </simulation>
  <simulation>
    <algorithm idref="de"/>
    <problem idref="step"/>
    <measurements idref="measurements" file="data/masters/de/binde/binde-step.txt"/>
  </simulation>
  <simulation>
    <algorithm idref="de"/>
    <problem idref="quartic"/>
    <measurements idref="measurements" file="data/masters/de/binde/binde-quartic.txt"/>
  </simulation>
  <simulation>
    <algorithm idref="de"/>
    <problem idref="foxholes"/>
    <measurements idref="measurements" file="data/masters/de/binde/binde-foxholes.txt"/>
  </simulation>
  <simulation>
    <algorithm idref="de"/>
```

```
      <problem idref="schaffer6"/>
      <measurements idref="measurements" file="data/masters/de/binde/binde-schaffer6.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="de"/>
      <problem idref="griewank"/>
      <measurements idref="measurements" file="data/masters/de/binde/binde-griewank.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="de"/>
      <problem idref="ackley"/>
      <measurements idref="measurements" file="data/masters/de/binde/binde-ackley.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="de"/>
      <problem idref="rosenbrock"/>
      <measurements idref="measurements" file="data/masters/de/binde/binde-rosenbrock.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="de"/>
      <problem idref="rastrigin"/>
      <measurements idref="measurements" file="data/masters/de/binde/binde-rastrigin.txt"/>
    </simulation>
  </simulations>
</simulator>
```

Normalized differential evolution

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE simulator [
<!ATTLIST algorithm id ID #IMPLIED>
<!ATTLIST problem id ID #IMPLIED>
<!ATTLIST measurements id ID #IMPLIED>
]>
<simulator>
  <algorithms>
    <algorithm id="de" class="ec.EC">
      <initialisationStrategy class="algorithm.initialisation.ClonedPopulationInitialisationStrategy">
        <entityType class="ec.Individual">
    <fitnessCalculator class="util.calculator.NormalizedFitnessCalculator" />
  </entityType>
  <entityNumber value="40"/>
      </initialisationStrategy>
      <iterationStrategy class="ec.iterationstrategies.DifferentialEvolutionIterationStrategy">
  <!--<trialVectorCreationStrategy class="entity.operators.creation.RandCreationStrategy">
    <scaleParameter class="controlparameter.ConstantControlParameter" parameter="1.0" />
  </trialVectorCreationStrategy>
  <crossoverStrategy class="entity.operators.crossover.DifferentialEvolutionBinomialCrossover">
    <crossoverProbability class="controlparameter.ConstantControlParameter" parameter="0.25"/>
  </crossoverStrategy>-->
  <boundaryConstraint class="problem.boundaryconstraint.ClampingBoundaryConstraint"/>
      </iterationStrategy>
      <addStoppingCondition class="stoppingcondition.MaximumIterations" maximumIterations="10000"/>
    </algorithm>
  </algorithms>
  <problems>
    <problem id="spherical" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^300">
        <function class="functions.discrete.BinaryAdapter" precision="2" bitsPerDimension="10" domain=
            "B^300">
          <function class="functions.continuous.unconstrained.Spherical" domain="R(-5.12,5.12)^30"/>
        </function>
      </function>
    </problem>
    <problem id="rosenbrock2d" class="problem.FunctionMinimisationProblem">
        <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^24">
      <function class="functions.discrete.BinaryAdapter" precision="3" bitsPerDimension="12" domain="B
          ^24">
        <function class="functions.continuous.unconstrained.Rosenbrock" domain="R(-2.048,2.048)^2"/>
        </function>
      </function>
    </problem>
    <problem id="step" class="problem.FunctionMinimisationProblem">
        <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^50">
      <function class="functions.discrete.BinaryAdapter" precision="2" bitsPerDimension="10" domain="B
          ^50">
        <function class="functions.continuous.Step" domain="R(-5.12,5.12)^5"/>
        </function>
      </function>
    </problem>
    <problem id="quartic" class="problem.FunctionMinimisationProblem">
        <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^240">
      <function class="functions.discrete.BinaryAdapter" precision="2" bitsPerDimension="8" domain="B
          ^240">
        <function class="functions.continuous.Quartic" domain="R(-1.28,1.28)^30"/>
      </function>
      </function>
    </problem>
    <problem id="foxholes" class="problem.FunctionMinimisationProblem">
        <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^34">
      <function class="functions.discrete.BinaryAdapter" precision="0" bitsPerDimension="17" domain="B
          ^34">
        <function class="functions.continuous.Foxholes" domain="R(-65536.0,65536.0)^2"/>
      </function>
      </function>
    </problem>
    <problem id="schaffer6" class="problem.FunctionMinimisationProblem">
        <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^16">
      <function class="functions.discrete.BinaryAdapter" precision="0" bitsPerDimension="8" domain="B
          ^16">
        <function class="functions.continuous.Schaffer6" domain="R(-100.0,100.0)^2"/>
      </function>
      </function>
    </problem>
   <problem id="griewank" class="problem.FunctionMinimisationProblem">
       <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^300">
      <function class="functions.discrete.BinaryAdapter" precision="0" bitsPerDimension="10" domain="B
          ^300">
        <function class="functions.continuous.unconstrained.Griewank" domain="R(-300.0,300.0)^30"/>
      </function>
      </function>
    </problem>
    <problem id="ackley" class="problem.FunctionMinimisationProblem">
        <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^180">
      <function class="functions.discrete.BinaryAdapter" precision="0" bitsPerDimension="6" domain="B
          ^180">
        <function class="functions.continuous.unconstrained.Ackley" domain="R(-30.0,30.0)^30"/>
      </function>
      </function>
    </problem>
    <problem id="rosenbrock" class="problem.FunctionMinimisationProblem">
        <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^360">
      <function class="functions.discrete.BinaryAdapter" precision="3" bitsPerDimension="12" domain="B
          ^360">
        <function class="functions.continuous.unconstrained.Rosenbrock" domain="R(-2.048,2.048)^30"/>
      </function>
      </function>
    </problem>
    <problem id="rastrigin" class="problem.FunctionMinimisationProblem">
        <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^300">
      <function class="functions.discrete.BinaryAdapter" precision="2" bitsPerDimension="10" domain="B
          ^300">
        <function class="functions.continuous.unconstrained.Rastrigin" domain="R(-5.12,5.12)^30"/>
      </function>
      </function>
    </problem>
  </problems>
  <measurements id="measurements" class="simulator.MeasurementSuite" samples="1" resolution="1">
    <addMeasurement class="measurement.single.Fitness"/>
  <!-- <addMeasurement class="measurement.single.Solution"/>-->
  </measurements>
  <simulations>
  <simulation>
      <algorithm idref="de"/>
      <problem idref="spherical"/>
      <measurements idref="measurements" file="data/masters/de/normde/normde-spherical.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="de"/>
      <problem idref="rosenbrock2d"/>
      <measurements idref="measurements" file="data/masters/de/normde/normde-rosenbrock2d.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="de"/>
      <problem idref="step"/>
      <measurements idref="measurements" file="data/masters/de/normde/normde-step.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="de"/>
      <problem idref="quartic"/>
      <measurements idref="measurements" file="data/masters/de/normde/normde-quartic.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="de"/>
      <problem idref="foxholes"/>
      <measurements idref="measurements" file="data/masters/de/normde/normde-foxholes.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="de"/>
      <problem idref="schaffer6"/>
      <measurements idref="measurements" file="data/masters/de/normde/normde-schaffer6.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="de"/>
      <problem idref="griewank"/>
      <measurements idref="measurements" file="data/masters/de/normde/normde-griewank.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="de"/>
      <problem idref="ackley"/>
      <measurements idref="measurements" file="data/masters/de/normde/normde-ackley.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="de"/>
      <problem idref="rosenbrock"/>
      <measurements idref="measurements" file="data/masters/de/normde/normde-rosenbrock.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="de"/>
      <problem idref="rastrigin"/>
      <measurements idref="measurements" file="data/masters/de/normde/normde-rastrigin.txt"/>
    </simulation>
  </simulations>
</simulator>
```

E.3.2 Binary simulations

Angle modulated differential evolution

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE simulator [
<!ATTLIST algorithm id ID #IMPLIED>
<!ATTLIST problem id ID #IMPLIED>
<!ATTLIST measurements id ID #IMPLIED>
]>
<simulator>
  <algorithms>
    <algorithm id="de" class="ec.EC">
      <initialisationStrategy
       class="algorithm.initialisation.ClonedPopulationInitialisationStrategy">
        <entityType class="ec.Individual" />
  <entityNumber value="40"/>
      </initialisationStrategy>
      <iterationStrategy
       class="ec.iterationstrategies.DifferentialEvolutionIterationStrategy">
  <boundaryConstraint
   class="problem.boundaryconstraint.ClampingBoundaryConstraint"/>
      </iterationStrategy>
      <addStoppingCondition class="stoppingcondition.MaximumIterations" maximumIterations="10000"/>
    </algorithm>
  </algorithms>
  <problems>
    <problem id="mdk-10" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.KnapSack" domain="B^10">
    <capacity value="269" />
    <numberOfObjects value="10" />
    <weight value="95,4,60,32,23,72,80,62,64,46" />
    <value value="55,10,47,5,4,50,8,61,85,87" />
  </function>
      </function>
    </problem>
    <problem id="mdk-20" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.KnapSack" domain="B^20">
    <capacity value="878" />
    <numberOfObjects value="20" />
    <weight value="92,4,43,83,84,68,92,82,6,44,32,18,56,83,25,96,70,48,14,58" />
    <value value="44,46,90,72,91,40,75,35,8,54,78,40,77,15,61,17,75,29,75,63" />
  </function>
      </function>
    </problem>
    <problem id="kc-4" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.KnightsCoverage" boardSize="4" />
      </function>
    </problem>
    <problem id="kc-5" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.KnightsCoverage" boardSize="5" />
      </function>
    </problem>
    <problem id="kc-6" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.KnightsCoverage" boardSize="6" />
      </function>
    </problem>
    <problem id="kc-7" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.KnightsCoverage" boardSize="7" />
      </function>
    </problem>
    <problem id="kc-8" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.KnightsCoverage" boardSize="8" />
      </function>
    </problem>
    <problem id="q-4" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.Queens" boardSize="4" />
      </function>
    </problem>
    <problem id="q-5" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.Queens" boardSize="5" />
      </function>
    </problem>
    <problem id="q-6" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.Queens" boardSize="6" />
      </function>
    </problem>
    <problem id="q-7" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.Queens" boardSize="7" />
      </function>
    </problem>
    <problem id="q-8" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.Queens" boardSize="8" />
      </function>
    </problem>
    <problem id="kt-4" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.RepairingKnightsTour" boardSize="4" />
      </function>
    </problem>
    <problem id="kt-5" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.RepairingKnightsTour" boardSize="5" />
      </function>
    </problem>
    <problem id="kt-6" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.RepairingKnightsTour" boardSize="6" />
      </function>
    </problem>
    <problem id="kt-7" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.RepairingKnightsTour" boardSize="7" />
      </function>
    </problem>
    <problem id="kt-8" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.RepairingKnightsTour" boardSize="8" />
      </function>
    </problem>
    <problem id="ipd" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.IteratedPrisonersDilemma" domain="B^64" />
      </function>
    </problem>
  </problems>
  <measurements id="measurements" class="simulator.MeasurementSuite" samples="30" resolution="1">
    <addMeasurement class="measurement.single.Fitness"/>
    <addMeasurement class="measurement.single.CollectiveFitness" />
  <!-- <addMeasurement class="measurement.single.Solution"/>-->
```

```
</measurements>
<simulations>
  <simulation>
    <algorithm idref="de"/>
    <problem idref="mdk−10"/>
    <measurements idref="measurements" file="data/masters/binary/de/amde/amde−mdk−10.txt"/>
  </simulation>
  <simulation>
    <algorithm idref="de"/>
    <problem idref="mdk−20"/>
    <measurements idref="measurements" file="data/masters/binary/de/amde/amde−mdk−20.txt"/>
  </simulation>
  <simulation>
    <algorithm idref="de"/>
    <problem idref="kc−4"/>
    <measurements idref="measurements" file="data/masters/binary/de/amde/amde−kc−4.txt"/>
  </simulation>
  <simulation>
    <algorithm idref="de"/>
    <problem idref="kc−5"/>
    <measurements idref="measurements" file="data/masters/binary/de/amde/amde−kc−5.txt"/>
  </simulation>
  <simulation>
    <algorithm idref="de"/>
    <problem idref="kc−6"/>
    <measurements idref="measurements" file="data/masters/binary/de/amde/amde−kc−6.txt"/>
  </simulation>
  <simulation>
    <algorithm idref="de"/>
    <problem idref="kc−7"/>
    <measurements idref="measurements" file="data/masters/binary/de/amde/amde−kc−7.txt"/>
  </simulation>
  <simulation>
    <algorithm idref="de"/>
    <problem idref="kc−8"/>
    <measurements idref="measurements" file="data/masters/binary/de/amde/amde−kc−8.txt"/>
  </simulation>
  <simulation>
    <algorithm idref="de"/>
    <problem idref="q−4"/>
    <measurements idref="measurements" file="data/masters/binary/de/amde/amde−q−4.txt"/>
  </simulation>
  <simulation>
    <algorithm idref="de"/>
    <problem idref="q−5"/>
    <measurements idref="measurements" file="data/masters/binary/de/amde/amde−q−5.txt"/>
  </simulation>
  <simulation>
    <algorithm idref="de"/>
    <problem idref="q−6"/>
    <measurements idref="measurements" file="data/masters/binary/de/amde/amde−q−6.txt"/>
  </simulation>
  <simulation>
    <algorithm idref="de"/>
    <problem idref="q−7"/>
    <measurements idref="measurements" file="data/masters/binary/de/amde/amde−q−7.txt"/>
  </simulation>
  <simulation>
    <algorithm idref="de"/>
    <problem idref="q−8"/>
    <measurements idref="measurements" file="data/masters/binary/de/amde/amde−q−8.txt"/>
  </simulation>
  <simulation>
    <algorithm idref="de" />
    <problem idref="kt−4" />
    <measurements idref="measurements" file="data/masters/binary/de/amde/amde−kt−4.txt" />
  </simulation>
```

```
    <simulation>
      <algorithm idref="de" />
      <problem idref="kt-5" />
      <measurements idref="measurements" file="data/masters/binary/de/amde/amde-kt-5.txt" />
    </simulation>
    <simulation>
      <algorithm idref="de" />
      <problem idref="kt-6" />
      <measurements idref="measurements" file="data/masters/binary/de/amde/amde-kt-6.txt" />
    </simulation>
    <simulation>
      <algorithm idref="de" />
      <problem idref="kt-7" />
      <measurements idref="measurements" file="data/masters/binary/de/amde/amde-kt-7.txt" />
    </simulation>
    <simulation>
      <algorithm idref="de" />
      <problem idref="kt-8" />
      <measurements idref="measurements" file="data/masters/binary/de/amde/amde-kt-8.txt" />
    </simulation>
    <simulation>
      <algorithm idref="de" />
      <problem idref="ipd" />
      <measurements idref="measurements" file="data/masters/binary/de/amde/amde-ipd.txt" />
    </simulation>
  </simulations>
</simulator>
```

Binary differential evolution

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE simulator [
<!ATTLIST algorithm id ID #IMPLIED>
<!ATTLIST problem id ID #IMPLIED>
<!ATTLIST measurements id ID #IMPLIED>
]>
<simulator>
  <algorithms>
    <algorithm id="de" class="ec.EC">
      <initialisationStrategy class="algorithm.initialisation.ClonedPopulationInitialisationStrategy">
        <entityType class="ec.Individual">
    <fitnessCalculator class="util.calculator.BinaryConvertedFitnessCalculator" />
  </entityType>
  <entityNumber value="40"/>
      </initialisationStrategy>
      <iterationStrategy class="ec.iterationstrategies.DifferentialEvolutionIterationStrategy">
  <!--<trialVectorCreationStrategy class="entity.operators.creation.RandCreationStrategy">
    <scaleParameter class="controlparameter.ConstantControlParameter" parameter="1.0" />
  </trialVectorCreationStrategy>
  <crossoverStrategy class="entity.operators.crossover.DifferentialEvolutionBinomialCrossover">
    <crossoverProbability class="controlparameter.ConstantControlParameter" parameter="0.25"/>
  </crossoverStrategy>-->
  <boundaryConstraint class="problem.boundaryconstraint.ClampingBoundaryConstraint"/>
      </iterationStrategy>
      <addStoppingCondition class="stoppingcondition.MaximumIterations" maximumIterations="10000"/>
    </algorithm>
  </algorithms>
  <problems>
    <problem id="mdk-10" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^10">
          <function class="functions.discrete.KnapSack" domain="B^10">
    <capacity value="269" />
    <numberOfObjects value="10" />
    <weight value="95,4,60,32,23,72,80,62,64,46" />
    <value value="55,10,47,5,4,50,8,61,85,87" />
        </function>
      </function>
    </problem>
    <problem id="mdk-20" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^20">
        <function class="functions.discrete.KnapSack" domain="B^20">
    <capacity value="878" />
    <numberOfObjects value="20" />
    <weight value="92,4,43,83,84,68,92,82,6,44,32,18,56,83,25,96,70,48,14,58" />
    <value value="44,46,90,72,91,40,75,35,8,54,78,40,77,15,61,17,75,29,75,63" />
  </function>
      </function>
    </problem>
    <problem id="kc-4" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^16">
        <function class="functions.discrete.KnightsCoverage" boardSize="4" />
      </function>
    </problem>
    <problem id="kc-5" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^25">
        <function class="functions.discrete.KnightsCoverage" boardSize="5" />
      </function>
    </problem>
    <problem id="kc-6" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^36">
        <function class="functions.discrete.KnightsCoverage" boardSize="6" />
      </function>
    </problem>
    <problem id="kc-7" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^49">
        <function class="functions.discrete.KnightsCoverage" boardSize="7" />
      </function>
    </problem>
    <problem id="kc-8" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^64">
        <function class="functions.discrete.KnightsCoverage" boardSize="8" />
      </function>
    </problem>
    <problem id="q-4" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^16">
        <function class="functions.discrete.Queens" boardSize="4" />
      </function>
    </problem>
    <problem id="q-5" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^25">
        <function class="functions.discrete.Queens" boardSize="5" />
      </function>
    </problem>
    <problem id="q-6" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^36">
        <function class="functions.discrete.Queens" boardSize="6" />
      </function>
    </problem>
    <problem id="q-7" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^49">
        <function class="functions.discrete.Queens" boardSize="7" />
      </function>
    </problem>
    <problem id="q-8" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^64">
        <function class="functions.discrete.Queens" boardSize="8" />
      </function>
    </problem>
    <problem id="kt-4" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^48">
```

```xml
        <function class="functions.discrete.RepairingKnightsTour" boardSize="4" />
      </function>
    </problem>
    <problem id="kt-5" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^75">
        <function class="functions.discrete.RepairingKnightsTour" boardSize="5" />
      </function>
    </problem>
    <problem id="kt-6" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^108">
        <function class="functions.discrete.RepairingKnightsTour" boardSize="6" />
      </function>
    </problem>
    <problem id="kt-7" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^147">
        <function class="functions.discrete.RepairingKnightsTour" boardSize="7" />
      </function>
    </problem>
    <problem id="kt-8" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^192">
        <function class="functions.discrete.RepairingKnightsTour" boardSize="8" />
      </function>
    </problem>
    <problem id="ipd" class="problem.FunctionMaximisationProblem">
        <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^64">
        <function class="functions.discrete.IteratedPrisonersDilemma" domain="B^64" />
        </function>
    </problem>
  </problems>
  <measurements id="measurements" class="simulator.MeasurementSuite" samples="30" resolution="1">
    <addMeasurement class="measurement.single.Fitness"/>
    <addMeasurement class="measurement.single.CollectiveFitness" />
  <!-- <addMeasurement class="measurement.single.Solution"/>-->
  </measurements>
  <simulations>
    <simulation>
      <algorithm idref="de"/>
      <problem idref="mdk-10"/>
      <measurements idref="measurements" file="data/masters/binary/de/binde/binde-mdk-10.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="de"/>
      <problem idref="mdk-20"/>
      <measurements idref="measurements" file="data/masters/binary/de/binde/binde-mdk-20.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="de"/>
      <problem idref="kc-4"/>
      <measurements idref="measurements" file="data/masters/binary/de/binde/binde-kc-4.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="de"/>
      <problem idref="kc-5"/>
      <measurements idref="measurements" file="data/masters/binary/de/binde/binde-kc-5.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="de"/>
      <problem idref="kc-6"/>
      <measurements idref="measurements" file="data/masters/binary/de/binde/binde-kc-6.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="de"/>
      <problem idref="kc-7"/>
      <measurements idref="measurements" file="data/masters/binary/de/binde/binde-kc-7.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="de"/>
      <problem idref="kc-8"/>
      <measurements idref="measurements" file="data/masters/binary/de/binde/binde-kc-8.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="de"/>
      <problem idref="q-4"/>
      <measurements idref="measurements" file="data/masters/binary/de/binde/binde-q-4.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="de"/>
      <problem idref="q-5"/>
      <measurements idref="measurements" file="data/masters/binary/de/binde/binde-q-5.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="de"/>
      <problem idref="q-6"/>
      <measurements idref="measurements" file="data/masters/binary/de/binde/binde-q-6.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="de"/>
      <problem idref="q-7"/>
      <measurements idref="measurements" file="data/masters/binary/de/binde/binde-q-7.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="de"/>
      <problem idref="q-8"/>
      <measurements idref="measurements" file="data/masters/binary/de/binde/binde-q-8.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="de" />
      <problem idref="kt-4" />
      <measurements idref="measurements" file="data/masters/binary/de/binde/binde-kt-4.txt" />
    </simulation>
        <simulation>
      <algorithm idref="de" />
      <problem idref="kt-5" />
      <measurements idref="measurements" file="data/masters/binary/de/binde/binde-kt-5.txt" />
    </simulation>
        <simulation>
      <algorithm idref="de" />
      <problem idref="kt-6" />
      <measurements idref="measurements" file="data/masters/binary/de/binde/binde-kt-6.txt" />
    </simulation>
        <simulation>
      <algorithm idref="de" />
      <problem idref="kt-7" />
      <measurements idref="measurements" file="data/masters/binary/de/binde/binde-kt-7.txt" />
    </simulation>
        <simulation>
      <algorithm idref="de" />
      <problem idref="kt-8" />
      <measurements idref="measurements" file="data/masters/binary/de/binde/binde-kt-8.txt" />
    </simulation>
    <simulation>
      <algorithm idref="de" />
      <problem idref="ipd" />
      <measurements idref="measurements" file="data/masters/binary/de/binde/binde-ipd.txt" />
    </simulation>
  </simulations>
</simulator>
```

Normalized differential evolution

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE simulator [
 <!ATTLIST algorithm id ID #IMPLIED>
<!ATTLIST problem id ID #IMPLIED>
<!ATTLIST measurements id ID #IMPLIED>
]>
<simulator>
  <algorithms>
    <algorithm id="de" class="ec.EC">
      <initialisationStrategy class="algorithm.initialisation.ClonedPopulationInitialisationStrategy">
        <entityType class="ec.Individual">
    <fitnessCalculator class="util.calculator.NormalizedFitnessCalculator" />
  </entityType>
  <entityNumber value="40"/>
      </initialisationStrategy>
      <iterationStrategy class="ec.iterationstrategies.DifferentialEvolutionIterationStrategy">
  <!--<trialVectorCreationStrategy class="entity.operators.creation.RandCreationStrategy">
    <scaleParameter class="controlparameter.ConstantControlParameter" parameter="1.0" />
  </trialVectorCreationStrategy>
  <crossoverStrategy class="entity.operators.crossover.DifferentialEvolutionBinomialCrossover">
    <crossoverProbability class="controlparameter.ConstantControlParameter" parameter="0.25"/>
  </crossoverStrategy>-->
  <boundaryConstraint class="problem.boundaryconstraint.ClampingBoundaryConstraint"/>
      </iterationStrategy>
      <addStoppingCondition class="stoppingcondition.MaximumIterations" maximumIterations="10000"/>
    </algorithm>
  </algorithms>
  <problems>
    <problem id="mdk-10" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^10">
          <function class="functions.discrete.KnapSack" domain="B^10">
    <capacity value="269" />
    <numberOfObjects value="10" />
    <weight value="95,4,60,32,23,72,80,62,64,46" />
    <value value="55,10,47,5,4,50,8,61,85,87" />
        </function>
      </function>
    </problem>
    <problem id="mdk-20" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^20">
        <function class="functions.discrete.KnapSack" domain="B^20">
    <capacity value="878" />
    <numberOfObjects value="20" />
    <weight value="92,4,43,83,84,68,92,82,6,44,32,18,56,83,25,96,70,48,14,58" />
    <value value="44,46,90,72,91,40,75,35,8,54,78,40,77,15,61,17,75,29,75,63" />
  </function>
      </function>
    </problem>
    <problem id="kc-4" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^16">
        <function class="functions.discrete.KnightsCoverage" boardSize="4" />
      </function>
    </problem>
    <problem id="kc-5" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^25">
        <function class="functions.discrete.KnightsCoverage" boardSize="5" />
      </function>
    </problem>
    <problem id="kc-6" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^36">
        <function class="functions.discrete.KnightsCoverage" boardSize="6" />
      </function>
    </problem>
    <problem id="kc-7" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^49">
        <function class="functions.discrete.KnightsCoverage" boardSize="7" />
      </function>
    </problem>
    <problem id="kc-8" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^64">
        <function class="functions.discrete.KnightsCoverage" boardSize="8" />
      </function>
    </problem>
    <problem id="q-4" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^16">
        <function class="functions.discrete.Queens" boardSize="4" />
      </function>
    </problem>
    <problem id="q-5" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^25">
        <function class="functions.discrete.Queens" boardSize="5" />
      </function>
    </problem>
    <problem id="q-6" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^36">
        <function class="functions.discrete.Queens" boardSize="6" />
      </function>
    </problem>
    <problem id="q-7" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^49">
        <function class="functions.discrete.Queens" boardSize="7" />
      </function>
    </problem>
    <problem id="q-8" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^64">
        <function class="functions.discrete.Queens" boardSize="8" />
      </function>
    </problem>
    <problem id="kt-4" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^48">
        <function class="functions.discrete.RepairingKnightsTour" boardSize="4" />
      </function>
    </problem>
    <problem id="kt-5" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^75">
        <function class="functions.discrete.RepairingKnightsTour" boardSize="5" />
      </function>
    </problem>
    <problem id="kt-6" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^108">
        <function class="functions.discrete.RepairingKnightsTour" boardSize="6" />
      </function>
    </problem>
    <problem id="kt-7" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^147">
        <function class="functions.discrete.RepairingKnightsTour" boardSize="7" />
      </function>
    </problem>
    <problem id="kt-8" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^192">
        <function class="functions.discrete.RepairingKnightsTour" boardSize="8" />
      </function>
    </problem>
    <problem id="ipd" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^64">
        <function class="functions.discrete.IteratedPrisonersDilemma" domain="B^64" />
      </function>
    </problem>
  </problems>
  <measurements id="measurements" class="simulator.MeasurementSuite" samples="30" resolution="1">
    <addMeasurement class="measurement.single.Fitness"/>
    <addMeasurement class="measurement.single.CollectiveFitness" />
  <!-- <addMeasurement class="measurement.single.Solution"/>-->
  </measurements>
```

```
<simulations>
  <simulation>
    <algorithm idref="de"/>
    <problem idref="mdk-10"/>
    <measurements idref="measurements" file="data/masters/binary/de/normde/normde-mdk-10.txt"/>
  </simulation>
  <simulation>
    <algorithm idref="de"/>
    <problem idref="mdk-20"/>
    <measurements idref="measurements" file="data/masters/binary/de/normde/normde-mdk-20.txt"/>
  </simulation>
  <simulation>
    <algorithm idref="de"/>
    <problem idref="kc-4"/>
    <measurements idref="measurements" file="data/masters/binary/de/normde/normde-kc-4.txt"/>
  </simulation>
  <simulation>
    <algorithm idref="de"/>
    <problem idref="kc-5"/>
    <measurements idref="measurements" file="data/masters/binary/de/normde/normde-kc-5.txt"/>
  </simulation>
  <simulation>
    <algorithm idref="de"/>
    <problem idref="kc-6"/>
    <measurements idref="measurements" file="data/masters/binary/de/normde/normde-kc-6.txt"/>
  </simulation>
  <simulation>
    <algorithm idref="de"/>
    <problem idref="kc-7"/>
    <measurements idref="measurements" file="data/masters/binary/de/normde/normde-kc-7.txt"/>
  </simulation>
  <simulation>
    <algorithm idref="de"/>
    <problem idref="kc-8"/>
    <measurements idref="measurements" file="data/masters/binary/de/normde/normde-kc-8.txt"/>
  </simulation>
  <simulation>
    <algorithm idref="de"/>
    <problem idref="q-4"/>
    <measurements idref="measurements" file="data/masters/binary/de/normde/normde-q-4.txt"/>
  </simulation>
  <simulation>
    <algorithm idref="de"/>
    <problem idref="q-5"/>
    <measurements idref="measurements" file="data/masters/binary/de/normde/normde-q-5.txt"/>
  </simulation>
  <simulation>
    <algorithm idref="de"/>
    <problem idref="q-6"/>
    <measurements idref="measurements" file="data/masters/binary/de/normde/normde-q-6.txt"/>
  </simulation>
  <simulation>
    <algorithm idref="de"/>
    <problem idref="q-7"/>
    <measurements idref="measurements" file="data/masters/binary/de/normde/normde-q-7.txt"/>
  </simulation>
  <simulation>
    <algorithm idref="de"/>
    <problem idref="q-8"/>
    <measurements idref="measurements" file="data/masters/binary/de/normde/normde-q-8.txt"/>
  </simulation>
  <simulation>
    <algorithm idref="de" />
    <problem idref="kt-4" />
    <measurements idref="measurements" file="data/masters/binary/de/normde/normde-kt-4.txt" />
  </simulation>
      <simulation>
      <algorithm idref="de" />
      <problem idref="kt-5" />
      <measurements idref="measurements" file="data/masters/binary/de/normde/normde-kt-5.txt" />
    </simulation>
        <simulation>
      <algorithm idref="de" />
      <problem idref="kt-6" />
      <measurements idref="measurements" file="data/masters/binary/de/normde/normde-kt-6.txt" />
    </simulation>
        <simulation>
      <algorithm idref="de" />
      <problem idref="kt-7" />
      <measurements idref="measurements" file="data/masters/binary/de/normde/normde-kt-7.txt" />
    </simulation>
    <simulation>
      <algorithm idref="de" />
      <problem idref="kt-8" />
      <measurements idref="measurements" file="data/masters/binary/de/normde/normde-kt-8.txt" />
    </simulation>
    <simulation>
      <algorithm idref="de" />
      <problem idref="ipd" />
      <measurements idref="measurements" file="data/masters/binary/de/normde/normde-ipd.txt" />
    </simulation>
  </simulations>
</simulator>
```

E.4 Particle swarm optimization

E.4.1 Continuous simulations

Angle modulated particle swarm optimization

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE simulator [
<!ATTLIST algorithm id ID #IMPLIED>
<!ATTLIST problem id ID #IMPLIED>
<!ATTLIST measurements id ID #IMPLIED>
]>
<simulator>
  <algorithms>
    <algorithm id="pso" class="pso.PSO">
      <initialisationStrategy
       class="algorithm.initialisation.ClonedPopulationInitialisationStrategy">
         <entityType class="pso.particle.StandardParticle" />
  <entityNumber value="40"/>
      </initialisationStrategy>
      <iterationStrategy
       class="pso.iterationstrategies.SynchronousIterationStrategy">
  <boundaryConstraint
   class="problem.boundaryconstraint.ClampingBoundaryConstraint"/>
      </iterationStrategy>
      <addStoppingCondition class="stoppingcondition.MaximumIterations" maximumIterations="10000"/>
    </algorithm>
  </algorithms>
  <problems>
    <problem id="spherical" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2" domain="R
           (-1.0,1.0)^4">
```

```
        <function class="functions.continuous.unconstrained.Spherical" domain="R(-5.12,5.12)^30"/>
      </function>
    </problem>
    <problem id="rosenbrock2d" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="3" domain="R
          (-1.0,1.0)^4">
        <function class="functions.continuous.unconstrained.Rosenbrock" domain="R(-2.048,2.048)^2"/>
      </function>
    </problem>
    <problem id="step" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2" domain="R
          (-1.0,1.0)^4">
        <function class="functions.continuous.Step" domain="R(-5.12,5.12)^5"/>
      </function>
    </problem>
    <problem id="quartic" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2" domain="R
          (-1.0,1.0)^4">
        <function class="functions.continuous.Quartic" domain="R(-1.28,1.28)^30"/>
      </function>
    </problem>
    <problem id="foxholes" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2" domain="R
          (-1.0,1.0)^4">
        <function class="functions.continuous.Foxholes" domain="R(-65536.0,65536.0)^2"/>
      </function>
    </problem>
    <problem id="schaffer6" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2" domain="R
          (-1.0,1.0)^4">
        <function class="functions.continuous.Schaffer6" domain="R(-100.0,100.0)^2"/>
      </function>
    </problem>
   <problem id="griewank" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2" domain="R
          (-1.0,1.0)^4">
        <function class="functions.continuous.unconstrained.Griewank" domain="R(-300.0,300.0)^30"/>
      </function>
    </problem>
    <problem id="ackley" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2" domain="R
          (-1.0,1.0)^4">
        <function class="functions.continuous.unconstrained.Ackley" domain="R(-30.0,30.0)^30"/>
      </function>
    </problem>
    <problem id="rosenbrock" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="3" domain="R
          (-1.0,1.0)^4">
        <function class="functions.continuous.unconstrained.Rosenbrock" domain="R(-2.048,2.048)^30"/>
      </function>
    </problem>
    <problem id="rastrigin" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2" domain="R
          (-1.0,1.0)^4">
        <function class="functions.continuous.unconstrained.Rastrigin" domain="R(-5.12,5.12)^30"/>
      </function>
    </problem>
  </problems>
  <measurements id="measurements" class="simulator.MeasurementSuite" samples="1" resolution="1">
    <addMeasurement class="measurement.single.Fitness"/>
  <!-- <addMeasurement class="measurement.single.Solution"/>-->
  </measurements>
  <simulations>
  <simulation>
      <algorithm idref="pso"/>
      <problem idref="spherical"/>
      <measurements idref="measurements" file="data/masters/pso/ampso-spherical.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="pso"/>
      <problem idref="rosenbrock2d"/>
      <measurements idref="measurements" file="data/masters/pso/ampso-rosenbrock2d.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="pso"/>
      <problem idref="step"/>
      <measurements idref="measurements" file="data/masters/pso/ampso/ampso-step.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="pso"/>
      <problem idref="quartic"/>
      <measurements idref="measurements" file="data/masters/pso/ampso-quartic.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="pso"/>
      <problem idref="foxholes"/>
      <measurements idref="measurements" file="data/masters/pso/ampso-foxholes.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="pso"/>
      <problem idref="schaffer6"/>
      <measurements idref="measurements" file="data/masters/pso/ampso-schaffer6.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="pso"/>
      <problem idref="griewank"/>
      <measurements idref="measurements" file="data/masters/pso/ampso-griewank.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="pso"/>
      <problem idref="ackley"/>
      <measurements idref="measurements" file="data/masters/pso/ampso-ackley.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="pso"/>
      <problem idref="rosenbrock"/>
      <measurements idref="measurements" file="data/masters/pso/ampso-rosenbrock.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="pso"/>
      <problem idref="rastrigin"/>
      <measurements idref="measurements" file="data/masters/pso/ampso-rastrigin.txt"/>
    </simulation>
  </simulations>
</simulator>
```

Binary particle swarm optimization

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE simulator [
 <!ATTLIST algorithm id ID #IMPLIED>
<!ATTLIST problem id ID #IMPLIED>
<!ATTLIST measurements id ID #IMPLIED>
]>
<simulator>
  <algorithms>
    <algorithm id="binpso" class="pso.PSO">
      <initialisationStrategy
       class="algorithm.initialisation.ClonedPopulationInitialisationStrategy">
        <entityType class="pso.particle.StandardParticle">
    <positionUpdateStrategy class="pso.positionupdatestrategies.BinaryPositionUpdateStrategy"/>
    <velocityUpdateStrategy
     class="pso.velocityupdatestrategies.StandardVelocityUpdate">
      <vMax class="controlparameter.ConstantControlParameter"
       parameter="4.0" />
    </velocityUpdateStrategy>
  </entityType>
  <entityNumber value="40"/>
      </initialisationStrategy>
      <iterationStrategy
       class="pso.iterationstrategies.SynchronousIterationStrategy">
  <boundaryConstraint
   class="problem.boundaryconstraint.ClampingBoundaryConstraint"/>
      </iterationStrategy>
      <addStoppingCondition class="stoppingcondition.MaximumIterations" maximumIterations="10000"/>
    </algorithm>
  </algorithms>
  <problems>
    <problem id="spherical" class="problem.FunctionMinimisationProblem">
      <function class="functions.discrete.BinaryAdapter" precision="2" bitsPerDimension="10" domain="B
          ^300">
        <function class="functions.continuous.unconstrained.Spherical" domain="R(-5.12,5.12)^30"/>
      </function>
    </problem>
    <problem id="rosenbrock2d" class="problem.FunctionMinimisationProblem">
      <function class="functions.discrete.BinaryAdapter" precision="3" bitsPerDimension="12" domain="B
          ^24">
        <function class="functions.continuous.unconstrained.Rosenbrock" domain="R(-2.048,2.048)^2"/>
      </function>
    </problem>
    <problem id="step" class="problem.FunctionMinimisationProblem">
      <function class="functions.discrete.BinaryAdapter" precision="2" bitsPerDimension="10" domain="B
          ^50">
        <function class="functions.continuous.Step" domain="R(-5.12,5.12)^5"/>
      </function>
    </problem>
    <problem id="quartic" class="problem.FunctionMinimisationProblem">
      <function class="functions.discrete.BinaryAdapter" precision="2" bitsPerDimension="8" domain="B
          ^240">
        <function class="functions.continuous.Quartic" domain="R(-1.28,1.28)^30"/>
      </function>
    </problem>
    <problem id="foxholes" class="problem.FunctionMinimisationProblem">
      <function class="functions.discrete.BinaryAdapter" precision="0" bitsPerDimension="17" domain="B
          ^34">
        <function class="functions.continuous.Foxholes" domain="R(-65536.0,65536.0)^2"/>
      </function>
    </problem>
    <problem id="schaffer6" class="problem.FunctionMinimisationProblem">
      <function class="functions.discrete.BinaryAdapter" precision="0" bitsPerDimension="8" domain="B
          ^16">
        <function class="functions.continuous.Schaffer6" domain="R(-100.0,100.0)^2"/>
      </function>
    </problem>
   <problem id="griewank" class="problem.FunctionMinimisationProblem">
      <function class="functions.discrete.BinaryAdapter" precision="0" bitsPerDimension="10" domain="B
          ^300">
        <function class="functions.continuous.unconstrained.Griewank" domain="R(-300.0,300.0)^30"/>
      </function>
    </problem>
    <problem id="ackley" class="problem.FunctionMinimisationProblem">
      <function class="functions.discrete.BinaryAdapter" precision="0" bitsPerDimension="6" domain="B
          ^180">
        <function class="functions.continuous.unconstrained.Ackley" domain="R(-30.0,30.0)^30"/>
      </function>
    </problem>
    <problem id="rosenbrock" class="problem.FunctionMinimisationProblem">
```

```
      <function class="functions.discrete.BinaryAdapter" precision="3" bitsPerDimension="12" domain="B
          ^360">
        <function class="functions.continuous.unconstrained.Rosenbrock" domain="R(-2.048,2.048)^30"/>
      </function>
    </problem>
    <problem id="rastrigin" class="problem.FunctionMinimisationProblem">
      <function class="functions.discrete.BinaryAdapter" precision="2" bitsPerDimension="10" domain="B
          ^300">
        <function class="functions.continuous.unconstrained.Rastrigin" domain="R(-5.12,5.12)^30"/>
      </function>
    </problem>
  </problems>
  <measurements id="measurements" class="simulator.MeasurementSuite" samples="1" resolution="1">
    <addMeasurement class="measurement.single.Fitness"/>
  </measurements>
  <simulations>
  <simulation>
      <algorithm idref="binpso"/>
      <problem idref="spherical"/>
      <measurements idref="measurements" file="data/masters/binpso/binpso-spherical.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binpso"/>
      <problem idref="rosenbrock2d"/>
      <measurements idref="measurements" file="data/masters/binpso/binpso-rosenbrock2d.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binpso"/>
      <problem idref="step"/>
      <measurements idref="measurements" file="data/masters/binpso/binpso-step.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binpso"/>
      <problem idref="quartic"/>
      <measurements idref="measurements" file="data/masters/binpso/binpso-quartic.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binpso"/>
      <problem idref="foxholes"/>
      <measurements idref="measurements" file="data/masters/binpso/binpso-foxholes.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binpso"/>
      <problem idref="schaffer6"/>
      <measurements idref="measurements" file="data/masters/binpso/binpso-schaffer6.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binpso"/>
      <problem idref="griewank"/>
      <measurements idref="measurements" file="data/masters/binpso/binpso-griewank.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binpso"/>
      <problem idref="ackley"/>
      <measurements idref="measurements" file="data/masters/binpso/binpso-ackley.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binpso"/>
      <problem idref="rosenbrock"/>
      <measurements idref="measurements" file="data/masters/binpso/binpso-rosenbrock.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binpso"/>
      <problem idref="rastrigin"/>
      <measurements idref="measurements" file="data/masters/binpso/binpso-rastrigin.txt"/>
    </simulation>
  </simulations>
```

```
</simulator>
```

E.4.2 Binary simulations

Angle modulated particle swarm optimization

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE simulator [
 <!ATTLIST algorithm id ID #IMPLIED>
<!ATTLIST problem id ID #IMPLIED>
<!ATTLIST measurements id ID #IMPLIED>
]>
<simulator>
  <algorithms>
    <algorithm id="pso" class="pso.PSO">
      <initialisationStrategy
       class="algorithm.initialisation.ClonedPopulationInitialisationStrategy">
        <entityType class="pso.particle.StandardParticle" />
  <entityNumber value="40"/>
      </initialisationStrategy>
      <iterationStrategy
       class="pso.iterationstrategies.SynchronousIterationStrategy">
  <boundaryConstraint
   class="problem.boundaryconstraint.ClampingBoundaryConstraint"/>
      </iterationStrategy>
      <addStoppingCondition class="stoppingcondition.MaximumIterations" maximumIterations="10000"/>
    </algorithm>
  </algorithms>
  <problems>
    <problem id="mdk-10" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.KnapSack" domain="B^10">
    <capacity value="269" />
    <numberOfObjects value="10" />
    <weight value="95,4,60,32,23,72,80,62,64,46" />
    <value value="55,10,47,5,4,50,8,61,85,87" />
  </function>
      </function>
    </problem>
    <problem id="mdk-20" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.KnapSack" domain="B^20">
    <capacity value="878" />
    <numberOfObjects value="20" />
    <weight value="92,4,43,83,84,68,92,82,6,44,32,18,56,83,25,96,70,48,14,58" />
    <value value="44,46,90,72,91,40,75,35,8,54,78,40,77,15,61,17,75,29,75,63" />
  </function>
      </function>
    </problem>
    <problem id="kc-4" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.KnightsCoverage" boardSize="4" />
      </function>
    </problem>
    <problem id="kc-5" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.KnightsCoverage" boardSize="5" />
      </function>
    </problem>
    <problem id="kc-6" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.KnightsCoverage" boardSize="6" />
      </function>
    </problem>
    <problem id="kc-7" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.KnightsCoverage" boardSize="7" />
      </function>
    </problem>
    <problem id="kc-8" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.KnightsCoverage" boardSize="8" />
      </function>
    </problem>
    <problem id="q-4" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.Queens" boardSize="4" />
      </function>
    </problem>
    <problem id="q-5" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.Queens" boardSize="5" />
      </function>
    </problem>
    <problem id="q-6" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.Queens" boardSize="6" />
      </function>
    </problem>
    <problem id="q-7" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.Queens" boardSize="7" />
      </function>
    </problem>
    <problem id="q-8" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.Queens" boardSize="8" />
      </function>
    </problem>
    <problem id="kt-4" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.RepairingKnightsTour" boardSize="4" />
      </function>
    </problem>
    <problem id="kt-5" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.RepairingKnightsTour" boardSize="5" />
      </function>
    </problem>
    <problem id="kt-6" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.RepairingKnightsTour" boardSize="6" />
      </function>
    </problem>
    <problem id="kt-7" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.RepairingKnightsTour" boardSize="7" />
      </function>
    </problem>
    <problem id="kt-8" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.RepairingKnightsTour" boardSize="8" />
      </function>
    </problem>
    <problem id="ipd" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.IteratedPrisonersDilemma" domain="B^64" />
      </function>
    </problem>
```

```
  </problems>
  <measurements id="measurements" class="simulator.MeasurementSuite" samples="30" resolution="1">
    <addMeasurement class="measurement.single.Fitness"/>
    <addMeasurement class="measurement.single.CollectiveFitness" />
  <!-- <addMeasurement class="measurement.single.Solution"/>-->
  </measurements>
  <simulations>
    <simulation>
      <algorithm idref="pso"/>
      <problem idref="mdk-10"/>
      <measurements idref="measurements" file="data/masters/binary/pso/ampso/ampso-mdk-10.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="pso"/>
      <problem idref="mdk-20"/>
      <measurements idref="measurements" file="data/masters/binary/pso/ampso/ampso-mdk-20.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="pso"/>
      <problem idref="kc-4"/>
      <measurements idref="measurements" file="data/masters/binary/pso/ampso/ampso-kc-4.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="pso"/>
      <problem idref="kc-5"/>
      <measurements idref="measurements" file="data/masters/binary/pso/ampso/ampso-kc-5.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="pso"/>
      <problem idref="kc-6"/>
      <measurements idref="measurements" file="data/masters/binary/pso/ampso/ampso-kc-6.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="pso"/>
      <problem idref="kc-7"/>
      <measurements idref="measurements" file="data/masters/binary/pso/ampso/ampso-kc-7.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="pso"/>
      <problem idref="kc-8"/>
      <measurements idref="measurements" file="data/masters/binary/pso/ampso/ampso-kc-8.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="pso"/>
      <problem idref="q-4"/>
      <measurements idref="measurements" file="data/masters/binary/pso/ampso/ampso-q-4.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="pso"/>
      <problem idref="q-5"/>
      <measurements idref="measurements" file="data/masters/binary/pso/ampso/ampso-q-5.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="pso"/>
      <problem idref="q-6"/>
      <measurements idref="measurements" file="data/masters/binary/pso/ampso/ampso-q-6.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="pso"/>
      <problem idref="q-7"/>
      <measurements idref="measurements" file="data/masters/binary/pso/ampso/ampso-q-7.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="pso"/>
      <problem idref="q-8"/>
      <measurements idref="measurements" file="data/masters/binary/pso/ampso/ampso-q-8.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="pso" />
      <problem idref="kt-4" />
      <measurements idref="measurements" file="data/masters/binary/pso/ampso/ampso-kt-4.txt" />
    </simulation>
    <simulation>
      <algorithm idref="pso" />
      <problem idref="kt-5" />
      <measurements idref="measurements" file="data/masters/binary/pso/ampso/ampso-kt-5.txt" />
    </simulation>
    <simulation>
      <algorithm idref="pso" />
      <problem idref="kt-6" />
      <measurements idref="measurements" file="data/masters/binary/pso/ampso/ampso-kt-6.txt" />
    </simulation>
    <simulation>
      <algorithm idref="pso" />
      <problem idref="kt-7" />
      <measurements idref="measurements" file="data/masters/binary/pso/ampso/ampso-kt-7.txt" />
    </simulation>
    <simulation>
      <algorithm idref="pso" />
      <problem idref="kt-8" />
      <measurements idref="measurements" file="data/masters/binary/pso/ampso/ampso-kt-8.txt" />
    </simulation>
    <simulation>
      <algorithm idref="pso" />
      <problem idref="ipd" />
      <measurements idref="measurements" file="data/masters/binary/pso/ampso/ampso-ipd.txt" />
    </simulation>
  </simulations>
</simulator>
```

Binary particle swarm optimization

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE simulator [
 <!ATTLIST algorithm id ID #IMPLIED>
<!ATTLIST problem id ID #IMPLIED>
<!ATTLIST measurements id ID #IMPLIED>
]>
<simulator>
  <algorithms>
    <algorithm id="binpso" class="pso.PSO">
      <initialisationStrategy
       class="algorithm.initialisation.ClonedPopulationInitialisationStrategy">
        <entityType class="pso.particle.StandardParticle">
    <positionUpdateStrategy class="pso.positionupdatestrategies.BinaryPositionUpdateStrategy"/>
    <velocityUpdateStrategy
     class="pso.velocityupdatestrategies.StandardVelocityUpdate">
      <vMax class="controlparameter.ConstantControlParameter"
       parameter="4.0" />
    </velocityUpdateStrategy>
  </entityType>
  <entityNumber value="40"/>
      </initialisationStrategy>
      <iterationStrategy
       class="pso.iterationstrategies.SynchronousIterationStrategy">
  <boundaryConstraint
   class="problem.boundaryconstraint.ClampingBoundaryConstraint"/>
      </iterationStrategy>
      <addStoppingCondition class="stoppingcondition.MaximumIterations" maximumIterations="10000"/>
    </algorithm>
```

```
</algorithms>
<problems>
  <problem id="mdk−10" class="problem.FunctionMaximisationProblem">
      <function class="functions.discrete.KnapSack" domain="R(−1000.0,1000.0)^10">
  <capacity value="269" />
  <numberOfObjects value="10" />
  <weight value="95,4,60,32,23,72,80,62,64,46" />
  <value value="55,10,47,5,4,50,8,61,85,87" />
</function>
  </problem>
  <problem id="mdk−20" class="problem.FunctionMaximisationProblem">
      <function class="functions.discrete.KnapSack" domain="R(−1000.0,1000.0)^20">
  <capacity value="878" />
  <numberOfObjects value="20" />
  <weight value="92,4,43,83,84,68,92,82,6,44,32,18,56,83,25,96,70,48,14,58" />
  <value value="44,46,90,72,91,40,75,35,8,54,78,40,77,15,61,17,75,29,75,63" />
</function>
  </problem>
  <problem id="kc−4" class="problem.FunctionMinimisationProblem">
      <function class="functions.discrete.KnightsCoverage" boardSize="4" domain="R(−1000.0,1000.0)
          ^16"/>
  </problem>
  <problem id="kc−5" class="problem.FunctionMinimisationProblem">
      <function class="functions.discrete.KnightsCoverage" boardSize="5" domain="R(−1000.0,1000.0)
          ^25"/>
  </problem>
  <problem id="kc−6" class="problem.FunctionMinimisationProblem">
      <function class="functions.discrete.KnightsCoverage" boardSize="6" domain="R(−1000.0,1000.0)
          ^36"/>
  </problem>
  <problem id="kc−7" class="problem.FunctionMinimisationProblem">
      <function class="functions.discrete.KnightsCoverage" boardSize="7" domain="R(−1000.0,1000.0)
          ^49"/>
  </problem>
  <problem id="kc−8" class="problem.FunctionMinimisationProblem">
      <function class="functions.discrete.KnightsCoverage" boardSize="8" domain="R(−1000.0,1000.0)
          ^64"/>
  </problem>
  <problem id="q−4" class="problem.FunctionMinimisationProblem">
      <function class="functions.discrete.Queens" boardSize="4" domain="R(−1000.0,1000.0)^16" />
  </problem>
  <problem id="q−5" class="problem.FunctionMinimisationProblem">
      <function class="functions.discrete.Queens" boardSize="5" domain="R(−1000.0,1000.0)^25" />
  </problem>
  <problem id="q−6" class="problem.FunctionMinimisationProblem">
      <function class="functions.discrete.Queens" boardSize="6" domain="R(−1000.0,1000.0)^36" />
  </problem>
  <problem id="q−7" class="problem.FunctionMinimisationProblem">
      <function class="functions.discrete.Queens" boardSize="7" domain="R(−1000.0,1000.0)^49" />
  </problem>
  <problem id="q−8" class="problem.FunctionMinimisationProblem">
      <function class="functions.discrete.Queens" boardSize="8" domain="R(−1000.0,1000.0)^64" />
  </problem>
  <problem id="kt−4" class="problem.FunctionMaximisationProblem">
      <function class="functions.discrete.RepairingKnightsTour" boardSize="4" domain="R
          (−1000.0,1000.0)^48" />
  </problem>
  <problem id="kt−5" class="problem.FunctionMaximisationProblem">
      <function class="functions.discrete.RepairingKnightsTour" boardSize="5" domain="R
          (−1000.0,1000.0)^75" />
  </problem>
  <problem id="kt−6" class="problem.FunctionMaximisationProblem">
      <function class="functions.discrete.RepairingKnightsTour" boardSize="6" domain="R
          (−1000.0,1000.0)^108" />
  </problem>
  <problem id="kt−7" class="problem.FunctionMaximisationProblem">
```

```
        <function class="functions.discrete.RepairingKnightsTour" boardSize="7" domain="R
            (-1000.0,1000.0)^147" />
    </problem>
    <problem id="kt-8" class="problem.FunctionMaximisationProblem">
        <function class="functions.discrete.RepairingKnightsTour" boardSize="8" domain="R
            (-1000.0,1000.0)^192" />
    </problem>
    <problem id="ipd" class="problem.FunctionMaximisationProblem">
        <function class="functions.discrete.IteratedPrisonersDilemma" domain="B^64" />
    </problem>
  </problems>
  <measurements id="measurements" class="simulator.MeasurementSuite" samples="30" resolution="1">
    <addMeasurement class="measurement.single.Fitness"/>
    <addMeasurement class="measurement.single.CollectiveFitness" />
  </measurements>
  <simulations>
    <simulation>
      <algorithm idref="binpso"/>
      <problem idref="mdk-10"/>
      <measurements idref="measurements" file="data/masters/binary/pso/binpso/binpso-mdk-10.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binpso"/>
      <problem idref="mdk-20"/>
      <measurements idref="measurements" file="data/masters/binary/pso/binpso/binpso-mdk-20.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binpso"/>
      <problem idref="kc-4"/>
      <measurements idref="measurements" file="data/masters/binary/pso/binpso/binpso-kc-4.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binpso"/>
      <problem idref="kc-5"/>
      <measurements idref="measurements" file="data/masters/binary/pso/binpso/binpso-kc-5.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binpso"/>
      <problem idref="kc-6"/>
      <measurements idref="measurements" file="data/masters/binary/pso/binpso/binpso-kc-6.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binpso"/>
      <problem idref="kc-7"/>
      <measurements idref="measurements" file="data/masters/binary/pso/binpso/binpso-kc-7.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binpso"/>
      <problem idref="kc-8"/>
      <measurements idref="measurements" file="data/masters/binary/pso/binpso/binpso-kc-8.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binpso"/>
      <problem idref="q-4"/>
      <measurements idref="measurements" file="data/masters/binary/pso/binpso/binpso-q-4.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binpso"/>
      <problem idref="q-5"/>
      <measurements idref="measurements" file="data/masters/binary/pso/binpso/binpso-q-5.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binpso"/>
      <problem idref="q-6"/>
      <measurements idref="measurements" file="data/masters/binary/pso/binpso/binpso-q-6.txt"/>
    </simulation>
    <simulation>
```

```
      <algorithm idref="binpso"/>
      <problem idref="q-7"/>
      <measurements idref="measurements" file="data/masters/binary/pso/binpso/binpso-q-7.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binpso"/>
      <problem idref="q-8"/>
      <measurements idref="measurements" file="data/masters/binary/pso/binpso/binpso-q-8.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="binpso" />
      <problem idref="kt-4" />
      <measurements idref="measurements" file="data/masters/binary/pso/binpso/binpso-kt-4.txt" />
    </simulation>
    <simulation>
      <algorithm idref="binpso" />
      <problem idref="kt-5" />
      <measurements idref="measurements" file="data/masters/binary/pso/binpso/binpso-kt-5.txt" />
    </simulation>
    <simulation>
      <algorithm idref="binpso" />
      <problem idref="kt-6" />
      <measurements idref="measurements" file="data/masters/binary/pso/binpso/binpso-kt-6.txt" />
    </simulation>
    <simulation>
      <algorithm idref="binpso" />
      <problem idref="kt-7" />
      <measurements idref="measurements" file="data/masters/binary/pso/binpso/binpso-kt-7.txt" />
    </simulation>
    <simulation>
      <algorithm idref="binpso" />
      <problem idref="kt-8" />
      <measurements idref="measurements" file="data/masters/binary/pso/binpso/binpso-kt-8.txt" />
    </simulation>
    <simulation>
      <algorithm idref="binpso" />
      <problem idref="ipd" />
      <measurements idref="measurements" file="data/masters/binary/pso/binpso/binpso-ipd.txt" />
    </simulation>
  </simulations>
</simulator>
```

E.5 Artificial bee colony

E.5.1 Continuous simulations

Angle modulated artificial bee colony

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE simulator [
 <!ATTLIST algorithm id ID #IMPLIED>
<!ATTLIST problem id ID #IMPLIED>
<!ATTLIST measurements id ID #IMPLIED>
]>
<simulator>
  <algorithms>
      <algorithm id="abc" class="boa.ABC">
        <initialisationStrategy class="algorithm.initialisation.ClonedPopulationInitialisationStrategy
            " entityNumber="40">
            <entityType class="boa.bee.WorkerBee"/>
        </initialisationStrategy>
        <addStoppingCondition class="stoppingcondition.MaximumIterations" maximumIterations="10000"/>
        <dancingSelectionStrategy class="entity.operators.selection.RouletteWheelSelectionStrategy"/>
        <workerBeePercentage class="controlparameter.ConstantControlParameter" parameter="0.5"/>
        <forageLimit class="controlparameter.ConstantControlParameter" parameter="500"/>
        <explorerBeeUpdateLimit class="controlparameter.ConstantControlParameter" parameter="1"/>
    </algorithm>
  </algorithms>
  <problems>
    <problem id="spherical" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2" domain="R
           (-1.0,1.0)^4">
        <function class="functions.continuous.unconstrained.Spherical" domain="R(-5.12,5.12)^30"/>
      </function>
    </problem>
    <problem id="rosenbrock2d" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="3" domain="R
           (-1.0,1.0)^4">
        <function class="functions.continuous.unconstrained.Rosenbrock" domain="R(-2.048,2.048)^2"/>
      </function>
    </problem>
    <problem id="step" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2" domain="R
           (-1.0,1.0)^4">
        <function class="functions.continuous.unconstrained.Step" domain="R(-5.12,5.12)^5"/>
      </function>
    </problem>
    <problem id="quartic" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2" domain="R
           (-1.0,1.0)^4">
        <function class="functions.continuous.Quartic" domain="R(-1.28,1.28)^30"/>
      </function>
    </problem>
    <problem id="foxholes" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2" domain="R
           (-1.0,1.0)^4">
        <function class="functions.continuous.Foxholes" domain="R(-65536.0,65536.0)^2"/>
      </function>
    </problem>
    <problem id="schaffer6" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2" domain="R
           (-1.0,1.0)^4">
        <function class="functions.continuous.Schaffer6" domain="R(-100.0,100.0)^2"/>
      </function>
    </problem>
   <problem id="griewank" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2" domain="R
           (-1.0,1.0)^4">
        <function class="functions.continuous.unconstrained.Griewank" domain="R(-300.0,300.0)^30"/>
      </function>
    </problem>
    <problem id="ackley" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2" domain="R
           (-1.0,1.0)^4">
        <function class="functions.continuous.unconstrained.Ackley" domain="R(-30.0,30.0)^30"/>
      </function>
    </problem>
    <problem id="rosenbrock" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="3" domain="R
           (-1.0,1.0)^4">
        <function class="functions.continuous.unconstrained.Rosenbrock" domain="R(-2.048,2.048)^30"/>
      </function>
    </problem>
    <problem id="rastrigin" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2" domain="R
           (-1.0,1.0)^4">
        <function class="functions.continuous.unconstrained.Rastrigin" domain="R(-5.12,5.12)^30"/>
      </function>
    </problem>
  </problems>
  <measurements id="measurements" class="simulator.MeasurementSuite" samples="1" resolution="1">
    <addMeasurement class="measurement.single.Fitness"/>
  <!-- <addMeasurement class="measurement.single.Solution"/>-->
  </measurements>
  <simulations>
  <simulation>
      <algorithm idref="abc"/>
      <problem idref="spherical"/>
      <measurements idref="measurements" file="data/masters/abc/amabc/amabc-spherical.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="abc"/>
      <problem idref="rosenbrock2d"/>
      <measurements idref="measurements" file="data/masters/abc/amabc/amabc-rosenbrock2d.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="abc"/>
      <problem idref="step"/>
      <measurements idref="measurements" file="data/masters/abc/amabc/amabc-step.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="abc"/>
      <problem idref="quartic"/>
      <measurements idref="measurements" file="data/masters/abc/amabc/amabc-quartic.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="abc"/>
      <problem idref="foxholes"/>
      <measurements idref="measurements" file="data/masters/abc/amabc/amabc-foxholes.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="abc"/>
      <problem idref="schaffer6"/>
      <measurements idref="measurements" file="data/masters/abc/amabc/amabc-schaffer6.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="abc"/>
      <problem idref="griewank"/>
      <measurements idref="measurements" file="data/masters/abc/amabc/amabc-griewank.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="abc"/>
      <problem idref="ackley"/>
      <measurements idref="measurements" file="data/masters/abc/amabc/amabc-ackley.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="abc"/>
      <problem idref="rosenbrock"/>
      <measurements idref="measurements" file="data/masters/abc/amabc/amabc-rosenbrock.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="abc"/>
      <problem idref="rastrigin"/>
      <measurements idref="measurements" file="data/masters/abc/amabc/amabc-rastrigin.txt"/>
    </simulation>
  </simulations>
</simulator>
```

Binary artificial bee colony

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE simulator [
<!ATTLIST algorithm id ID #IMPLIED>
<!ATTLIST problem id ID #IMPLIED>
<!ATTLIST measurements id ID #IMPLIED>
]>
<simulator>
  <algorithms>
    <algorithm id="abc" class="boa.ABC">
        <initialisationStrategy class="algorithm.initialisation.ClonedPopulationInitialisationStrategy" entityNumber="40">
            <entityType class="boa.bee.WorkerBee">
                <fitnessCalculator class="util.calculator.BinaryConvertedFitnessCalculator" />
            </entityType>
        </initialisationStrategy>
        <addStoppingCondition class="stoppingcondition.MaximumIterations" maximumIterations="10000"/>
        <dancingSelectionStrategy class="entity.operators.selection.RouletteWheelSelectionStrategy"/>
        <workerBeePercentage class="controlparameter.ConstantControlParameter" parameter="0.5"/>
        <forageLimit class="controlparameter.ConstantControlParameter" parameter="500"/>
        <explorerBeeUpdateLimit class="controlparameter.ConstantControlParameter" parameter="1"/>
    </algorithm>
  </algorithms>
  <problems>
    <problem id="spherical" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^300">
        <function class="functions.discrete.BinaryAdapter" precision="2" bitsPerDimension="10" domain="B^300">
          <function class="functions.continuous.unconstrained.Spherical" domain="R(-5.12,5.12)^30"/>
        </function>
      </function>
    </problem>
    <problem id="rosenbrock2d" class="problem.FunctionMinimisationProblem">
        <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^24">
      <function class="functions.discrete.BinaryAdapter" precision="3" bitsPerDimension="12" domain="B^24">
        <function class="functions.continuous.unconstrained.Rosenbrock" domain="R(-2.048,2.048)^2"/>
      </function>
      </function>
    </problem>
    <problem id="step" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^50">
        <function class="functions.discrete.BinaryAdapter" precision="2" bitsPerDimension="10" domain="B^50">
          <function class="functions.continuous.Step" domain="R(-5.12,5.12)^5"/>
        </function>
      </function>
    </problem>
    <problem id="quartic" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^240">
        <function class="functions.discrete.BinaryAdapter" precision="2" bitsPerDimension="8" domain="B^240">
          <function class="functions.continuous.Quartic" domain="R(-1.28,1.28)^30"/>
        </function>
      </function>
    </problem>
    <problem id="foxholes" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^34">
      <function class="functions.discrete.BinaryAdapter" precision="0" bitsPerDimension="17" domain="B^34">
        <function class="functions.continuous.Foxholes" domain="R(-65536.0,65536.0)^2"/>
      </function>
      </function>
    </problem>
    <problem id="schaffer6" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^16">
      <function class="functions.discrete.BinaryAdapter" precision="0" bitsPerDimension="8" domain="B
           ^16">
        <function class="functions.continuous.Schaffer6" domain="R(-100.0,100.0)^2"/>
      </function>
      </function>
    </problem>
   <problem id="griewank" class="problem.FunctionMinimisationProblem">
       <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^300">
      <function class="functions.discrete.BinaryAdapter" precision="0" bitsPerDimension="10" domain="B
           ^300">
        <function class="functions.continuous.unconstrained.Griewank" domain="R(-300.0,300.0)^30"/>
      </function>
      </function>
    </problem>
    <problem id="ackley" class="problem.FunctionMinimisationProblem">
        <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^180">
      <function class="functions.discrete.BinaryAdapter" precision="0" bitsPerDimension="6" domain="B
           ^180">
        <function class="functions.continuous.unconstrained.Ackley" domain="R(-30.0,30.0)^30"/>
      </function>
      </function>
    </problem>
    <problem id="rosenbrock" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^360">
      <function class="functions.discrete.BinaryAdapter" precision="3" bitsPerDimension="12" domain="B
           ^360">
        <function class="functions.continuous.unconstrained.Rosenbrock" domain="R(-2.048,2.048)^30"/>
      </function>
      </function>
    </problem>
    <problem id="rastrigin" class="problem.FunctionMinimisationProblem">
        <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^300">
      <function class="functions.discrete.BinaryAdapter" precision="2" bitsPerDimension="10" domain="B
           ^300">
        <function class="functions.continuous.unconstrained.Rastrigin" domain="R(-5.12,5.12)^30"/>
      </function>
      </function>
    </problem>
  </problems>
  <measurements id="measurements" class="simulator.MeasurementSuite" samples="1" resolution="1">
    <addMeasurement class="measurement.single.Fitness"/>
  <!-- <addMeasurement class="measurement.single.Solution"/>-->
  </measurements>
  <simulations>
  <simulation>
      <algorithm idref="abc"/>
      <problem idref="spherical"/>
      <measurements idref="measurements" file="data/masters/abc/binabc/binabc-spherical.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="abc"/>
      <problem idref="rosenbrock2d"/>
      <measurements idref="measurements" file="data/masters/abc/binabc/binabc-rosenbrock2d.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="abc"/>
      <problem idref="step"/>
      <measurements idref="measurements" file="data/masters/abc/binabc/binabc-step.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="abc"/>
      <problem idref="quartic"/>
      <measurements idref="measurements" file="data/masters/abc/binabc/binabc-quartic.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="abc"/>
      <problem idref="foxholes"/>
```

```
      <measurements idref="measurements" file="data/masters/abc/binabc/binabc-foxholes.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="abc"/>
      <problem idref="schaffer6"/>
      <measurements idref="measurements" file="data/masters/abc/binabc/binabc-schaffer6.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="abc"/>
      <problem idref="griewank"/>
      <measurements idref="measurements" file="data/masters/abc/binabc/binabc-griewank.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="abc"/>
      <problem idref="ackley"/>
      <measurements idref="measurements" file="data/masters/abc/binabc/binabc-ackley.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="abc"/>
      <problem idref="rosenbrock"/>
      <measurements idref="measurements" file="data/masters/abc/binabc/binabc-rosenbrock.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="abc"/>
      <problem idref="rastrigin"/>
      <measurements idref="measurements" file="data/masters/abc/binabc/binabc-rastrigin.txt"/>
    </simulation>
  </simulations>
</simulator>
```

Normalized artificial bee colony

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE simulator [
 <!ATTLIST algorithm id ID #IMPLIED>
<!ATTLIST problem id ID #IMPLIED>
<!ATTLIST measurements id ID #IMPLIED>
]>
<simulator>
  <algorithms>
        <algorithm id="abc" class="boa.ABC">
        <initialisationStrategy class="algorithm.initialisation.ClonedPopulationInitialisationStrategy
              " entityNumber="40">
            <entityType class="boa.bee.WorkerBee">
                <fitnessCalculator class="util.calculator.BinaryConvertedFitnessCalculator" />
            </entityType>
        </initialisationStrategy>
        <addStoppingCondition class="stoppingcondition.MaximumIterations" maximumIterations="10000"/>
        <dancingSelectionStrategy class="entity.operators.selection.RouletteWheelSelectionStrategy"/>
        <workerBeePercentage class="controlparameter.ConstantControlParameter" parameter="0.5"/>
        <forageLimit class="controlparameter.ConstantControlParameter" parameter="500"/>
        <explorerBeeUpdateLimit class="controlparameter.ConstantControlParameter" parameter="1"/>
    </algorithm>
  </algorithms>
  <problems>
    <problem id="spherical" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^300">
        <function class="functions.discrete.BinaryAdapter" precision="2" bitsPerDimension="10" domain=
              "B^300">
          <function class="functions.continuous.unconstrained.Spherical" domain="R(-5.12,5.12)^30"/>
        </function>
      </function>
    </problem>
    <problem id="rosenbrock2d" class="problem.FunctionMinimisationProblem">
        <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^24">
      <function class="functions.discrete.BinaryAdapter" precision="3" bitsPerDimension="12" domain="B
          ^24">
        <function class="functions.continuous.unconstrained.Rosenbrock" domain="R(-2.048,2.048)^2"/>
        </function>
      </function>
    </problem>
    <problem id="step" class="problem.FunctionMinimisationProblem">
        <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^50">
      <function class="functions.discrete.BinaryAdapter" precision="2" bitsPerDimension="10" domain="B
          ^50">
        <function class="functions.continuous.Step" domain="R(-5.12,5.12)^5"/>
        </function>
      </function>
    </problem>
    <problem id="quartic" class="problem.FunctionMinimisationProblem">
        <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^240">
      <function class="functions.discrete.BinaryAdapter" precision="2" bitsPerDimension="8" domain="B
          ^240">
        <function class="functions.continuous.Quartic" domain="R(-1.28,1.28)^30"/>
      </function>
      </function>
    </problem>
    <problem id="foxholes" class="problem.FunctionMinimisationProblem">
        <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^34">
      <function class="functions.discrete.BinaryAdapter" precision="0" bitsPerDimension="17" domain="B
          ^34">
        <function class="functions.continuous.Foxholes" domain="R(-65536.0,65536.0)^2"/>
      </function>
      </function>
    </problem>
    <problem id="schaffer6" class="problem.FunctionMinimisationProblem">
        <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^16">
      <function class="functions.discrete.BinaryAdapter" precision="0" bitsPerDimension="8" domain="B
          ^16">
        <function class="functions.continuous.Schaffer6" domain="R(-100.0,100.0)^2"/>
      </function>
      </function>
    </problem>
   <problem id="griewank" class="problem.FunctionMinimisationProblem">
       <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^300">
      <function class="functions.discrete.BinaryAdapter" precision="0" bitsPerDimension="10" domain="B
          ^300">
        <function class="functions.continuous.unconstrained.Griewank" domain="R(-300.0,300.0)^30"/>
      </function>
      </function>
    </problem>
    <problem id="ackley" class="problem.FunctionMinimisationProblem">
        <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^180">
      <function class="functions.discrete.BinaryAdapter" precision="0" bitsPerDimension="6" domain="B
          ^180">
        <function class="functions.continuous.unconstrained.Ackley" domain="R(-30.0,30.0)^30"/>
      </function>
      </function>
    </problem>
    <problem id="rosenbrock" class="problem.FunctionMinimisationProblem">
        <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^360">
      <function class="functions.discrete.BinaryAdapter" precision="3" bitsPerDimension="12" domain="B
          ^360">
        <function class="functions.continuous.unconstrained.Rosenbrock" domain="R(-2.048,2.048)^30"/>
      </function>
      </function>
    </problem>
    <problem id="rastrigin" class="problem.FunctionMinimisationProblem">
        <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^300">
      <function class="functions.discrete.BinaryAdapter" precision="2" bitsPerDimension="10" domain="B
         ^300">
        <function class="functions.continuous.unconstrained.Rastrigin" domain="R(-5.12,5.12)^30"/>
      </function>
      </function>
    </problem>
  </problems>
  <measurements id="measurements" class="simulator.MeasurementSuite" samples="1" resolution="1">
    <addMeasurement class="measurement.single.Fitness"/>
  <!-- <addMeasurement class="measurement.single.Solution"/>-->
  </measurements>
  <simulations>
  <simulation>
      <algorithm idref="abc"/>
      <problem idref="spherical"/>
      <measurements idref="measurements" file="data/masters/abc/normabc/normabc-spherical.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="abc"/>
      <problem idref="rosenbrock2d"/>
      <measurements idref="measurements" file="data/masters/abc/normabc/normabc-rosenbrock2d.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="abc"/>
      <problem idref="step"/>
      <measurements idref="measurements" file="data/masters/abc/normabc/normabc-step.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="abc"/>
      <problem idref="quartic"/>
      <measurements idref="measurements" file="data/masters/abc/normabc/normabc-quartic.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="abc"/>
      <problem idref="foxholes"/>
      <measurements idref="measurements" file="data/masters/abc/normabc/normabc-foxholes.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="abc"/>
      <problem idref="schaffer6"/>
      <measurements idref="measurements" file="data/masters/abc/normabc/normabc-schaffer6.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="abc"/>
      <problem idref="griewank"/>
      <measurements idref="measurements" file="data/masters/abc/normabc/normabc-griewank.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="abc"/>
      <problem idref="ackley"/>
      <measurements idref="measurements" file="data/masters/abc/normabc/normabc-ackley.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="abc"/>
      <problem idref="rosenbrock"/>
      <measurements idref="measurements" file="data/masters/abc/normabc/normabc-rosenbrock.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="abc"/>
      <problem idref="rastrigin"/>
      <measurements idref="measurements" file="data/masters/abc/normabc/normabc-rastrigin.txt"/>
    </simulation>
  </simulations>
</simulator>
```

E.5.2 Binary simulations

Angle modulated artificial bee colony

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE simulator [
<!ATTLIST algorithm id ID #IMPLIED>
<!ATTLIST problem id ID #IMPLIED>
<!ATTLIST measurements id ID #IMPLIED>
]>
<simulator>
  <algorithms>
      <algorithm id="abc" class="boa.ABC">
        <initialisationStrategy class="algorithm.initialisation.ClonedPopulationInitialisationStrategy
             " entityNumber="40">
            <entityType class="boa.bee.WorkerBee"/>
        </initialisationStrategy>
        <addStoppingCondition class="stoppingcondition.MaximumIterations" maximumIterations="10000"/>
        <dancingSelectionStrategy class="entity.operators.selection.RouletteWheelSelectionStrategy"/>
        <workerBeePercentage class="controlparameter.ConstantControlParameter" parameter="0.5"/>
        <forageLimit class="controlparameter.ConstantControlParameter" parameter="500"/>
        <explorerBeeUpdateLimit class="controlparameter.ConstantControlParameter" parameter="1"/>
    </algorithm>
  </algorithms>
  <problems>
      <problem id="mdk-10" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.KnapSack" domain="B^10">
    <capacity value="269" />
    <numberOfObjects value="10" />
    <weight value="95,4,60,32,23,72,80,62,64,46" />
    <value value="55,10,47,5,4,50,8,61,85,87" />
  </function>
      </function>
    </problem>
    <problem id="mdk-20" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.KnapSack" domain="B^20">
    <capacity value="878" />
    <numberOfObjects value="20" />
    <weight value="92,4,43,83,84,68,92,82,6,44,32,18,56,83,25,96,70,48,14,58" />
    <value value="44,46,90,72,91,40,75,35,8,54,78,40,77,15,61,17,75,29,75,63" />
  </function>
      </function>
    </problem>
    <problem id="kc-4" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.KnightsCoverage" boardSize="4" />
      </function>
    </problem>
    <problem id="kc-5" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.KnightsCoverage" boardSize="5" />
      </function>
    </problem>
    <problem id="kc-6" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.KnightsCoverage" boardSize="6" />
      </function>
    </problem>
    <problem id="kc-7" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.KnightsCoverage" boardSize="7" />
      </function>
    </problem>
    <problem id="kc-8" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.KnightsCoverage" boardSize="8" />
      </function>
    </problem>
        <problem id="q-4" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.Queens" boardSize="4" />
      </function>
    </problem>
    <problem id="q-5" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.Queens" boardSize="5" />
      </function>
    </problem>
    <problem id="q-6" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.Queens" boardSize="6" />
      </function>
    </problem>
    <problem id="q-7" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.Queens" boardSize="7" />
      </function>
    </problem>
    <problem id="q-8" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.Queens" boardSize="8" />
      </function>
    </problem>
    <problem id="kt-4" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.RepairingKnightsTour" boardSize="4" />
      </function>
    </problem>
    <problem id="kt-5" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.RepairingKnightsTour" boardSize="5" />
      </function>
    </problem>
    <problem id="kt-6" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.RepairingKnightsTour" boardSize="6" />
      </function>
    </problem>
    <problem id="kt-7" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.RepairingKnightsTour" boardSize="7" />
      </function>
    </problem>
    <problem id="kt-8" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.RepairingKnightsTour" boardSize="8" />
      </function>
    </problem>
    <problem id="ipd" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.AngleModulation" precision="2">
        <function class="functions.discrete.IteratedPrisonersDilemma" domain="B^64" />
      </function>
    </problem>
  </problems>
  <measurements id="measurements" class="simulator.MeasurementSuite" samples="30" resolution="1">
    <addMeasurement class="measurement.single.Fitness"/>
    <addMeasurement class="measurement.single.CollectiveFitness"/>
  <!-- <addMeasurement class="measurement.single.Solution"/>-->
  </measurements>
  <simulations>
```

```
    <simulation>
      <algorithm idref="abc"/>
      <problem idref="mdk-10"/>
      <measurements idref="measurements" file="data/masters/binary/abc/amabc/amabc-mdk-10.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="abc"/>
      <problem idref="mdk-20"/>
      <measurements idref="measurements" file="data/masters/binary/abc/amabc/amabc-mdk-20.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="abc"/>
      <problem idref="kc-4"/>
      <measurements idref="measurements" file="data/masters/binary/abc/amabc/amabc-kc-4.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="abc"/>
      <problem idref="kc-5"/>
      <measurements idref="measurements" file="data/masters/binary/abc/amabc/amabc-kc-5.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="abc"/>
      <problem idref="kc-6"/>
      <measurements idref="measurements" file="data/masters/binary/abc/amabc/amabc-kc-6.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="abc"/>
      <problem idref="kc-7"/>
      <measurements idref="measurements" file="data/masters/binary/abc/amabc/amabc-kc-7.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="abc"/>
      <problem idref="kc-8"/>
      <measurements idref="measurements" file="data/masters/binary/abc/amabc/amabc-kc-8.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="abc"/>
      <problem idref="q-4"/>
      <measurements idref="measurements" file="data/masters/binary/abc/amabc/amabc-q-4.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="abc"/>
      <problem idref="q-5"/>
      <measurements idref="measurements" file="data/masters/binary/abc/amabc/amabc-q-5.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="abc"/>
      <problem idref="q-6"/>
      <measurements idref="measurements" file="data/masters/binary/abc/amabc/amabc-q-6.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="abc"/>
      <problem idref="q-7"/>
      <measurements idref="measurements" file="data/masters/binary/abc/amabc/amabc-q-7.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="abc"/>
      <problem idref="q-8"/>
      <measurements idref="measurements" file="data/masters/binary/abc/amabc/amabc-q-8.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="abc" />
      <problem idref="kt-4" />
      <measurements idref="measurements" file="data/masters/binary/abc/amabc/amabc-kt-4.txt" />
    </simulation>
    <simulation>
      <algorithm idref="abc" />
      <problem idref="kt-5" />
      <measurements idref="measurements" file="data/masters/binary/abc/amabc/amabc-kt-5.txt" />
    </simulation>
    <simulation>
      <algorithm idref="abc" />
      <problem idref="kt-6" />
      <measurements idref="measurements" file="data/masters/binary/abc/amabc/amabc-kt-6.txt" />
    </simulation>
    <simulation>
      <algorithm idref="abc" />
      <problem idref="kt-7" />
      <measurements idref="measurements" file="data/masters/binary/abc/amabc/amabc-kt-7.txt" />
    </simulation>
    <simulation>
      <algorithm idref="abc" />
      <problem idref="kt-8" />
      <measurements idref="measurements" file="data/masters/binary/abc/amabc/amabc-kt-8.txt" />
    </simulation>
    <simulation>
      <algorithm idref="abc" />
      <problem idref="ipd" />
      <measurements idref="measurements" file="data/masters/binary/abc/amabc/amabc-ipd.txt" />
    </simulation>
  </simulations>
</simulator>
```

Binary artificial bee colony

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE simulator [
<!ATTLIST algorithm id ID #IMPLIED>
<!ATTLIST problem id ID #IMPLIED>
<!ATTLIST measurements id ID #IMPLIED>
]>
<simulator>
  <algorithms>
    <algorithm id="abc" class="boa.ABC">
        <initialisationStrategy class="algorithm.initialisation.ClonedPopulationInitialisationStrategy
            " entityNumber="40">
            <entityType class="boa.bee.WorkerBee">
                <fitnessCalculator class="util.calculator.BinaryConvertedFitnessCalculator" />
            </entityType>
        </initialisationStrategy>
        <addStoppingCondition class="stoppingcondition.MaximumIterations" maximumIterations="10000"/>
        <dancingSelectionStrategy class="entity.operators.selection.RouletteWheelSelectionStrategy"/>
        <workerBeePercentage class="controlparameter.ConstantControlParameter" parameter="0.5"/>
        <forageLimit class="controlparameter.ConstantControlParameter" parameter="500"/>
        <explorerBeeUpdateLimit class="controlparameter.ConstantControlParameter" parameter="1"/>
    </algorithm>
  </algorithms>
  <problems>
    <problem id="mdk-10" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^10">
          <function class="functions.discrete.KnapSack" domain="B^10">
    <capacity value="269" />
    <numberOfObjects value="10" />
    <weight value="95,4,60,32,23,72,80,62,64,46" />
    <value value="55,10,47,5,4,50,8,61,85,87" />
        </function>
      </function>
    </problem>
    <problem id="mdk-20" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^20">
```

```
        <function class="functions.discrete.KnapSack" domain="B^20">
      <capacity value="878" />
      <numberOfObjects value="20" />
      <weight value="92,4,43,83,84,68,92,82,6,44,32,18,56,83,25,96,70,48,14,58" />
      <value value="44,46,90,72,91,40,75,35,8,54,78,40,77,15,61,17,75,29,75,63" />
    </function>
        </function>
      </problem>
      <problem id="kc-4" class="problem.FunctionMinimisationProblem">
        <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^16">
          <function class="functions.discrete.KnightsCoverage" boardSize="4" />
        </function>
      </problem>
      <problem id="kc-5" class="problem.FunctionMinimisationProblem">
        <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^25">
          <function class="functions.discrete.KnightsCoverage" boardSize="5" />
        </function>
      </problem>
      <problem id="kc-6" class="problem.FunctionMinimisationProblem">
        <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^36">
          <function class="functions.discrete.KnightsCoverage" boardSize="6" />
        </function>
      </problem>
      <problem id="kc-7" class="problem.FunctionMinimisationProblem">
        <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^49">
          <function class="functions.discrete.KnightsCoverage" boardSize="7" />
        </function>
      </problem>
      <problem id="kc-8" class="problem.FunctionMinimisationProblem">
        <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^64">
          <function class="functions.discrete.KnightsCoverage" boardSize="8" />
        </function>
      </problem>
      <problem id="q-4" class="problem.FunctionMinimisationProblem">
        <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^16">
          <function class="functions.discrete.Queens" boardSize="4" />
        </function>
      </problem>
      <problem id="q-5" class="problem.FunctionMinimisationProblem">
        <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^25">
          <function class="functions.discrete.Queens" boardSize="5" />
        </function>
      </problem>
      <problem id="q-6" class="problem.FunctionMinimisationProblem">
        <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^36">
          <function class="functions.discrete.Queens" boardSize="6" />
        </function>
      </problem>
      <problem id="q-7" class="problem.FunctionMinimisationProblem">
        <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^49">
          <function class="functions.discrete.Queens" boardSize="7" />
        </function>
      </problem>
      <problem id="q-8" class="problem.FunctionMinimisationProblem">
        <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^64">
          <function class="functions.discrete.Queens" boardSize="8" />
        </function>
      </problem>
      <problem id="kt-4" class="problem.FunctionMaximisationProblem">
        <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^48">
          <function class="functions.discrete.RepairingKnightsTour" boardSize="4" />
        </function>
      </problem>
      <problem id="kt-5" class="problem.FunctionMaximisationProblem">
        <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^75">
          <function class="functions.discrete.RepairingKnightsTour" boardSize="5" />
        </function>
      </problem>
      <problem id="kt-6" class="problem.FunctionMaximisationProblem">
        <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^108">
          <function class="functions.discrete.RepairingKnightsTour" boardSize="6" />
        </function>
      </problem>
      <problem id="kt-7" class="problem.FunctionMaximisationProblem">
        <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^147">
          <function class="functions.discrete.RepairingKnightsTour" boardSize="7" />
        </function>
      </problem>
      <problem id="kt-8" class="problem.FunctionMaximisationProblem">
        <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^192">
          <function class="functions.discrete.RepairingKnightsTour" boardSize="8" />
        </function>
      </problem>
      <problem id="ipd" class="problem.FunctionMaximisationProblem">
          <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^64">
          <function class="functions.discrete.IteratedPrisonersDilemma" domain="B^64" />
          </function>
      </problem>
    </problems>
    <measurements id="measurements" class="simulator.MeasurementSuite" samples="30" resolution="1">
      <addMeasurement class="measurement.single.Fitness"/>
      <addMeasurement class="measurement.single.CollectiveFitness" />
    <!-- <addMeasurement class="measurement.single.Solution"/>-->
    </measurements>
    <simulations>
      <simulation>
        <algorithm idref="abc"/>
        <problem idref="mdk-10"/>
        <measurements idref="measurements" file="data/masters/binary/abc/binabc/binabc-mdk-10.txt"/>
      </simulation>
      <simulation>
        <algorithm idref="abc"/>
        <problem idref="mdk-20"/>
        <measurements idref="measurements" file="data/masters/binary/abc/binabc/binabc-mdk-20.txt"/>
      </simulation>
      <simulation>
        <algorithm idref="abc"/>
        <problem idref="kc-4"/>
        <measurements idref="measurements" file="data/masters/binary/abc/binabc/binabc-kc-4.txt"/>
      </simulation>
      <simulation>
        <algorithm idref="abc"/>
        <problem idref="kc-5"/>
        <measurements idref="measurements" file="data/masters/binary/abc/binabc/binabc-kc-5.txt"/>
      </simulation>
      <simulation>
        <algorithm idref="abc"/>
        <problem idref="kc-6"/>
        <measurements idref="measurements" file="data/masters/binary/abc/binabc/binabc-kc-6.txt"/>
      </simulation>
      <simulation>
        <algorithm idref="abc"/>
        <problem idref="kc-7"/>
        <measurements idref="measurements" file="data/masters/binary/abc/binabc/binabc-kc-7.txt"/>
      </simulation>
      <simulation>
        <algorithm idref="abc"/>
        <problem idref="kc-8"/>
        <measurements idref="measurements" file="data/masters/binary/abc/binabc/binabc-kc-8.txt"/>
      </simulation>
      <simulation>
        <algorithm idref="abc"/>
        <problem idref="q-4"/>
        <measurements idref="measurements" file="data/masters/binary/abc/binabc/binabc-q-4.txt"/>
      </simulation>
      <simulation>
        <algorithm idref="abc"/>
        <problem idref="q-5"/>
        <measurements idref="measurements" file="data/masters/binary/abc/binabc/binabc-q-5.txt"/>
      </simulation>
      <simulation>
        <algorithm idref="abc"/>
        <problem idref="q-6"/>
        <measurements idref="measurements" file="data/masters/binary/abc/binabc/binabc-q-6.txt"/>
      </simulation>
      <simulation>
        <algorithm idref="abc"/>
        <problem idref="q-7"/>
        <measurements idref="measurements" file="data/masters/binary/abc/binabc/binabc-q-7.txt"/>
      </simulation>
      <simulation>
        <algorithm idref="abc"/>
        <problem idref="q-8"/>
        <measurements idref="measurements" file="data/masters/binary/abc/binabc/binabc-q-8.txt"/>
      </simulation>
      <simulation>
        <algorithm idref="abc" />
        <problem idref="kt-4" />
        <measurements idref="measurements" file="data/masters/binary/abc/binabc/binabc-kt-4.txt" />
      </simulation>
          <simulation>
        <algorithm idref="abc" />
        <problem idref="kt-5" />
        <measurements idref="measurements" file="data/masters/binary/abc/binabc/binabc-kt-5.txt" />
      </simulation>
          <simulation>
        <algorithm idref="abc" />
        <problem idref="kt-6" />
        <measurements idref="measurements" file="data/masters/binary/abc/binabc/binabc-kt-6.txt" />
      </simulation>
          <simulation>
        <algorithm idref="abc" />
        <problem idref="kt-7" />
        <measurements idref="measurements" file="data/masters/binary/abc/binabc/binabc-kt-7.txt" />
      </simulation>
          <simulation>
        <algorithm idref="abc" />
        <problem idref="kt-8" />
        <measurements idref="measurements" file="data/masters/binary/abc/binabc/binabc-kt-8.txt" />
      </simulation>
      <simulation>
        <algorithm idref="abc" />
        <problem idref="ipd" />
        <measurements idref="measurements" file="data/masters/binary/abc/binabc/binabc-ipd.txt" />
      </simulation>
    </simulations>
  </simulator>
```

Normalized artificial bee colony

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE simulator [
 <!ATTLIST algorithm id ID #IMPLIED>
<!ATTLIST problem id ID #IMPLIED>
<!ATTLIST measurements id ID #IMPLIED>
]>
<simulator>
```

```
<algorithms>
      <algorithm id="abc" class="boa.ABC">
      <initialisationStrategy class="algorithm.initialisation.ClonedPopulationInitialisationStrategy
           " entityNumber="40">
          <entityType class="boa.bee.WorkerBee">
              <fitnessCalculator class="util.calculator.BinaryConvertedFitnessCalculator" />
          </entityType>
      </initialisationStrategy>
      <addStoppingCondition class="stoppingcondition.MaximumIterations" maximumIterations="10000"/>
      <dancingSelectionStrategy class="entity.operators.selection.RouletteWheelSelectionStrategy"/>
      <workerBeePercentage class="controlparameter.ConstantControlParameter" parameter="0.5"/>
      <forageLimit class="controlparameter.ConstantControlParameter" parameter="500"/>
      <explorerBeeUpdateLimit class="controlparameter.ConstantControlParameter" parameter="1"/>
  </algorithm>
</algorithms>
<problems>
  <problem id="mdk-10" class="problem.FunctionMaximisationProblem">
    <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^10">
        <function class="functions.discrete.KnapSack" domain="B^10">
  <capacity value="269" />
  <numberOfObjects value="10" />
  <weight value="95,4,60,32,23,72,80,62,64,46" />
  <value value="55,10,47,5,4,50,8,61,85,87" />
      </function>
    </function>
  </problem>
  <problem id="mdk-20" class="problem.FunctionMaximisationProblem">
    <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^20">
      <function class="functions.discrete.KnapSack" domain="B^20">
  <capacity value="878" />
  <numberOfObjects value="20" />
  <weight value="92,4,43,83,84,68,92,82,6,44,32,18,56,83,25,96,70,48,14,58" />
  <value value="44,46,90,72,91,40,75,35,8,54,78,40,77,15,61,17,75,29,75,63" />
</function>
    </function>
  </problem>
  <problem id="kc-4" class="problem.FunctionMinimisationProblem">
    <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^16">
      <function class="functions.discrete.KnightsCoverage" boardSize="4" />
    </function>
  </problem>
  <problem id="kc-5" class="problem.FunctionMinimisationProblem">
    <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^25">
      <function class="functions.discrete.KnightsCoverage" boardSize="5" />
    </function>
  </problem>
  <problem id="kc-6" class="problem.FunctionMinimisationProblem">
    <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^36">
      <function class="functions.discrete.KnightsCoverage" boardSize="6" />
    </function>
  </problem>
  <problem id="kc-7" class="problem.FunctionMinimisationProblem">
    <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^49">
      <function class="functions.discrete.KnightsCoverage" boardSize="7" />
    </function>
  </problem>
  <problem id="kc-8" class="problem.FunctionMinimisationProblem">
    <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^64">
      <function class="functions.discrete.KnightsCoverage" boardSize="8" />
    </function>
  </problem>
  <problem id="q-4" class="problem.FunctionMinimisationProblem">
    <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^16">
      <function class="functions.discrete.Queens" boardSize="4" />
    </function>
  </problem>
  <problem id="q-5" class="problem.FunctionMinimisationProblem">
```

```
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^25">
        <function class="functions.discrete.Queens" boardSize="5" />
      </function>
    </problem>
    <problem id="q-6" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^36">
        <function class="functions.discrete.Queens" boardSize="6" />
      </function>
    </problem>
    <problem id="q-7" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^49">
        <function class="functions.discrete.Queens" boardSize="7" />
      </function>
    </problem>
    <problem id="q-8" class="problem.FunctionMinimisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^64">
        <function class="functions.discrete.Queens" boardSize="8" />
      </function>
    </problem>
    <problem id="kt-4" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^48">
        <function class="functions.discrete.RepairingKnightsTour" boardSize="4" />
      </function>
    </problem>
    <problem id="kt-5" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^75">
        <function class="functions.discrete.RepairingKnightsTour" boardSize="5" />
      </function>
    </problem>
    <problem id="kt-6" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^108">
        <function class="functions.discrete.RepairingKnightsTour" boardSize="6" />
      </function>
    </problem>
    <problem id="kt-7" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^147">
        <function class="functions.discrete.RepairingKnightsTour" boardSize="7" />
      </function>
    </problem>
    <problem id="kt-8" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^192">
        <function class="functions.discrete.RepairingKnightsTour" boardSize="8" />
      </function>
    </problem>
    <problem id="ipd" class="problem.FunctionMaximisationProblem">
      <function class="functions.continuous.decorators.ForwardingFunction" domain="R(-2.0,2.0)^64">
        <function class="functions.discrete.IteratedPrisonersDilemma" domain="B^64" />
      </function>
    </problem>
  </problems>
  <measurements id="measurements" class="simulator.MeasurementSuite" samples="30" resolution="1">
    <addMeasurement class="measurement.single.Fitness"/>
    <addMeasurement class="measurement.single.CollectiveFitness" />
  <!-- <addMeasurement class="measurement.single.Solution"/>-->
  </measurements>
  <simulations>
    <simulation>
      <algorithm idref="abc"/>
      <problem idref="mdk-10"/>
      <measurements idref="measurements" file="data/masters/binary/abc/normabc/normabc-mdk-10.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="abc"/>
      <problem idref="mdk-20"/>
      <measurements idref="measurements" file="data/masters/binary/abc/normabc/normabc-mdk-20.txt"/>
    </simulation>
    <simulation>
```

```
      <algorithm idref="abc"/>
      <problem idref="kc-4"/>
      <measurements idref="measurements" file="data/masters/binary/abc/normabc/normabc-kc-4.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="abc"/>
      <problem idref="kc-5"/>
      <measurements idref="measurements" file="data/masters/binary/abc/normabc/normabc-kc-5.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="abc"/>
      <problem idref="kc-6"/>
      <measurements idref="measurements" file="data/masters/binary/abc/normabc/normabc-kc-6.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="abc"/>
      <problem idref="kc-7"/>
      <measurements idref="measurements" file="data/masters/binary/abc/normabc/normabc-kc-7.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="abc"/>
      <problem idref="kc-8"/>
      <measurements idref="measurements" file="data/masters/binary/abc/normabc/normabc-kc-8.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="abc"/>
      <problem idref="q-4"/>
      <measurements idref="measurements" file="data/masters/binary/abc/normabc/normabc-q-4.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="abc"/>
      <problem idref="q-5"/>
      <measurements idref="measurements" file="data/masters/binary/abc/normabc/normabc-q-5.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="abc"/>
      <problem idref="q-6"/>
      <measurements idref="measurements" file="data/masters/binary/abc/normabc/normabc-q-6.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="abc"/>
      <problem idref="q-7"/>
      <measurements idref="measurements" file="data/masters/binary/abc/normabc/normabc-q-7.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="abc"/>
      <problem idref="q-8"/>
      <measurements idref="measurements" file="data/masters/binary/abc/normabc/normabc-q-8.txt"/>
    </simulation>
    <simulation>
      <algorithm idref="abc" />
      <problem idref="kt-4" />
      <measurements idref="measurements" file="data/masters/binary/abc/normabc/normabc-kt-4.txt" />
    </simulation>
        <simulation>
      <algorithm idref="abc" />
      <problem idref="kt-5" />
      <measurements idref="measurements" file="data/masters/binary/abc/normabc/normabc-kt-5.txt" />
    </simulation>
        <simulation>
      <algorithm idref="abc" />
      <problem idref="kt-6" />
      <measurements idref="measurements" file="data/masters/binary/abc/normabc/normabc-kt-6.txt" />
    </simulation>
        <simulation>
      <algorithm idref="abc" />
      <problem idref="kt-7" />
      <measurements idref="measurements" file="data/masters/binary/abc/normabc/normabc-kt-7.txt" />
    </simulation>
        <simulation>
      <algorithm idref="abc" />
      <problem idref="kt-8" />
      <measurements idref="measurements" file="data/masters/binary/abc/normabc/normabc-kt-8.txt" />
    </simulation>
    <simulation>
      <algorithm idref="abc" />
      <problem idref="ipd" />
      <measurements idref="measurements" file="data/masters/binary/abc/normabc/normabc-ipd.txt" />
    </simulation>
  </simulations>
</simulator>
```

E.6 Additional Java sources

The following Java sources files are additions to the main CIlib tree and are required to allow the simulations to execute.

E.6.1 ForwardingFunction.java

```java
/**
 * Copyright (C) 2003 - 2009
 * Computational Intelligence Research Group (CIRG@UP)
 * Department of Computer Science
 * University of Pretoria
 * South Africa
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 */
package net.sourceforge.cilib.functions.continuous.decorators;

import net.sourceforge.cilib.functions.ContinuousFunction;
import net.sourceforge.cilib.functions.Function;
import net.sourceforge.cilib.type.types.container.Vector;

/**
 *
 */
public class ForwardingFunction extends ContinuousFunction {
    private static final long serialVersionUID = 1470824724049394166L;

    private Function function;

    public ForwardingFunction() {

    }

    @Override
    public ContinuousFunction getClone() {
        return this;
    }

    @Override
    public Double evaluate(Vector input) {
        return ((Number) this.function.evaluate(input)).doubleValue();
    }

    public Function getFunction() {
        return function;
    }

    public void setFunction(Function function) {
        this.function = function;
    }

}
```

E.6.2 BinaryConvertedFitnessCalculator.java

```java
/**
 * Copyright (C) 2003 - 2009
 * Computational Intelligence Research Group (CIRG@UP)
 * Department of Computer Science
 * University of Pretoria
 * South Africa
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 */
package net.sourceforge.cilib.util.calculator;

import net.sourceforge.cilib.algorithm.AbstractAlgorithm;
import net.sourceforge.cilib.algorithm.Algorithm;
import net.sourceforge.cilib.entity.Entity;
import net.sourceforge.cilib.functions.activation.ActivationFunction;
import net.sourceforge.cilib.functions.activation.Sigmoid;
import net.sourceforge.cilib.problem.Fitness;
import net.sourceforge.cilib.type.types.Bit;
import net.sourceforge.cilib.type.types.container.Vector;

/**
 * This class determines the fitness of a provided candidate solution by
 * first converting the candidate solution into a binary represenation.
 *
 * This binary conversion process is the same as the process followed by
 * the BinaryPSO, whereby a sigmoid function is used to perform the conversion.
 */
public class BinaryConvertedFitnessCalculator implements FitnessCalculator<Entity> {
    private static final long serialVersionUID = 2217505781797844357L;

    private ActivationFunction activationFunction;

    public BinaryConvertedFitnessCalculator() {
        this.activationFunction = new Sigmoid();
    }

    @Override
    public FitnessCalculator<Entity> getClone() {
        return this;
    }

    @Override
    public Fitness getFitness(Entity entity) {
        Vector target = toBitVector((Vector) entity.getCandidateSolution());
        Algorithm algorithm = AbstractAlgorithm.get();
        Fitness result = algorithm.getOptimisationProblem().getFitness(target);
        return result;
    }

    Vector toBitVector(Vector candidateSolution) {
        Vector result = new Vector(candidateSolution.size());
        for (int i = 0; i < candidateSolution.size(); i++) {
            double activation = activationFunction.evaluate(candidateSolution.getReal(i));
            if (activation >= 0.5)
                result.add(new Bit(false));
            else
                result.add(new Bit(true));
        }
        return result;
    }
}
```

E.6.3 NormalizedFitnessCalculator.java

```
/**
 * Copyright (C) 2003 - 2009
 * Computational Intelligence Research Group (CIRG@UP)
 * Department of Computer Science
 * University of Pretoria
 * South Africa
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 */
package net.sourceforge.cilib.util.calculator;

import net.sourceforge.cilib.algorithm.AbstractAlgorithm;
```

```java
import net.sourceforge.cilib.algorithm.Algorithm;
import net.sourceforge.cilib.entity.Entity;
import net.sourceforge.cilib.problem.Fitness;
import net.sourceforge.cilib.type.types.Bit;
import net.sourceforge.cilib.type.types.container.Vector;

/**
 *
 */
public class NormalizedFitnessCalculator implements FitnessCalculator<Entity> {
    private static final long serialVersionUID = 9166955791746197888L;

    @Override
    public FitnessCalculator<Entity> getClone() {
        return this;
    }

    @Override
    public Fitness getFitness(Entity entity) {
        Vector target = normalize((Vector) entity.getCandidateSolution());

        Algorithm algorithm = AbstractAlgorithm.get();
        Fitness result = algorithm.getOptimisationProblem().getFitness(target);
        return result;
    }

    private Vector normalize(Vector vector) {
        double min = Double.MAX_VALUE;
        double max = Double.MIN_VALUE;
        Vector result = new Vector();

        for (int i = 0; i < vector.size(); i++) {
            max = Math.max(max, vector.getReal(i));
            min = Math.min(min, vector.getReal(i));
        }

        for (int i = 0; i < vector.size(); i++) {
            double value = ((vector.getReal(i) - min)/(max - min));

            if (value < 0.5)
                result.add(new Bit(false));
            else
                result.add(new Bit(true));
        }

        return result;
    }

}
```

E.6.4 IteratedPrisonersDilemma.java

```java
/**
 * Copyright (C) 2003 - 2009
 * Computational Intelligence Research Group (CIRG@UP)
 * Department of Computer Science
 * University of Pretoria
 * South Africa
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 */
package net.sourceforge.cilib.functions.discrete;

import com.google.common.base.Function;
import com.google.common.collect.Lists;
import java.util.List;
import net.sourceforge.cilib.algorithm.AbstractAlgorithm;
import net.sourceforge.cilib.algorithm.population.PopulationBasedAlgorithm;
import net.sourceforge.cilib.entity.Entity;
import net.sourceforge.cilib.functions.DiscreteFunction;
import net.sourceforge.cilib.functions.continuous.decorators.AngleModulation;
import net.sourceforge.cilib.type.types.container.Vector;

/**
 *
 */
public class IteratedPrisonersDilemma extends DiscreteFunction {
    private static final long serialVersionUID = -4799591573304887490L;

    private AngleModulation am;

    @Override
    public IteratedPrisonersDilemma getClone() {
        return this;
    }

    /**
     * This method evaluates the current strategy against the developed strategies
     * of ALL the other entities within the currently executing algorithm.
     *
     * Defect is defined to be a value of 1 (true bit) and cooperate is a value of 0 (false bit)
     * @param input
     * @return
     */
    @Override
    public Integer evaluate(Vector input) {
        if (am == null) {
            am = new AngleModulation();
            am.setFunction(this);
        }

        int result = 0;
        PopulationBasedAlgorithm pba = (PopulationBasedAlgorithm) AbstractAlgorithm.get();
        List<Vector> vectors  = Lists.transform(pba.getTopology(), new Function<Entity, Vector>() {
            @Override
            public Vector apply(Entity from) {
                return toBitVector((Vector) from.getCandidateSolution());
            }

            Vector toBitVector(Vector candidateSolution) {
                return am.decodeBitString(am.generateBitString(candidateSolution, 1), 1);
            }
        });

        for (Vector vector : vectors) {
            if (vector == input) {
                continue;
            }

            for (int i = 0; i < input.size(); i++) {
                boolean mine = input.getBit(i);
                boolean other = vector.getBit(i); //With AM, the size of this vector is 4....

                if (defect(mine) && cooperate(other))
                    result += 5; // 5 + 0
                else if (defect(mine) && defect(other))
                    result += 1; // 1 + 1
                else if (cooperate(mine) && cooperate(other))
                    result += 6; // 3 + 3
                else
                    result += 0; // 0 + 0
            }
        }

        return result;
    }

    private boolean defect(boolean choice) {
        return choice ? true : false;
    }

    private boolean cooperate(boolean choice) {
        return !defect(choice);
    }

    @Override
    public Integer getMaximum() {
        return getDomainRegistry().getDimension() * 5;
    }

    @Override
    public Integer getMinimum() {
        return 0;
    }

}
```

E.7 Summary

In this appendix, the simulation specifications for the performed experiments have been listed. Additionally, supplemental Java source files have been provided. These Java classes are required to ensure that the simulations may execute.