Dynamic Multi-Objective Optimization for Financial Markets

by

Frederick Ditliac Atiah

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

December 2019

Publication data:

Frederick Ditliac Atiah. Dynamic Multi-Objective Optimization for Financial Markets. Master's dissertation, University of Pretoria, Department of Computer Science, Pretoria, South Africa, December 2019.

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

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

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

Dynamic Multi-Objective Optimization for Financial Markets

by

Frederick Ditliac Atiah
E-mail: u16403381@tuks.co.za

Abstract

The foreign exchange (Forex) market has over 5 trillion USD turnover per day. In addition, it is one of the most volatile and dynamic markets in the world. Market conditions continue to change every second. Algorithmic trading in Financial markets have received a lot of attention in recent years. However, only few literature have explored the applicability and performance of various dynamic multi-objective algorithms (DMOAs) in the Forex market. This dissertation proposes a dynamic multi-swarm multi-objective particle swarm optimization (DMS-MOPSO) to solve dynamic MOPs (DMOPs). In order to explore the performance and applicability of DMS-MOPSO, the algorithm is adapted for the Forex market. This dissertation also explores the performance of different variants of dynamic particle swarm optimization (PSO), namely the charge PSO (cPSO) and quantum PSO (qPSO), for the Forex market. However, since the Forex market is not only dynamic but have different conflicting objectives, a single-objective optimization algorithm (SOA) might not yield profit over time. For this reason, the Forex market was defined as a multi-objective optimization problem (MOP). Moreover, maximizing profit in a financial time series, like Forex, with computational intelligence (CI) techniques is very challenging. It is even more challenging to make a decision from the solutions of a MOP, like automated Forex trading. This dissertation also explores the effects of five decision models (DMs) on DMS-MOPSO and other three state-of-the-art DMOAs, namely the dynamic vector-evaluated particle swarm optimization (DVEPSO) algorithm, the multi-objective particle swarm optimization algorithm with crowded distance (MOPSO-CD) and dynamic non-dominated sorting genetic algorithm II (DNSGA-II). The effects of constraints handling and the, knowledge sharing approach amongst sub-swarms were

explored for DMS-MOPSO. DMS-MOPSO is compared against other state-of-the-art multi-objective algorithms (MOAs) and dynamic SOAs. A sliding window mechanism is employed over different types of currency pairs. The focus of this dissertation is to optimized technical indicators to maximized the profit and minimize the transaction cost.

The obtained results showed that both dynamic single-objective optimization (SOO) algorithms and dynamic multi-objective optimization (MOO) algorithms performed better than static algorithms on dynamic poroblems. Moreover, the results also showed that a multi-swarm approach for MOO can solve dynamic MOPs.

Keywords: Dynamic multi-objective optimization, nature-inspired computation, technical indicators, foreign exchange, Forex, computational intelligence, swarm intelligence; NSGA-II, DVEPSO, MOPSO, Multi-objective optimization.

Supervisor : Dr. Mardé Helbig
Co-supervisor: Dr. Anna Bosman
Department : Department of Computer Science
Degree : Master of Science

"What has been will be again, what has been done will be done again; there is nothing new under the sun."

— Ecclesiastes 1:9, The Bible.

Acknowledgments

I will like to express my gratitude to the following people for their assistance during the production of this dissertation:

- Dr. Mardé Helbig, my supervisor, and Dr. Anna Bosman, my co-supervisor, for their continuous guidance, inspiration, patience and assistance during the entire journey of my masters
- My wife, Donah, for her endless love, support, encouragement and patience throughout the many years it took to complete the dissertation.
- The members of the Computational Intelligence Research Group (CIRG), for their advice and assistance.
- The Centre for High Performance Computing (CHPC) for the use of their infrastructure to run the simulations and obtaining the required data. A special word of thanks to the technical support team, who provided support when ever problems occurred.
- My mom, my siblings for their love and encouragement and for always believing in me.
- Mastercard Foundation Scholars Program for their financial support to help concentrate on my studies.

Contents

List of Figures

List of Algorithms

List of Tables

Chapter 1

Introduction

> *Look here, you who say, "Today or tomorrow we are going to a certain town and will stay there a year. We will do business there and make a profit." How do you know what your life will be like tomorrow? Your life is like the morning fog—it's here a little while, then it's gone. What you ought to say is, "If the Lord wants us to, we will live and do this or that."*
>
> — James 4:13 - 15, The Bible.

The financial market, especially the stock and foreign exchange (Forex) markets, is one of the most complex and dynamic markets in the world. Because of its complexity, many factors have to be considered before an investor can profit from the market. A number of traditional tools called technical indicators (TIs) have been developed to aid investors to predict the trend of price movement in order to make informed decisions and maximize profit. However, traders are still faced with issues like: 1. Which TI is best suited for a particular market, 2. The best parameter combination for a TI for a particular market. 3. How to combine the signals of TIs to confirm a trend or an investment decision.

The availability of advance technology and data has encouraged and given rise to the use of computational intelligence (CI) techniques to address the above problems in relation to the use of TIs to maximize profit in the Forex market [40]. Moreover, the application of CI algorithms to the Forex or stock markets have shown good results

[8, 35, 43] despite the contradictory statement of efficient-market hypothesis (EMH), namely that the market follows a random walk and that any profit is by chance [28]. CI algorithms, however, were not traditionally developed for finance and economics. In view of that, a high level of adaptability is needed in order to apply CI algorithms to financial markets [37]. CI algorithms or applications need to factor into account the following practical realities of the financial market:

- **Dynamic**: the financial market is very difficult to predict. A good solution now might be the worse solution in the next minute. The nature of the market keeps changing all the time, since the market is being influenced by many factors, including economic news [30]. Using a static algorithm for Forex market might not yield returns over time.

- **Multimodal**: the algorithms should be able to provide a set of solutions to enable investors to make an informed decision. Moreover, different pairs of parameter sets can generate the same profit output in training data, but a different or the same profit output on test data. Hence, the financial market is multimodal in nature.

- **Diversity**: Not only is the financial market dynamic, but the nature of every market is different. For example, in the Forex market, the EURUSD currency pair has less volatility as the USDZAR currency pair (see Table 3.1). The algorithm should be able to adapt to such diversity. It is difficult for one algorithm or model to perform well across all markets.

- **Multi-objective and decision making**: The main objective of many investors is profit. However, investors also consider factors like risk, transaction cost, etc. to make decisions. The algorithms should be able to find good trade-off solutions for decision making.

In this dissertation, a new dynamic multi-swarm multi-objective particle swarm optimization (DMS-MOPSO) is proposed. The purpose of the muti-swarm is to enable the algorithms to track different promising solutions (or trends). DMS-MOPSO is used to optimize four TIs. To apply of metaheuristics to real world problems (Forex) two conflicting objectives are optimized: the net profit and transaction cost are maximized

and minimized simultaneously. DMS-MOPSO is compared against other nature inspired state-of-the-art dynamic multi-objective algorithms (DMOAs).

1.1 Objectives

The main objective of this dissertation is to develop a particle swarm optimization (PSO) based multi-objective algorithm (MOA), namely DMS-MOPSO, to solve dynamic multi-objective optimization problems (MOPs). To accomplish this objective, the following sub-objectives were identified:

- To provide a literature review of the trend in DMOA and single-objective optimization algorithms (SOAs) for the financial market.
- Develop a simulation for automated trading for the Forex market.
- Identify different types of trading currency pairs to run the simulation on.
- Identify different types of technical indicators and trading rules for the experiment.
- Identify a set of performance measures that adequately quantifies the performance of DMOAs for the FOREX market.
- Identify and analyse the performance of different state-of-the-art DMOAs.
- Develop and analyse the performance of DMS-MOPSO.

1.2 Contributions

The novel contributions of this dissertation include the following:

- The first analysis of the applicability of dynamic PSO algorithms to the Forex market.
- The first analysis of the effects of decision models (DMs) on MOAs during automated trading.

- The first adaptation of non-dominance based multi-swarm PSO for a MOP.
- The adaptation of MOAs for the Forex market.
- An overview of current applications of MOAs to the Forex market.

1.3 Dissertation Outline

The remainder of the dissertation is organised as follows:

- **Chapter 2** presents background information on optimization. The concepts of dynamic problems and environments, and also presents some background information on DMOAs and dynamic SOAs were presented. Moreover, single-objective optimization (SOO) variants of PSO algorithms used in this study, which also forms the bases of the proposed algorithm, DMS-MOPSO, and highlights some of the state-of-the-art DMOAs and DMs for DMOAs were presented.
- **Chapter 3** discusses and provides background information of the financial markets, especially the Forex market, which is the real world dynamic problems that optimization algorithms introduced in the study solves. The Chapter also provides a literature review of work already done on the application of computational intelligence algorithms to the financial market. Moreover, details on the experimental setup for this study, and the simulated trading system for the experiment, including the data used, also discussed.
- **Chapter 4** investigates the performance of different variants of dynamic PSO, namely the quantum PSO (qPSO) and charge PSO (cPSO), on optimizing TIs in the Forex market in order to maximize profit. The results obtained are compared with both the performance of standard particle swarm optimization (sPSO) and a time-series particle swarm optimization (tPSO) algorithm on the USDJPY and USDZAR currency pairs.
- **Chapter 5** investigates the effects of five decision models on three state-of-the-art dynamic MOAs, namely the dynamic vector-evaluated particle swarm optimization

(DVEPSO) algorithm, the multi-objective particle swarm optimization algorithm with crowded distance (MOPSO-CD) and the dynamic non-dominated sorting genetic algorithm II (DNSGA-II).

- **Chapter 6** discusses the proposed algorithm, i.e. the DMS-MOPSO algorithm. Moreover, this chapter also discusses the results obtained by different variants of DMS-MOPSO and other state-of-the-art algorithms.
- **Chapter 7** summarises the conclusions of the studies conducted in this dissertation and highlights possible future work from the study.

Incuded are the below appendices which gives more information, and for quick and easy referencing:

- **Appendix A** provides a list of the important acronyms used or newly defined in the course of this work, as well as their associated definitions.
- **Appendix B** lists and defines the mathematical symbols used in this work, categorised according to the relevant chapter in which they appear.
- **Appendix C** lists the publications derived from this work.

Chapter 2

Background

Optimization Theories

This section presents background information on optimization. Section 2.1 discusses the concept and theories of optimization, including optimization problems and the type of solutions. Section 2.2 specifically discusses the definition and concepts of multi-objective optimization (MOO) and MOPs.

2.1 Single-Objective Optimization Theories

The aim of optimization algorithms is to find a solution to an optimization problem by using a search mechanism. An optimization problem is made up of a set of decision variables and one or more objective functions, which are subject to a set of constraints. An optimization problem may differ with respect to the type of decision variables and covariants, the number of optima, etc. Important concepts, such as objective functions, decision variables, constraints, the types of solutions and single-objective optimization problems (SOPs), are discussed below.

2.1.1 Objective function

The aim of an optimization algorithm is to maximize or minimize the quantity represented by the objective function. An optimization problem with only one objective is

known as a SOP, while an optimization problem with more than one objective function to be optimized simultaneously is called a MOP. An optimization problem can be classified as a linear, quadratic or nonlinear optimization problem if the objective function is linear, quadratic or nonlinear in nature respectively.

2.1.2 Decision variables

The objective function value is a result of decision variables. The decision variables are a n dimensional vector. An optimization problem with one decision variable ($n = 1$) is known as a *univariate* problem, while an optimization problem with more than one ($n > 1$) is known as a *multivariate* problem. An optimization algorithm makes changes to the decision variable(s) in each iteration in order to find a good solution. Moreover, decision variables with continuous, integer or a permutation of integer values, make an optimization problem a continuous, discrete or combinatorial optimization problem respectively.

2.1.3 Constraints

Any optimization problem may be subject to a set of constraints. These constraints limit the values that decision variables can have or be assigned by the optimization algorithm. A constraint can either be an *equality* or *inequality* constraint. An equality constraint limits the assignment of values to a decision variable to a specific value, such as $g_x = 3$. An inequality constraint on the other hand, limits value assignment to a range or boundary, such as $-1 < g_{\mathrm{x}} < 1$, or $g_{\mathrm{x}} < 0$ or $g_{\mathrm{x}} > 0$.

2.1.4 Type of Solutions

The aim of a single objective optimization algorithm when solving a SOP is to find a single near optimal solution. However, the obtained solution can be classified according to its quality, as follows:

Global minimum

A global solution is called a global minimum when minimizing or a global maximum when maximizing. A minimization problem is assumed for this section. As depicted in Figure 2.1, the global optimum is the best solution amongst all solutions. A global optimum is mathematically defined as $f(\mathrm{x}^*) <= f(\mathrm{x}), \forall \mathrm{x} \in F$, where x^* is the global optimum of the objective function, f, and F is the feasible solutions from the search space. A problem with only one global optimum is called *unimodal*, while a problem with multiple optima is referred to as *multimodal*.

Strong local minimum

A strong local optimum is mathematically defined as $f(\mathrm{x}_N^*) < f(\mathrm{x}), \forall x \in N$, with $\mathrm{x} \neq \mathrm{x}^*$ and $\mathrm{x}_N^* \neq \mathrm{x}^*$, where x_N^* is the strong local optimum of the objective function, f, and $N \subseteq F$. Figure 2.1 depicts a strong local minimum.

Weak local minimum

A solution is said to be a weak local optimum if $f(\mathrm{x}_N^*) \leq f(\mathrm{x}), \forall \mathrm{x} \in N$, with $x_N^* \neq \mathrm{x}^*$, where x_N^* is the weak local optimum of the objective function, f, and $N \subseteq F$. Figure 2.1 illustrates a weak local minimum.

2.1.5 Single-objective Optimization Problem

Using the concepts discussed above, a SOP is mathematically defined as:

$$\begin{aligned} \text{minimize: } & f(\mathrm{x}) \\ \text{subject to: } & g_i(\mathrm{x}) \leq 0, \quad i = 1, ..., n_g \\ & h_j(\mathrm{x}) = 0, \quad j = 1, ..., n_h \\ & \mathrm{x} \in [\mathrm{x}_{min}, \mathrm{x}_{max}]^{n_x} \end{aligned} \tag{2.1}$$

where n_x is the number of decision variables, $\mathrm{x} = (x_1, x_2, ..., x_{n_x}) \in S \subseteq \mathbb{R}^{n_x}$. Furthermore, g_i, h_j and $\mathrm{x} \in [\mathrm{x}_{min}, \mathrm{x}_{max}]$ are the inequality, equality and boundary constraints respectively.

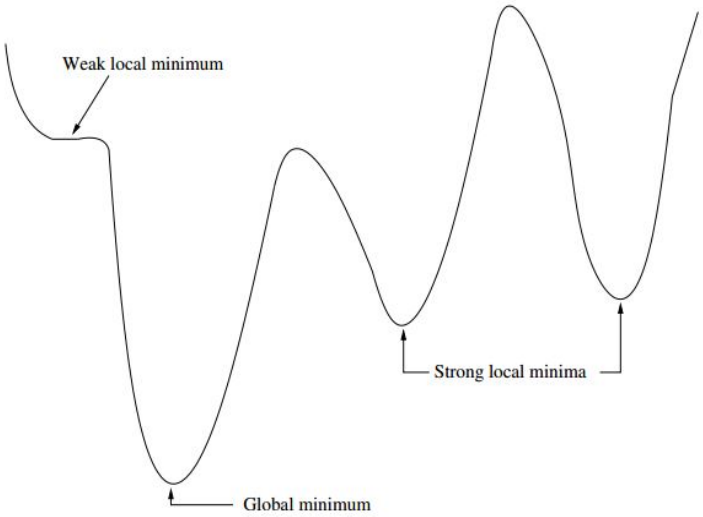


Figure 2.1: Types of optima

2.2 Multi-Objective Optimization

The aim of a MOO is to find a trade-off between multiple objectives. In a real world decision-making process, there exist multiple criteria or objectives. The availability of multiple objectives (which may be in conflict with each other) create a challenge to arrive at a specific or a single solution. In the context of the Forex and stock markets, an investor and decision makers have to make a decision to buy or sell a currency or shares by taking into consideration factors like: how to minimize risk, maximize profit or minimize loss, minimize the cost of trade or transaction, among others. In order to maximize profit, which is the main aim of an investor, more profitable transactions or trades have to be made. However, the more trades you make, the higher the cost of trade or transaction (whether you make profit or loss on a trade).

2.2.1 Multi-objective problems

Unlike a SOP where it is required to minimize or maximize a single objective function or to find only one optimal solution, a MOP consists of more than one objective function, which have to be minimized or maximized, and may be conflicting in nature. Both an MOP and a SOP are subject to a number of constraints. A MOP is mathematically

defined as:

$$\begin{aligned} \text{minimize: } & F(X) = (f_1(x), ..., f_k(x)) \\ \text{subject to: } & g_i(X) \leq 0, \quad i = 1, ..., n_g \\ & h_j(X) = 0, \quad j = 1, ..., n_h \\ & X \in [X_{min}, X_{max}]^{n_x} \end{aligned} \tag{2.2}$$

where g_i, h_j and $X \in [X_{min}, X_{max}]$ are the inequality, equality and boundary constraints respectively.

2.2.2 Pareto-optimal Set and Pareto Optimal Front

The optimal solution or optimum for a SOP is not the same for a MOP. It is much more complicated to solve MOPs, especially when objectives are conflicting. In MOO, a good improvement for one objective might lead to the worsening of other objectives. MOO algorithms need to find a good trade-off between objective functions, hence the task is to find a set of solutions that balance the trade-off, namely the *non-dominated set* or *Pareto-optimal set (POS)*. The *Pareto front (POF)*, on the other hand, is the associated objective vectors of the decision vectors that lead to the POS [27]. The *POS* and *POF* are defined below.

Domination

A decision vector, X_1, is said to dominate a decision vector, X_2 ($X_1 \prec X_2$), if and only if the following conditions are met:

- X_1 is at least as good as X_2 in all objectives, *i.e.* $f_k(X_1) \leq f_k(X_2)$, $\forall k = 1, .., n_k$, and
- X_1 is strictly better than X_2 for at least one objective, *i.e.*, $\exists k = 1, ..., n_k : f_k(X_1) < f_k(X_2)$.

where n_k is the number of objective functions.

Weak Domination

A decision vector, X_1, is said to weakly dominate, $\preceq$, a decision vector, X_2 ($X_1 \preceq X_2$), if and only if:

- X_1 is at least as good as X_2 in all objectives, *i.e.* $f_k(X_1) \leq f_k(X_2), \forall k = 1, .., n_k,$

A decision vector, x^*, is called **Pareto-optimal** when it leads to the best trade-off. In other words, there is no other decision vector, $x \neq x^* \in F$ (feasible space), that dominates x^*. The POS is a set of all Pareto-optimal decision vectors, and the associated objective vectors form the POF.

Pareto-optimality is one of the methods used to solve MOPs. Moreover, one of simplest ways to solve MOPs is by using weighted aggregation. This method assigns weights to each objective function where the weights sum up to 1. It allows for a MOP to be solved as a SOP. However, the main problem is how to determine the best weights for each objective function. For details on both weighted aggregation and Pareto-optimality methods, refer to [27, 36].

Dynamic Environments

Most CI algorithms and their applications, such as PSO applications for the stock and Forex market, assume that the search space or the environment is static. However, the financial market is one of the most complex and dynamic environments in the world, and it is very sensitive to external information, such as economic news [30, 17]. A good optimal TI might perform poorly in the advent of new information.

This Section discusses the concepts of dynamic problems and environments, which include both background information on DMOAs and dynamic SOAs. Section 2.3 discusses the definition and concepts of SOPs and environments. Section 2.4 discusses the definition and concepts of dynamic multi-objective optimization (DMOO) and the various type of environments for DMOO.

2.3 Dynamic Single-objective optimization problem

A dynamic SOP is defined as:

$$\begin{aligned} \text{minimize: } & f(\mathrm{x}, t) \\ \text{subject to: } & g_i(\mathrm{x}, t) \leq 0, \quad i = 1, ..., n_g \\ & h_j(\mathrm{x}, t) = 0, \quad j = 1, ..., n_h \\ & \mathrm{x} \in [\mathrm{x}_{min}, \mathrm{x}_{max}]^{n_x} \end{aligned} \tag{2.3}$$

where n_x is the number of decision variables, $\mathrm{x} = (x_1, x_2, ..., x_{n_x}) \in S \subseteq \mathbb{R}^{n_x}$. g_i, h_j and $\mathrm{x} \in [\mathrm{x}_{min}, \mathrm{x}_{max}]$ are the inequality, equality and boundary constraints respectively.
The aim of a dynamic optimization algorithm is to find:

$$\mathrm{x}^*(t) \quad = \quad min_{\mathrm{x} \in F(t)} f(\mathrm{x}, t) \tag{2.4}$$

where $\mathrm{x}^*(t)$ is the optimum at time step t.
To keep or improve the performance of an optimization algorithm in a dynamic environment such as Forex, one main objective is the continued tracking of the optimum by detecting changes and adapting to the changing environment [27, 47]. The following subsection discuss various classification of dynamic environments.

2.3.1 Eberhart et al.'s Classification

Eberhart et al. defined the following types of dynamic environments [26]:

- **Type I environment**, where the value of the objective function, $f(\mathrm{x}^*, t)$ does not change. However, the location of the optimum, (x^*, t), in the search space is subject to change.
- **Type II environment**, which is the reverse of type I. The location of the optimum, (x^*, t), in the search space does not change. However, the value of the objective function, $f(\mathrm{x}^*, t)$, do change.
- **Type III environment**, where both the location of optimum, (x^*, t), and the objective function, $f(\mathrm{x}^*, t)$, are subject to change.

The severity of change leading to the new location of the optimum is measured by the severity parameter, ζ. The above environment types are summarized in Table 2.1.

Table 2.1: Dynamic Environment defined by Eberhart et al.

	Optimum Location	
Optimum Value	No Change	Change
No Change	Static	Type I
Change	Type II	Type III

2.3.2 Branke's Classification

Branke defined the following characteristics of dynamic environments [13, 12]:

- **Frequency of change**, also referred to as *temporal severity*, which indicates whether changes occur continuously at a regular or irregular time interval (or between evaluations).
- **Severity of change**, which is also known as *spatial severity*, measures how far or close the location of the new optimum (x^*, t) is from its old location (or old optimum's). The changes can be gradual or abrupt.
- **Predictability of change**, indicating whether the changes follow a pattern or trend which can be learned or predicted.
- **Cycle length / cycle accuracy**, that indicates whether the environment returns to its formal (or to similar) state after a certain period of time.

2.3.3 Duhain's Classification

Duhain also proposed new categories of dynamic environment based on temporal and spatial severity [23], namely:

- **Static or quasi-static**, where the environment is considered static if there are no temporal and spatial severity. Moreover, if the environmental changes are insignificant to the scale of the problem and do not also affect the performance of the optimization algorithm, the environment can be considered as static or quasi-static.

Table 2.2: Dynamic Environment defined by Duhain [23, 33].

	Spatial Severity	
Temporal Severity	Low	High
Low	Static	Abrupt
High	Progressive	Chaotic

- **Progressively changing**. This is the type of environment where the **frequency of change** (*temporal severity*) is high or frequent, but with a low or small **severity of change** (*spatial severity*) rate. Progressively changing environments give room for optimization algorithms to utilize previous knowledge to find a nearby optimum. Moreover, static optimization algorithms might still perform well in this environment, since continuous evaluation might lead to the optimum [27].

- **Abruptly changing** is the reverse of a progressively changing environment. There is a higher rate of *spatial severity* with a lower rate of *temporal severity*. Unlike progressively changing environments, knowledge of the previous optimum will not assist in predicting the next optimum. This means static optimization algorithms might perform poorly in an abruptly changing environment.

- **Chaotic**, where both *spatial severity* and *temporal severity* have higher rates of severity. In chaotic environments optimization find it difficult algorithms to track the optimum and to adapt to change, since the algorithms have a small time window to make the required adjustments.

Duhain's types of environments are summarized in Table 2.2.

2.4 Dynamic Multi-objective Optimization

As discussed in Section 2, SOA is required to minimize or maximize a single objective function or find only one optimal solution. MOPs, however, consist of more than one objective function, which have to be minimized or maximized, and may be conflicting in

nature. Just like a SOP, a MOP can also be dynamic, which is the situation in financial markets and other real world domains.

2.4.1 Dynamic Multi-objective optimization problem

A dynamic MOP (DMOP) is defined as:

$$\begin{aligned} \text{minimize: } & F(X,t) = (f_1(\mathrm{x},t), ..., f_k(\mathrm{x},t)) \\ \text{subject to: } & g_i(X,t) \leq 0, \quad i = 1, ..., n_g \\ & h_j(X,t) = 0, \quad j = 1, ..., n_h \\ & X \in [X_{min}, X_{max}]^{n_x} \end{aligned} \tag{2.5}$$

where n_x is the number of decision variables, $\mathrm{x} = (x_1, x_2, ..., x_{n_x}) \in S \subseteq \mathbb{R}^{n_x}$. g_i, h_j and $\mathrm{X} \in [\mathrm{X}_{min}, \mathrm{X}_{max}]$ are the inequality, equality and boundary constraints respectively.
The aim of a dynamic optimization algorithm is not to find a single optimum, but to track the POF over time:

$$(POF^*, t) = F(\mathrm{X}^*(t)) = (f_1(\mathrm{x}^*, t), ..., f_k(\mathrm{x}^*, t)) \; \forall X^* \in (POS^*, t) \tag{2.6}$$

where (POF^*, t) is the optimum POF found at time step t.
The next section presents the different types of dynamic environments for DMOO.

2.4.2 Dynamic Environments for multi-objective optimization

Farina et al. [29] proposed four types of dynamic environments for a MOP, which is similar to what Eberhart et al. [26] introduced, namely:

- **Type I environment**, where the values of the objective functions (POF^*, t) do not change. However, the location of optima or decision variables (POS^*, t) in the search space is subject to change.
- **Type II environment**. In this environment, both the location of optima (POS^*, t) and the values of the objective functions (POF^*, t) are subject to change.
- **Type III environment**. This type of environment is the reverse of type I. The location of the optima (POS^*, t) in the search space does not change. However, the values of the objective functions (POF^*, t) do change.

Table 2.3: DMOO Dynamic Environment defined by Farina et al.

	POS	
POF	No Change	Change
No Change	Type IV	Type I
Change	Type III	Type II

- **Type IV environment**, where both the location of optima (POS^*, t), and the values of the objective functions (POF^*, t), do not change. However, the problem is subject to change.

Farina et al's. types of DMOO environments are summarized in Table 2.3.

Single-Objective Optimisation Algorithms

This Section discusses SOO variants of PSO used in this study. Section 2.5 provides an overview of the fundamentals of PSO, while Section 2.6 discusses the various topological structures used in PSO algorithms. Moreover, Section 2.7 presents different variants of dynamic PSOs used in this study.

2.5 Particle Swarm Optimisation

PSO is a population-based optimisation algorithm, which was inspired by the social behavior of birds within a flock. Particles move within a search space to find an optimal solution. The movement of a particle within the search space or solution space is influenced by its experience (personal best or *pBest*) and the knowledge of its neighbours (global best or *gBest* of the swarm).

2.5.1 Basic Particle Swarm Optimisation

PSO was formally introduced by Kennedy and Eberhart [25]. Each particle (individual) within a swarm (population) represents a candidate solution in the search space. The

position of a particle is represented as an n-dimensional vector and moves within a n-dimensional search space. Each particle has a fitness value, which is used to determine the $pBest$ and $gBest$ which influences movement within the search space. Additionally, each particle has a velocity with which it moves within the search space, since the step size is determined by the velocity.

The position of particle $x_i(t)$ at time step t is updated by adding the velocity $v_i(t)$, to the previous position, $x_i(t-1)$, mathematically defined as:

$$x_i(t) = x_i(t-1) + v_i(t) \tag{2.7}$$

with

$$v_i(t) = \omega v_i(t-1) + c_1 r_{1i}(t)[x_{pBest,i}(t) - x_i(t)] + c_2 r_{2i}(t)[x_{gBest}(t) - x_i(t)] \tag{2.8}$$

where ω is the inertia weight which controls the exploration and exploitation of the swarm [25]. The influence of the personal best, $pBest$, and global best, $gBest$, is controlled by acceleration coefficients c_1 and c_2 respectively. r_1 and r_2 are sampled from a uniform distribution $U(0,1)$.

Information or knowledge sharing amongst particles is based on the topological structure of the PSO. Section 2.6 gives an overview of different variants of neighborhood topologies for PSO.

2.6 Neighborhood Topologies

This section discusses the most widely used PSO neighborhood topologies, namely star, ring and Von Newmann.

2.6.1 Star Topology

The original PSO introduced by Kennedy and Eberhart [25] utilized the star topology. The star topology is where all particles are directly connected to each other as illustrated in Figure 2.2(a). The star topology is also known as the $gBest$ PSO, where all particles move towards the global best (best solution in the swarm) [27]. The star topology is commonly used for PSO implementations (including applications for financial markets),

Algorithm 1 Basic PSO

Create and Initialize an n-dimensional swarm
While *stopping condition is* $false$
  For `each particle i = 1, . . . ,` n_s
// set the personal best
    If $f(x_i) < pBest$
      $pBest \leftarrow x_i$
    EndIf
// set the global best
    If $pBest < gBest$
      $gBest \leftarrow pBest$
    EndIf
  EndFor
  For `each particle i = 1, . . . ,` n_s
  `update the velocity using equation` 2.8
  `update the position using equation` 2.7
  EndFor
EndWhile

since it has the ability to converge faster than other topologies. However, it also has a higher tendency of getting trapped in local minima [27].

2.6.2 Ring Topology

The ring topology was also introduced by Kennedy and Eberhart [25]. The ring topology is also known as the *lBest* PSO, where each particle is connected to the n immediate particle(s) to form a neighborhood. The overlapping of neighborhoods, as depicted in Figure 2.2(b) (n=2), assist in convergence to a single solution. Each particle is attracted towards its best neighbor. Convergence is very slow since information flow or sharing is slow. Moreover, the ring topology covers more of the search space (better exploration) as compared to the star topology, resulting in a quality solution and better performance on multi-modal problems [27].

2.6.3 Von Neumann Topology

The Von Neumann topology was introduced by Kennedy and Mendes [38]. The Von Neumann topology is very similar to the ring topology. However, particles are connected in a grid like structure. Figures 2.2(c) and 2.2(d) depict the 2D and 3D versions of the Von Neumann topology, respectively. Just like the ring topology, the Von Neumann topology's convergence is also very slow, since information flow or sharing is slow, which also reduces the chance of premature convergence.

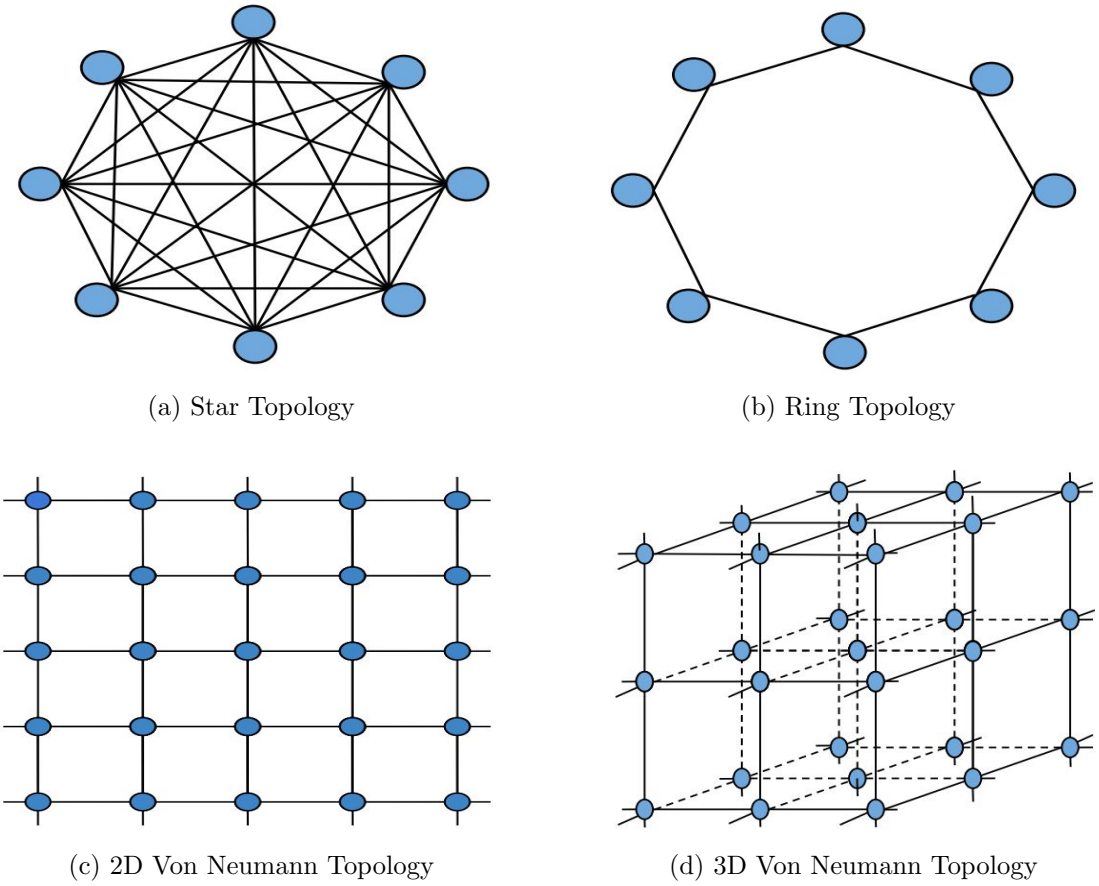
The selection of a PSO neighborhood topology is problem dependent. Each topology has its advantages and disadvantages. Therefore, there is no single topology that can be deemed as the best. However, the Von Neumann topology has outperformed other topologies on a large number of problems [38]. Various other topology forms have been proposed, such as the pyramid, wheel, four cluster, etc. [27].

2.7 Dynamic Particle Swarm Optimisation Algorithms

The standard PSO was not developed for dynamic environments. However, most real world problems are dynamic in nature. To keep or improve the performance of a PSO in a dynamic environment, such as Forex, one main objective is to continue tracking optima by detecting changes and adapting to the changing environment [27, 47]. Most studies use standard PSO and try to fine tune parameters to detect and follow trends in the Forex market. But there is an extent to which PSO parameter tuning limits or promotes optima tracking [27]. In dynamic environments PSO tends to suffer from outdated or stale memory and a loss of diversity [27].

Many techniques have been developed to make the standard PSO self-adaptive to dynamic environments. Some dynamic variant PSOs include: cooperative split PSO [55], predator-prey PSO [54]), cPSO [10] and qPSO [9].

One of the techniques to detect change in a dynamic environment is to use sentry particles [27]. Change is detected by comparing the previously stored sentry particle cost/fitness to its currently evaluated fitness. Change is said to have occurred if there is some difference between the two values. When a change is detected, the algorithm has to react accordingly to increase diversity to enable tracking of the optima. A few simple



(a) Star Topology

(b) Ring Topology

(c) 2D Von Neumann Topology

(d) 3D Von Neumann Topology

Figure 2.2: PSO Neighborhood Topologies

approaches to increase diversity were introduced [27]:

- Do not reinitialize the swarm. This is with the hope that the changes are minor and the swarm has not converged.
- Reinitialize the entire swarm. Here any gained knowledge will be lost.
- Reinitialize part of the swarm. Part of the gained knowledge is retained.

Although the above solutions do not completely eliminate the problem of outdated or stale memory, they assist to increase diversity and exploration [27, 47].

The following subsections discuss the various dynamic PSO algorithms used in this dissertation.

2.7.1 Charged Particle Swarm Optimisation

The cPSO was intruduced by Blackwell and Bentley[10], which is based on the principles of electrostatic energy with charged particles. The goal of the charged particles is to introduce a repulsive force among particles to preserve diversity. At the same time particles are also attracted to the center of mass of the swarm to enable convergence to a solution. The velocity update equation is modified by adding particle acceleration, a_i, as follows:

$$a_i(t) = \sum_{l=1, i \neq l}^{n_s} a_{il}(t) \tag{2.9}$$

where the repulsion between particles i and l is defined as:

$$a_{il}(t) = \begin{cases} \left(\frac{Q_i Q_l}{||df_{il}(t)||^3}\right)(df_{il}(t)) & \text{if } R_c \leq ||df_{il}(t)|| \leq R_p \quad (2.10a) \\ \left(\frac{Q_i Q_l (df_{il}(t))}{R_c^2 ||df_{il}(t)||}\right) & \text{if } ||df_{il}(t)|| < R_c \quad (2.10b) \\ 0 & \text{if } ||df_{il}(t)|| > R_p \quad (2.10c) \end{cases}$$

where $df_{il}(t) = x_i(t) - x_l(t)$, Q_i and Q_l are the charged particles i and l, and R_c and R_p are the core radius and perception limit of each particle i, respectively. Particles with $Q = 0$ are natural particles, and do not contribute to the velocity calculation. In other words, velocity or acceleration is calculated using the standard PSO (equation 2.8). However, particles with $Q > 0$ (charged) contribute to the velocity calculation, and thereby, experience inter particle repulsion from other charged particles. But, charged particles only experience repulsion when the difference between two charged particles falls within $[R_c, Rp]$.

Blackwell and Bentley[10] proposed three types of swarms: a natural swarm, a charged swarm and an atomic swarm. For the purpose of this study, only an atomic swarm, which consists of both charged and natural particles in the swarm, is used, since the atomic swarm performs the best in dynamic environments [10].

2.7.2 Quantum PSO (qPSO)

QPSO is an extension of an atomic swarm version of cPSO, and was also introduced by Blackwell and Bentley[9]. It is based on quantum principles by replacing the orbiting electrons with a quantum cloud. Electrons' movement or position is based on a probability distribution governed by the cloud, and is not based on an electron's previous position or classical trajectory dynamics. The calculation of the position of charged particles does not use a velocity update, but the new position is randomized within a ball of radius r_{cloud}, centered around the global best at every iteration. The difference between cPSO's and qPSO's charged particles is that qPSO's charged particles do not repel from each other and are not influenced by the local and global best. Natural particles follow the classical trajectory dynamics to enable converge to a single solution. The randomization of charged particles might improve the chance that a good solution is useful for natural particles. Hence, having both charged and natural particles within a swarm enables a good balance between exploration and exploitation [9].

Dynamic Multi-Objective Optimization Algorithms

When sloving DMOOs, the aim of MOAs is to keep track of the POF, i.e. be able to detect changes in the environment and respond accordingly. Similar to dynamic SOPs a commonly used technique for detecting a change is the use of sentry particles. Moreover, the algorithms need to respond to the change accordingly. Most responses are geared towards increasing population diversity.

This Section discusses the state-of-the-art DMOAs and DMs for DMOAs. These algorithms are compared against the newly proposed DMOAs in this dissertation. Section 2.8 discusses the components of MOPSO-CD and how the SOO PSO algorithm was adapted for MOPs. Sections 2.9 and 2.10 discuss DNSGA-II and DVEPSO algorithms respectively. Finally, Section 2.11 highlights decision models used to make a decision from a POS.

2.8 Multi-Objective Particle Swarm Optimization with Crowding Distance

The original multi-objective particle swarm optimization (MOPSO) algorithm was proposed by Coello Coello et al [18]. Since its proposal, different variants have been proposed to improve its performance. MOPSO uses an external archive to store non-dominated solutions and a mutation operator to mutate a percentage of the population to introduce diversity. MOPSO was the only MOA to cover the whole POF for the test functions used in [18]. A new variant of MOPSO, namely MOPSO-CD [49], was proposed to increase population diversity. MOPSO-CD incorporates crowded distance into MOPSO to select the global best and to remove solutions with the least crowded distance from the external archive when the archive exceeds its limit. This approach therefore also improves diversity in the external archive. The global best, $gBest$, is selected from the top n percent of the sorted external archive. Algorithm 2 lists the various steps of the MOPSO-CD algorithm. The main components of MOPSO-CD are discussed in more details below.

2.8.1 Archive

The purpose of an archive in MOPSO-CD is to keep track of the found non-dominated solutions at each iteration. Non-dominated solutions are added to the archive based on the following:

- There is pair-wise comparison between the solution of a particle of the main population during initialization. Only the non-dominated solutions are added to the external archive A.
- When the archive, A, reaches its limit, A is sorted in descending order by crowding distance value. Randomly selected particles (e.g. 5 percent) from the bottom of A is removed and replaced with new non-dominated solutions from the population. Sorting according to crowding distance helps introduce diversity into the archive, unlike the original MOPSO algorithm which uses an adaptive grid.

Algorithm 2 MOPSO-CD

Create and Initialize a swarm, n_s
Store the non-dominated particles in Archive, A
While *stopping condition is false*
`Compute the crowding distance for each particle in` A
`Sort the non-dominated solutions in` A `in descending crowding distance values`
  For `each particle i = 1, ...,` n_s
    `Randomly select the gBest from` A
    `Compute the new velocity`
    `Calculate the new position`
    `Manage boundary constraint violations`
    `Perform mutation`
    `Evaluate particle`
  EndFor
  `Update Archive,` A
  For `each particle i = 1, ...,` n_s
    `Update the pBest`
  EndFor
EndWhile

2.8.2 Selecting a gBest or a leader

Adapting a SOO PSO for MOO, one of the challenges is how to select and update the gBest. At each iteration MOPSO-CD selects the gBest from the top of the archive (non-dominated solutions) A, which is the sparse (least crowded) region of the archive, leading to a good balance between exploration and exploitation.

2.8.3 Mutation Operator

The mutation operator of MOPSO was maintained for MOPSO-CD. One of advantages of PSO is its fast convergence. This feature can be good and bad at the same time. Faster convergence can make the algorithm get stuck in a local optimum. The mutation helps

Algorithm 3 MOPSO-CD Mutation Operator

Procedure mutationOperator($particle, dims, curIt, maxIt, mutrate$)
if $flip((curIt/maxIt)^5/mutrate)$ **then**
  $whichdim \leftarrow random(0, dims - 1)$
  $mutrange \leftarrow (upperbound[wichdim] - lowerbound[wichdim]) * (1 - currentgen/totgen)^5/mutrate$
  $ub \leftarrow particle[whichdm] + mutrange$
  $lb \leftarrow particle[whichdm] - mutrange$
  if $lb < lowerbound[whichdim]$ **then**
    $lb \leftarrow lowerbound[whichdim]$
  end if
  if $ub > upperbound[whichdim]$ **then**
    $ub \leftarrow upperbound[whichdim]$
  end if
  $particle[whichdim] \leftarrow RealRandom(lb, ub)$
end if
EndProcedure
Comment: `where particle, dims, curIt, maxIt and mutrate are particle to be mutated, dimensions, current iteration, maximum iterations and mutation rate respectively.`

to prevent the algorithm from producing a false POF. Mutation is applied to all particles from the beginning of the run. Then the rate of mutation is linearly reduced after every iteration. This makes the algorithm explore more at the initial stage and explore less at the later stage. Algorithm 3 highlights the various steps of the MOPSO-CD mutation operator.

2.8.4 Update pBest

Unlike the standard PSO operations, the pBest is the last to be updated. The current pBest is replaced by the new position if the new position dominates the current pBest. However, if none dominates the other, the pBest is randomly selected between the current

pBest and the new position.

2.9 Dynamic Non-dominated Sorting Genetic Algorithm II

The non-dominated sorting genetic algorithm II (NSGA-II) was proposed by Deb et al. [22] to solve the problems other multi-objective evolutionary algorithms faced in relation to computational complexity and non-elitism. NSGA-II introduced a less computational non-dominate sorting approach, which performed better than the compared algorithms on tested benchmarks. However, it was not developed to solve dynamic problems. DNSGA-II is a modified version of NSGA-II for dynamic problems [21]. Individuals are randomly selected from the parent population and re-evaluated to check for change. If a change is detected, a percentage of the population (randomly selected) is either replaced with random solutions, or mutated solutions [22, 21]. This dissertation's study only employs the mutated version of DNSGA-II. Algorithm 4 lists the various steps of the DNSGA-II algorithm. The main components of NSGA-II are discussed in more detail below.

2.9.1 Fast Non-dominated Sorting

Deb el al. [22] introduced fast non-dominated sorting in NSGA-II to speed up the sorting process for combining parents and offspring, R. The fast non-dominated sorting follows the following process:

- The first individual in R is placed in a new population, nP.
- A pair-wise comparison is made between each individual in R, x_i, with individuals in nP, $x_i^{'}$.
- If x_i dominates any individual in nP, $x_i^{'}$ is removed from nP.
- If no individual in R dominates x_i, x_i is placed in nP.

- However, x_i remains in R and is not placed in nP if an individual in R dominates x_i.
- After a pair-wise comparison between all individuals in R, the new population, nP, becomes the first front. The individuals in nP are removed from R and the whole process is repeated to determine the next front until all fronts have been found (R is empty). All individuals within a front is assigned a rank equal to the front number, referred to as the Pareto-rank.

2.9.2 Selecting a New Generation

After each individual has been assigned a Pareto-rank, the crowding distance for each individual is computed. Tournament selection is used to select an individual for the next generation, P_{i+1}. In order to select the next generation, the following criteria are used:

- If P_{i+1} has not reached its limit, individuals are added to P_{i+1} starting from the first front.
- If the remaining individuals in the same front will exceed the limit of P_{i+1}, crowded distance is computed for the front and the individuals in the front sorted in descending order. The top individuals are selected, hence individuals with the highest crowded distance values are selected. This approach introduces more diversity into the set of non-dominated solutions.

Unlike MOPSO, DNSGA-II does not use an archive to preserve elitism. However, selecting the next generation using crowding distance may remove non-dominated solutions from more dense regions of the POF and preserve non-dominated solutions from less dense regions [33].

2.9.3 New solutions

A percentage of the parent solutions is re-evaluated to detect any change in the environment. Deb et al. introduced new solutions into the new generation if a change was detected. One of the following approaches is used to respond to changes:

Algorithm 4 DNSGA-II

1: Create and initialize a random parent population, P
2: Compute and sort parent population based on non-domination level
3: **While** *stopping condition is false*
4: `Select parents for crossover to produce offspring`
5: `Perform offspring mutation`
6: `Check for change`
7: `Combine parents and offspring,` R
8: `Respond to change`
9: `Compute Pareto-ranking for,` R
10: `Sort` R `according to Pareto-ranking`
11: `Select new generation,` P_{i+1}
12: **EndWhile**

- A percentage of R is randomly selected and replaced with new randomly created solutions. This approach does very well in an abrupt environment.
- With the second approach, a percentage of R is mutated. The solutions to be mutated are randomly selected. This approach performs well in environments with a low frequency of change.

2.10 Dynamic VEPSO

DVEPSO is an adapted version of the vector-evaluated particle swarm optimization (VEPSO) algorithm for dynamic environments, proposed by Helbig and Engelbrecht [33]. It is a co-operative approach for solving dynamic MOPs. The population of particles is divided into different sub-swarms, where each sub-swarm optimizes only one objective function and is only evaluated with the same objective fucntion. Information is shared among the sub-swarms by selecting the gBest from a neighbouring sub-swarm according to a knowledge sharing strategy [31]. With the random selection strategy, selecting the gBest can result in selecting the gBest from the same or another sub-swarm. Random

Algorithm 5 DVEPSO

```
Create and Initialize a population and sub-swarms
While stopping condition is false
Check for change
Respond to change
Perform PSO iteration
if new solutions are non-dominated then
  if space in archive then
    add new solutions to archive
    else
    remove solutions from archive
    add new solutions to archive
  end if
end if
Select sentry particles
EndWhile
```

selection of gBest showed better performance than the ring strategy [31]. Therefore, this dissertation's study employs the random selection strategy.

DVEPSO uses sentry particles to check for changes at every iteration, and if a change is detected by a particular sub-swarm, a portion of the swarm is re-initialized or re-evaluated. In addition, the external archive is cleared; or previously non-dominated solutions, that are now dominated are replaced with either the new non-dominated solutions or dominated solutions which became non-dominated solutions through hill climbing. Algorithm 5 presents the various steps of the DVEPSO algorithm.

2.11 Decision making models in MOO

Since the MOO process outputs the POF and POS, and not a single solution or decision vector, decision makers need to select one solution out of a set of solutions to make a decision. There are a number of tools and techniques that can be used to select one

Pareto-Optimal solution from the POS [34, 52, 45]. Unless stated otherwise, the goal of this study is to maximize the objectives. This section discusses some of the most commonly used DMs:

2.11.1 Technique for Order of Preference by Similarity to Ideal Solution

Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) was developed by Hwang and Yoon in 1981 [34], and is one of the most widely used tools for selecting a Pareto-optimal solution. The aim of TOPSIS is to select a point whose Euclidean distance is closer to the ideal solution and far from the worst solution. TOPSIS follows five steps. The pair of weights used in this dissertation's study is [0.5,0.5]. The solution with the highest closeness value, C_i, is used to make a decision. Refer to [34] for more information and formulas. The five steps of TOPSIS are:

Step 1

A m*n normalised objectives matrix, F_{ij}, is created from all objective functions (f^1 and f^2) and solutions, where m is the number of solutions and n is the number of objective functions. The normalised objectives matrix, F_{ij}, is given by:

$$F_{ij} = \frac{f_{ij}^1}{\sum_{i=1}^{m} f_{ij}^2} \tag{2.11}$$

where f^1 and f^2 are function one and two respectively. Note: net profit is f^1, while total transaction cost is f^2.

Step 2

A weighted normalised matrix, wF_{ij}, is created from the normalised objective matrix, F_{ij}. Each objective is assigned a weight, w_j, which determines the value placed on each objective function. The associated weight is assigned based on how the investor or decision maker prioritizes each objective. The associated weights must fall within

[0,1] and sum up to 1. The pair of weights used in this study is [0.5,0.5]. The weighted normalised matrix is defined as:

$$wF_{ij} = F_{ij} * w_j \tag{2.12}$$

Step 3

The ideal, A^+, and worst, A^-, solutions are determined by selecting the best value for each objective from wF_{ij} (in step 2). Since profit and cost are maximized, the maximum value is the best value and the minimum value will be the worst.

Step 4

The Euclidean distance between each solution wF_{ij} in step 2 and the ideal, A^+, and worst, A^-, solution in step 3 is calculated as follows:

$$S_{i+} = \sqrt{\sum_{j=1}^{n}(wF_{ij} - A_j^+)^2} \tag{2.13}$$

$$S_{i-} = \sqrt{\sum_{j=1}^{n}(wF_{ij} - A_j^-)^2} \tag{2.14}$$

where $i = 1, 2, 3, ..., m$. S_{i+} and S_{i-} are the Euclidean distance to the ideal, A^+, and worst, A^-, solutions respectively.

Step 5

Calculate the closeness of each solution using S_{i+} and S_{i-} from step 4 as follows:

$$C_i = \frac{S_{i-}}{S_{i-} + S_{i+}} \tag{2.15}$$

where C_i is the closeness value.

2.11.2 Simple Additive Weighting

Simple additive weighting (SAW) [6] is the sum of the product of the normalised objectives and their weights. The solution with the highest score is selected. The pair of weights, w_j, used in this study is [0.5,0.5]. The SAW value, As, is defined as:

$$As_i = \sum_{i=1}^{n_s} nF_{ij} \ . \ w_j \tag{2.16}$$

where n_s is the number of solutions in the POS and, w_j is the weights associated with each criteria i (objective function). Then, the normalised objective matrix, F_{ij}, is defined as:

$$F_{ij} = \frac{f_{ij}}{f_j^{max}} \tag{2.17}$$

where f_j^{max} are the maximum objective function values / vector for the objective functions, j.
The maximum objective values, f_j^{max}, should not be zero.

2.11.3 Gray Relational Analysis

Gray relational analysis (GRA) describes the gray or similarity between each solution and the best values of each objective. GRA follows three steps: data normalization, ideal sequence and calculating the gray relational coefficient (GRC). The solution with the highest GRC value (when maximizing) is selected as the best solution. GRC is defined as:

$$GRC_i = \frac{1}{n_s} \sum_{i=1}^{n_s} \frac{I_j^{min} + I_j^{max}}{I_{ij} + I_j^{max}}, i = 1, ..., n_s, \ j = 1, ..., m \tag{2.18}$$

where I_j^{max} and I_j^{min} are the maximum and minimum value of the point difference, j respectively. Point difference, I_{ij}, is defined as:

$$I_{ij} = |F_j^{max} - F_{ij}|, \text{with } i = 1, ..., n_s, \ j = 1, ..., m \tag{2.19}$$

where the normalization objective matrix, F_{ij}, is defined as:

$$Fij = \frac{f_{ij} - f_j^{min}}{f_j^{max} - f_j^{min}}, \text{with } i = 1, ..., n_s, \ j = 1, ..., m \tag{2.20}$$

where f_j^{min} and f_j^{max} are the minimum and maximum objective values for objective, j, respectively. n_s and m are the number of solutions in POS and the number of objective functions respectively.

2.11.4 Objective SUM

Objective sum (SUM) is the sum of all objective values, and the highest (when maximizing) value is selected as the best solution. If there is a tie between two or more solutions, one is randomly selected to make the decision. The objective sum is calculated as:

$$S_j = \sum_{i=1}^{n_s} f_{ij}, i = 1, ..., n_s, j = 1, ..., m \tag{2.21}$$

2.11.5 Highest Profit

Highest profit (HPF) is one of the simplest decisions to use. Since the aim of every investor is to make a profit, decision makers will simply pick the solution with the highest net profit.

2.12 Summary

This chapter discussed the concept of optimization and optimization problems for both single-objective and multi-objective optimization. Section 2.1 gave an overview of single-objective optimization and concepts such as the objective function, decision variables, type of optimization solutions and constraints. Multi-objective optimization problems were defined in Section 2.2, as well as the concept of domination.

Moreover, this chapter discussed the various dynamic optimization problems and the types of dynamic environments. Most real world problems are dynamic in nature and optimization algorithms should be able to keep track of changes and adapt accordingly. This chapter introduced both dynamic SOO and SOO concepts in Section 2.3 and 2.4 respectively. Dynamic optimization problems differ in many aspect based on the type of environment. Sections 2.3.1 to 2.3.3 discussed the types of dynamic environments for dynamic SOO and how they affect optimization. Section 2.4.2 addressed DMOO environment types.

This chapter also discussed how PSO operates, since PSO is the foundation of the proposed algorithm DMS-MOPSO. Section 2.5 discussed the standard PSO algorithms. PSOs are affected by the way information is shared amongst particles. Section 2.6 provided an overview of various topological structures. Section 2.7 discussed different variants of dynamic PSO algorithms, proposed for dynamic environments.

Finally, the state-of-the-art DMOAs were also discussed in this chapter. Section 2.8 discussed the components of MOPSO-CD and how SOO PSO was adapted for MOPs. Sections 2.9 and 2.10 discussed the details of DNSGA-II and DVEPSO respectively. The commonly used decision models to select a solution from the PSO for decision making were presented in Section 2.11.

Chapter 3

Optimization Problems and Simulation for Financial Markets

Financial Markets

This Section provides background of the financial markets, especially the Forex market, which is a real world dynamic problem that optimization algorithms introduced in the study will solve. Section 3.1 discusses how the financial market is forecasted. Section 3.2 provides background information of TIs which is used to analyze the trends of the Forex market. Moreover, the trade rules (TRs) regarding TIs are discussed in Section 3.3.

The financial market is a market that investors or people trade in (buy and sell) financial securities, such as stock, bonds, cryptocurrency, commodity markets, Forex, etc. As mentioned in Chapter 1, this dissertation focuses on the Forex market which form part of the financial market. The Forex market is for trading in currencies, which has over 5 trillion USD turnover per day. In addition, the Forex market is one of the most volatile, dynamic and by far the largest market in the world [3]. The most traded currency pairs in the Forex market are EURUSD, USDJPY, and GBPUSD. The next section discusses the traditional tools used in forecasting Forex.

3.1 Financial market forecasting

In order to forecast the financial market, the two most commonly used techniques are fundamental and technical analysis. Fundamental analysis uses factors, which directly and indirectly influence the behavior of market trends. These factors include: interest rate, economic and political news, government and company policies, inflation rates, etc [16, 44]. Technical analysis, on the other hand, uses historical data of the financial market to try and predict the movement of the market [15, 46]. In other words, technical analysis explicitly does not take into account the internal and external influencers of a currency pair or company stock. Fundamental analysis works well for long term investment, while technical analysis works well for short term investment [15].

The EMH states that stock and Forex prices follow a random walk and that any profit due to the study of patterns is obtained by chance. Therefore, using only historical data might not produce much profit and cannot assist with decision making [28]. It further states that the current price of a stock or Forex market takes into account factors that have already influenced the current price. In contrast, investors have made profit with just technical analysis, which is based on past information [15]. Technical analysis analyzes historical price data to find patterns, believing that the price movement pattern repeats itself [15].

Technical analysts believe that fundamental factors that have affected the financial markets have already been incorporated into technical analysis calculations [42]. Most literature (including this dissertation) only concentrate on maximizing profit with TIs (such as moving average (MA) and moving average convergence divergence (MACD)) by finding near optimal parameter values.

3.2 Technical Indicators

TIs are mathematical formulas which uses past data (Forex prices) to generate another time series data. TIs can be grouped into 3 types, namely **trends, momentum and volatility indicators**. Trend indicators, also known as *lagging* indicators, follow the price action. While momentum indicators, which is also known as *leading* indicator depict, the rate of change in price. And finally, volatility indicators display the rapid

change in volatility in price [19]. The study in this dissertation uses four commonly used TIs that are explained below.

3.2.1 Moving Average

MA, which is a lagging indicator, shows the average price within a specific time frame by smoothing the price data which makes the current trend very clear [5]. There are different types of MA. In this study, only two types of MA will be used: exponential moving average (EMA) and simple moving average (SMA). The MAs are defined as:

$$SMA = \frac{\sum_{i=1}^{n} cP_i}{n} \tag{3.1}$$

$$EMA = (cP * per) + EMA(prev) * 1 - per \tag{3.2}$$

with:

$$per = \frac{2}{n+1} \tag{3.3}$$

where n is the number of time frames or period, while cP is the closing price. per is the percentage which determines the weight of recent cP and $EMA(prev)$ is the exponential moving average of the previous day.

3.2.2 Moving average convergence/divergence

MACD is another trends-following momentum indicator and was developed by Gerald Appel [7]. MACD turns two moving average trends into a momentum oscillator by subtracting the long MA from the shorter MA to create the MACD line. The third MA (MACD line) is used to obtain a signal line. The choice of the type of MA depends on the investor or analyst. EMA is used for this study. MACD fluctuates above and below the zero line as the moving averages converge, cross and diverge [5]. MACD is given by:

$$macdL = EMA(cP, n1) - EMA(cP, n2) \tag{3.4}$$

$$sigLine = EMA(macdL, n3) \tag{3.5}$$

where $macdL$ and $sigLine$ are the MACD and signal lines respectively. $EMA(cP, n1)$, $EMA(cP, n2)$, $EMA(cP, n3)$ are exponential moving averages of the closing price, cP, in $n1, n2$, and $n3$ time frames.

3.2.3 Relative strength index

Relative strength index (RSI) is a momentum indicator developed by Welles Wilder [59]. It measures the speed and changes in price movement. RSI relative strength, RS, is calculated by finding the ratio of the average gain divided by average loss. RSI oscillates (or swings) between 0 and 100. By default, RSI is considered overbought when it is greater than 70 and oversold when it is less than 30 [5]. RSI is defined by:

$$RS = \frac{avG}{avL} \tag{3.6}$$

$$RSIv = 100 - \frac{100}{1 + RS} \tag{3.7}$$

with:

$$avG = avG(prev) * (rsin - 1) + cG/rsin \tag{3.8}$$

$$avL = avL(prev) * (rsin - 1) + cL/rsin \tag{3.9}$$

where $rsin$ is the look back parameter or time frame used for the calculation. avG and avL are the average gain and loss in the $rsin$ time frame respectively. $avL(prev)$ and $avG(prev)$ are the previous average loss and gain respectively. RS and $RSIv$ denote the relative strength and relative strength indicator values respectively.

TIs, however, do not operate alone and use rules to generate a signal whether to buy or sell. The following subsection discusses the TRs or strategies.

3.3 Trading Rules or Strategies

TRs or strategies are an important aspect of a trading system. The TR generates a buy or sell signal, determining whether the system should enter the market or not. In the trading system "1" represents a buy signal, "-1" represents a sell signal and "0" signals

to do nothing or hold. Any time the system opens a trade with a buy, it has to exit or close the trade with a sell and vice versa. This approach is motivated by the *"always in the market"* strategy [46]. This means that an exit signal does not only exit the market, but opens a new trade with a sell, of which it also must exit with a buy signal. TRs with respect to the TIs used in this study are implemented as discussed below in Sections 3.3.1 to 3.3.3.

3.3.1 Moving average double crossover

Both SMA and EMA use the double crossover rule. Two MAs are used to generate the moving average double crossover rule, whereby one's time period, n, used for calculation will be short or longer than the other moving average, eg. $SMA(cP, 20)$ and $SMA(cP, 50)$ where 20 and 50 are the specific time periods, n. The following strategy is used to generate a buy/sell signal:

$$signal = \begin{cases} buy, & \text{if } sSMA > lSMA \quad (3.10a) \\ sell, & \text{if } sSMA < lSMA \quad (3.10b) \\ hold, & otherwise \quad (3.10c) \end{cases}$$

where $sSMA$ and $lSMA$ are short and long simple moving averages. The same strategy applies to EMA.

3.3.2 Signal Line Crossover

This is the most popular strategy for MACD, as most traders think it provides better timing [5]. A sell signal is generated when a MACD line, $macdL$, crosses below a signal line, $sigLine$, and vice versa:

$$signal = \begin{cases} buy, & \text{if } sigLine < macdL \quad (3.11a) \\ sell, & \text{if } sigLine > macdL \quad (3.11b) \\ hold, & otherwise \quad (3.11c) \end{cases}$$

3.3.3 RSI Crossover

RSI crossover is a RSI indicator TR. It is similar to other crossover rules, with the only difference being that the buy signal is generated when the RSI value, $RSIv$, is below or cross below the buy limit, lim_b. A sell signal is generated when the RSI value, $RSIv$, crosses over the sell limit, lim_s [5]. The strategy is defined as follows:

$$signal = \begin{cases} buy, & \text{if } RSIv < lim_b \quad (3.12a) \\ sell, & \text{if } RSIv > lim_s \quad (3.12b) \\ hold, & otherwise \quad (3.12c) \end{cases}$$

Related Work

Considerable work has been done to forecast the financial market (especially Forex and stock markets) to maximize profit either with fundamental or technical analysis. Great strives have been made with algorithms, such as Artificial Neural Networks (ANNs), which are widely used to forecast the Forex and stock markets [15, 43]. However, few studies have used computational intelligence techniques, such as PSO and genetic algorithms (GAs) to optimize TIs to maximize profit. Moreover, little research have explored the performance of various dynamic PSO techniques on real world dynamic problems, such as the Forex market. When applying the sPSO algorithm to dynamic environments, it might not be able to detect changes and adapt to these changes over time.

Section 3.4 provides a literature review on SOAs applied to the financial market, while Section 3.5 provides a literature review on the work done applying MOAs to the financial market. Section 3.6 discusses some of the work done on how to make a decision from the POS to make a trade in the financial market.

3.4 Single-objective Optimization for Financial Markets

Little research has explored the performance of various dynamic PSO techniques on real world dynamic problems, such as the Forex market. When applying sPSO to dynamic environments it might struggle to detect and adapt to changes over time.

In [57], the authors developed a stock trading strategy based on the tPSO algorithms by combining MA and trading range break-out (TRB) TIs. 140 TRs were created from both MA and TRB TIs, of which each TR was assigned a starting weight. A single signal was generated by summing the weights and signals. The tPSO was used to optimize buy and sell thresholds. The system outperformed both TRB and MA best strategies. However, parameters for all 140 TRs were set manually. This approach will lose valuable information from other parameter values which were not included for MA or TRB TI. Additionally, the system was trained once on training data and the best optimal values after training were used for the rest of trading. Such a technique in a Forex market might lose performance when using previously optimal parameters. This is because the environment may change over time, hence the nature of a dynamic environment. The tPSO does not really detect or adapt to changes in the search space, since the algorithm does not make use of information from the search space or particles to detect changes [56].

Butler and Kazakov [14] used sPSO to find parameters for the bollinger bands (BB) TI for the stock market. They solved the problem of manually setting parameters values for TIs [57]. However, sPSO was not adapted for a dynamic environment. Since the stock market is a dynamic market, sPSO will struggle to track optima or trends. Furthermore, using a single indicator does not give a clear picture on how PSO can optimize TIs with respect to the changing nature of the market.

Some authors have also used GAs to optimize several TIs [46]. In the study of [46] 24 TIs were used to generate 38 TRs and a GA was used to find the best parameters, which is an extension of the work of Butler and Kazakov [14]. The model was trained with a constant set of data and then tested with another set of data. The disadvantage of this approach is that when the search space of the test data completely changes during testing, the model will underperform.

Chen and Huang [16] tried to address the problems [39, 46, 57, 58] that this approach faced, by using a modified PSO to forecast the Forex market by implementing a sliding window to track changes in the training and test data. Fitness inheritance was also implemented to make use of previous or historic data of particles or solutions. 13 currency pairs and 10 economic indicators were used as input (independent variables) to predict

the exchange rate of the NTDUSD currency pair (dependent variable) by experimenting with different sizes of sliding windows and number of particles. A prediction accuracy of approximately 70 percent was achieved with a window size of 25 days. Since the model was not used to predict different currency pairs, it is difficult to know how it will perform when predicting different currency pairs. Although the focus of the study was not to optimize or use TIs as [39, 46, 57, 58] did, the model might yield good results when applied to the problem of optimizing TIs.

The main objective of solving a dynamic environment problem is to be able to keep track of the changing optima and to adapt to the changing environment [27]. Most literature did not explore the possibility of tracking and identifying changes and making the algorithm respond to the changes. Some authors used a sliding window to adapt to changes in data, but the algorithm did not detect and adapt to changes [16, 43]. Fine tuning the sPSO parameters to track and adapt to changes assumes that the swarm did not converge. When the particles converge, the contribution of the cognitive and social components is negligible [27]. Therefore, fine tuning the sPSO parameters for the Forex market, which is very dynamic, might result in reduced performance over time.

3.5 Multi-objective Optimization for Financial Markets

Some authors have used SOAs to successfully select parameters for TRs to maximize profit [16, 14, 46]. This approach helps to solve the problem of having to manually set each parameter [57]. However, the algorithms used to select parameters for the TRs or strategies were designed for static environments. Atiah and Helbig [8] applied quantum particle swarm optimization, which is a dynamic algorithm, to the Forex market, and it showed good performance in comparison to static optimization algorithms. However, like many real-world problems, the Forex and stock markets have more than one objective to be minimized or maximized.

Lohpetch and Corne [41] adapted the genetic programming (GP) algorithm for a MOP, applied it to the stock market and compared it against a single-objective GP. The downside was that TI parameters were manually configured. Diago et al [11] also used

a multi objective evolutionary algorithm with super individual (MOEASI) for the stock market and compared it against single-objective evolutionary algorithm (EA). However, the a single-objective EA performed better than MOEASI.

Ricardo, et al [20] used a variant of a differential evaluation (DE) MOA, called spherical pruning multi-objective differential evolution (spMODE), to optimize four TIs to maximize profit, minimize the number of trades (which has an impact on transaction cost) and minimize risk. The performance of spMODE was compared against some default values of the respective TIs used and the buy and hold strategy. spMODE's optimized TIs outperformed all default valued TIs and the buy and hold strategy. However, transaction cost was not considered or optimized directly. The authors assumed minimizing the number of trades will lead to low transactional cost.

Antonio and Prospero [39] generated TRs by combining five optimized TIs' signals with the MOPSO-CD) with two objectives: percent profit and Sharpe ratio. The results were compared against the NSGA-II, which is one of the most widely used MOAs. MOPSO-CD outperformed NSGA-II in both the testing and training data. Moreover, MOPSO-CD also outperformed the buy and hold strategy in two out of three testing and training periods.

3.6 Decision Models' Effects on MOO for Financial Markets

Many studies have addressed financial time series forecasting with computational intelligence techniques, like ANNs and optimization [15]. However, one of the main challenges with MOO for automated trading is to make a good decision to place an order or trade from the POS. Making a decision for SOO is easy since the algorithm only outputs a single solution.

Ricardo et al [20] used a variant of a DE MOA, called spherical pruning multi-objective differential evolution (spMODE), to optimize four TIs to maximize profit, minimize the number of trades (which has an impact on transaction cost) and minimize risk. The performance of spMODE was compared against some default values of the respective TIs used and the buy and hold strategy. spMODE's optimized TIs out-

performed all default valued TIs and the buy and hold strategy. However, no DM was used. The authors analyzed all solutions on test data and selected the best solution for comparison.

Lohpetch and Corne [41] adapted a GP algorithm for a MOP, applied it to the stock market and compared it against a single-objective GP. The authors addressed the decision problem by employing the majority-voting approach (MJV) amongst points (solutions) [20]. The multi-objective GP showed very good performance against the single-objective GP. The problem with the authors' decision making approach is that, if under performing solutions form the majority, the system will make losses and vice versa.

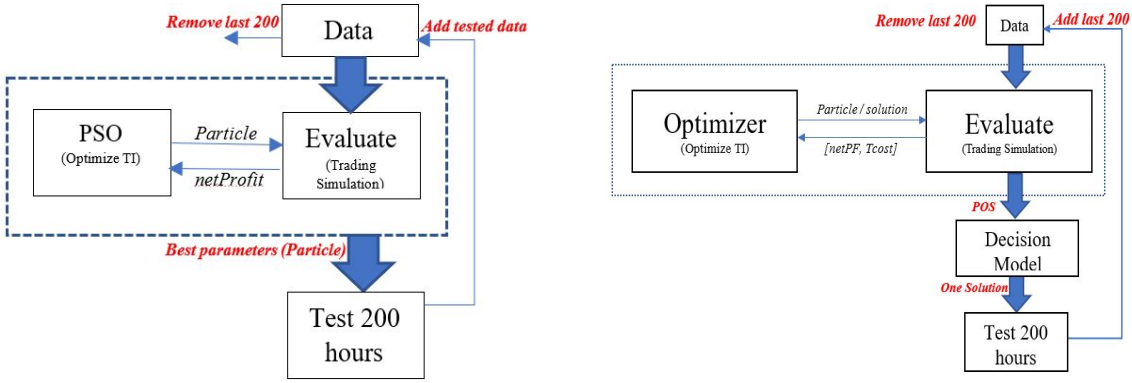
Antonio and Prospero [39] used the bagplot [51] to analyze the POF and the depth mean to select a solution. This approach may not guarantee the best solution, since other regions of the Pareto front were not analyzed.

The financial market is a dynamic market, especially Forex, which is the focus of this study. However, CI techniques applied to the Forex and stock markets, including the above reviewed literature, normally are static algorithms. The Forex market is traded 24 hours per day from Monday to Friday. The advent of new data or information may change the behavior of the market [17, 30]. In order to make profitable trades, the investor or any automated trading system should be able to keep track of the trends (price movements) at all times. Directly applying static algorithms to the Forex market might lead to a degrading performance of the algorithm over time.

Some literature have used a sliding window in the single-objective space to keep track of changes in the data [16]. However, the algorithms were not used to detect changes and adapt to changes. Moreover, in the context of MOPs, the system (or algorithm) should not only track a single optimum as for SOPs, but has to track a set of optima (solutions) for an investor or an automated trading system to make a decision [27].

Experimental Setup

The Section provides details about the experimental setup for this study. The main aim of this study is to explore the performance of SOAs, MOAs and the proposed DMS-



(a) Trading System Overview for SOA (b) Trading System Overview for MOA

Figure 3.1: Trading System Overview

MOPSO algorithm on a dynamic financial market, namely the Forex market.

The rest of the Section is organized as follows: Section 3.7 discusses the simulated trading system for the experiment. The general experimental setup for this study is then discussed in Section 3.8.

3.7 Financial Simulations

This section provides information about the trading system, including the data used for this experiment. Moreover, the sliding window mechanism and evaluation model used are discussed.

3.7.1 Trading System

This section discusses the various components of the trading system, used to optimize TIs. Figure 3.1 gives an overview of the whole system for both SOO and MOO algorithms.

The main aim of the study has to find or select a good combination of parameter values for various TIs. The performance of a TR is largely based on its parameter combination. Discretized versions of the optimization algorithms are used to select near optimal parameter values.

Table 3.1: Currency pairs with volatility rates on both training and testing data sets

Currency Pair		Training	Testing
EURGBP	Hourly	0.1225	0.1308
	Annual	1.9443	2.0757
USDZAR	Hourly	0.2266	0.1808
	Annual	3.5971	2.8696
EURUSD	Hourly	0.1184	0.1271
	Annual	1.8802	2.0175
USDJPY	Hourly	0.1190	0.1338
	Annually	1.8885	2.1238

3.7.2 Data

The data used for the experiment is the Forex hourly closing price data from January 2014 to December 2017 for all currency pairs used. The selection of currency pairs are categorized into three types, namely **major**, **exotic** and **crosses** currency pairs [1]. Major currency pairs are the most traded pairs and are paired with the United States Dollar (USD). The major currency pairs used for this experiment is the **EURUSD** and **USDJPY**. The currencies of countries that depend heavily on commodity export, like gold and silver, are referred to as commodity currencies. Examples are Australia and New Zealand. However, real commodities, such as silver and gold, are also paired with the USD and traded. **XAGUSD** and **XAUUSD** are examples of such commodities. Crosses currency pairs are currency pairs that are not paired with the USD. The one used for this study is the **EURGBP** currency pair. Lastly, exotic currency pairs are the pairing of the currencies of developing countries with developed countries. The **USDZAR** is the exotic currency pair selected for this study. Data was exported from MetaTrader 5. Table 3.1 lists all the currency pairs with their respective hourly and annual volatility rates [4]. The selected currency pairs with different rates of volatility provided different data dynamics to test the algorithms.

Figures 3.2 to 3.5 depict the actual price movement for both the training and testing datasets.

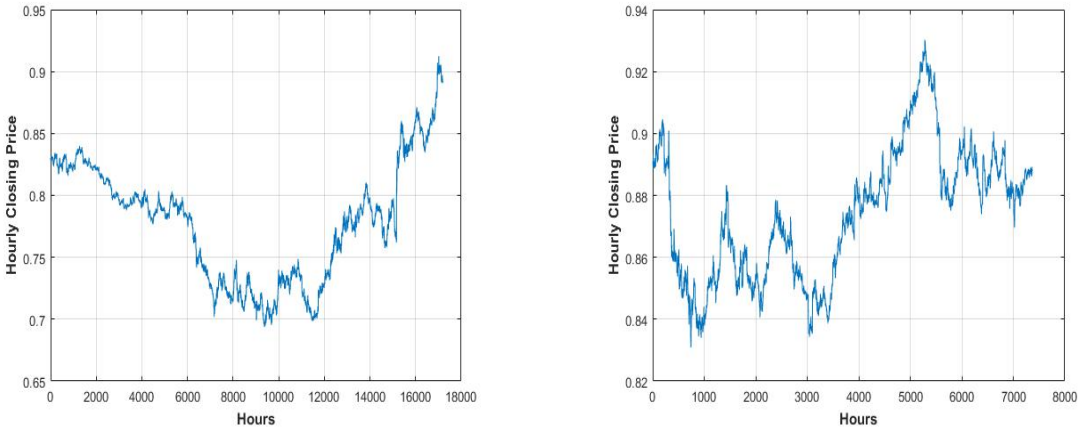
3.7.3 Sliding window

A sliding window was employed to assist the algorithm to adapt to the dynamic nature of the data-set [16]. As the window slided, the furthest data (training data set) was removed and the recently used test data added to the window. The window was made up of a training and testing data-set, of which 70 percent was training data and n hours were testing data, as depicted in Figure 3.6.

The model was trained with training data and the attained POS was passed through the decision model to select a solution. The selected solution was then applied to the next 200 hours of trading data (testing). The sliding window then moved forward to include the next n hours' records of testing data and excluded the last n hours' records of training data. This process continued until the end of the testing data was reached.

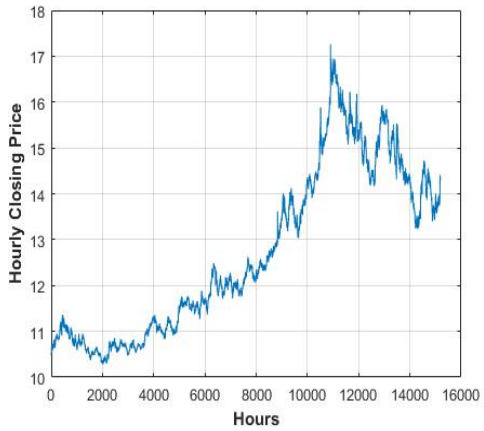
In the case of MOO, as depicted in Figure 3.1(b), the model was trained with training data and the attained POS was passed through the decision model to select a solution. The selected solution was applied to the next n hours of trading data (testing). As a result, the model was re-trained after every n hours of trade to accommodate any dynamic changes.

A window size of 200 was chosen based on experimental analysis of the performance of various optimization algorithms for other window sizes (50, 100, 200, 250, 300, 350

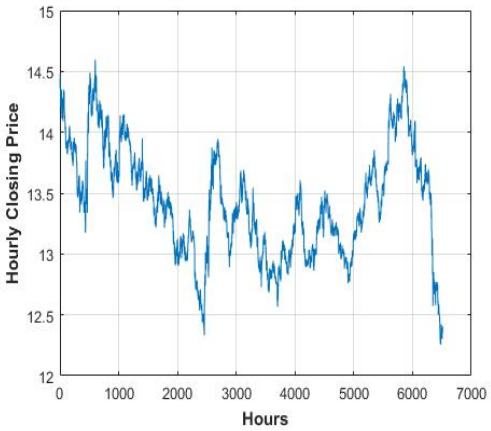


(a) EURGBP training (b) EURGBP testing

Figure 3.2: Actual price movement of EURGBP dataset

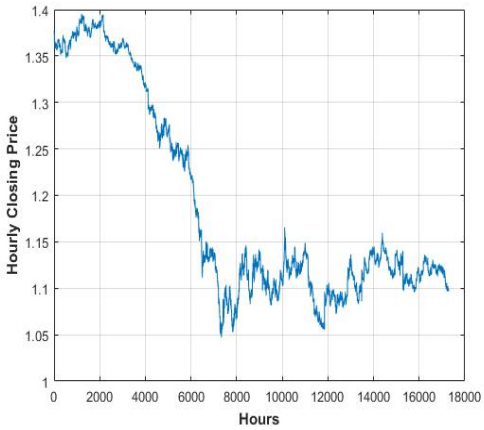


(a) USDZAR training

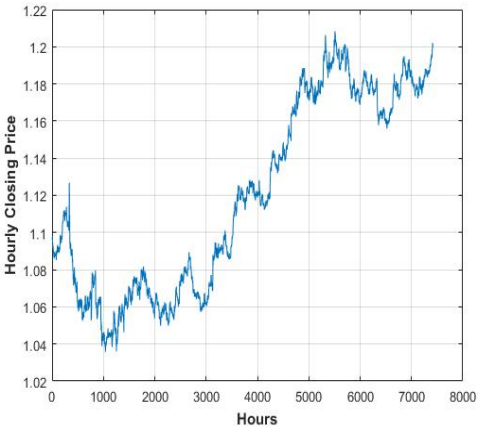


(b) USDZAR testing

Figure 3.3: Actual price movement of USDZAR dataset



(a) EURUSD training



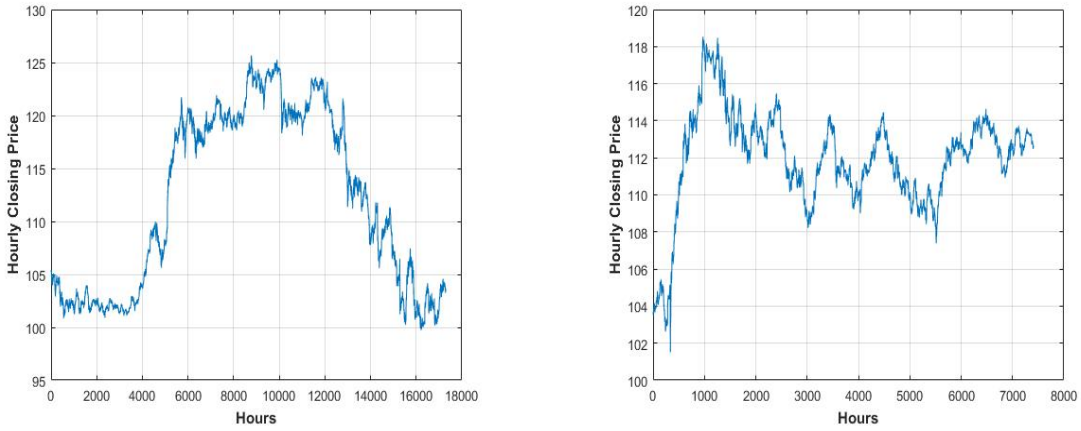
(b) EURUSD testing

Figure 3.4: Actual price movement of EURUSD dataset

and 400), since a window size of 200 generally performed better for all algorithms.

3.7.4 Evaluation model

Due to *"always in the market"* as discussed in Section 3.3, the system entered and exited at the same time. Algorithm 6 simulates the net profit and transaction cost. If the previous signal was "buy", it could either be opening or closing a market. But more



(a) USDJPY training (b) USDJPY testing

Figure 3.5: Actual price movement of USDJPY dataset

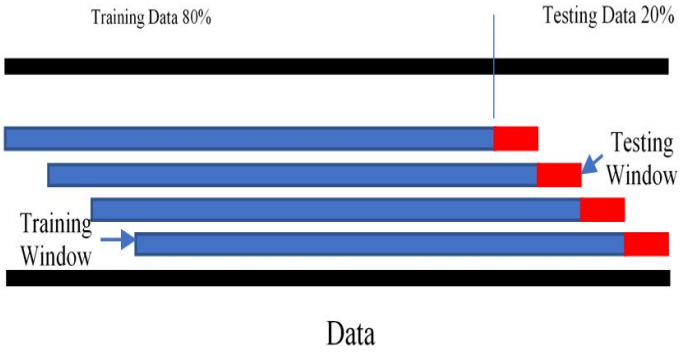


Figure 3.6: Sliding window

importantly, buying is to go long, i.e. to buy at a low price and sell when the price is high, by subtracting the previous price from the current price. On the other hand, when a signal was sell, the current price was subtracted from the previous price. On the other hand, when a signal was sell, it signified going short or by borrowing at the current price (50 USD) with the hope that the price would drop to enable buying at a lower price (45 USD) and paying back the loan by keeping the difference in price value. That is, by subtracting the current low price (45 USD) from the previous high price (50 USD) and keeping a profit of 5 USD.

Algorithm 6 Pseudo code for returns (gain and loss), net profit and transaction cost.

Procedure Returns($sigArray, cP, tCost$) {*Signal and Price Array*}
$prevPrice \leftarrow 0$ {*Entering the trade Price*}
$prevSignal \leftarrow 0$ {*Previous signal*}
$pF[size(cP)]$ {*Array of returns*}
$tCost[size(cP)]$ {*Array of transaction cost*}
$T_n \leftarrow 0$ {*Total Number of transactions*}
While $i \leq size(sigArray)$
if $sigArra[i] =$ "1"&$preSignal =$ " $-$ 1" **then**
  $prof \leftarrow prePrice - cP[i]$
  if $prof > 0$ **then**
    $tCost[i] \leftarrow tCost * prof$
    $pF[i] \leftarrow prof - tCost[i]$
      Else
    $tCost[i] \leftarrow tCost * (-pF[i])$
  end if
  $preSignal \leftarrow$ "1"
  $prePrice \leftarrow cP[i]$
  $T_n \leftarrow T_n + 1$
    ElsIf $sigArra[i]$" $-$ 1"&$preSignal =$ "1"
  $pF[i] \leftarrow cP[i] - prePrice$
  if $prof > 0$ **then**
    $tCost[i] \leftarrow tCost * prof$
    $pF[i] \leftarrow prof - tCost[i]$
      Else
    $tCost[i] \leftarrow tCost * (-prof)$
    $pF[i] \leftarrow prof - tCost[i]$
  end if
  $preSignal \leftarrow$ " $-$ 1"
  $prePrice \leftarrow cP[i]$
  $T_n \leftarrow T_n + 1$
end if
$i = i + 1$
EndWhile
$netPF \leftarrow sum(pF)$ {net profit}
$T_{Cost} \leftarrow sum(tCost)$ {Total transaction cost}
if $Tn > 0$ **then**
  $avgPF \leftarrow netPF/T_n$ {average profit}
    Else
  $avgPF \leftarrow 0$ {average profit}
end if
$tGains \leftarrow sum(pF(pF > 0))$
$tLoss \leftarrow sum(pF(pF < 0))$
return $netPF, avgPF, tLoss, tGains, Tn, pF, T_{Cost}$
EndProcedure

Table 3.2: Solution representation

SMA		**EMA**		**RSI**			**MACD**		
n	n	n	n	lim_b	lim_s	n	$n2$	$n3$	$n1$
20	100	20	100	25	71	20	50	250	20

3.8 Optimization Setup

This section discusses how the optimization algorithms used in this study were adapted to optimize the Forex market and how the problem was represented. The algorithms were implemented in MATLAB. All experiments were run 30 times per algorithm, and an average of the algorithms' performance values was calculated.

3.8.1 Solution representation

The trading system used the algorithms to select good parameters for each TI. Each solution vector was encoded as depicted in Table 3.2. There were 10 dimensions, each representing a parameter of all 4 TIs to be optimized. The short and long time frames of SMA and EMA make up four dimensions, two dimensions each for both SMA and EMA TIs. RSI on the other hand occupies three dimensions; representing the buy limit, sell limit and time frame respectively. Additionally, MACD occupies the remaining three dimensions, each representing all the three time frames used for MACD and signal lines in signal line crossover strategy. In terms of the DNSGA-II, chromosomes were made up of four blocks of genes, each representing parameters of a specific TI.

3.8.2 Velocity and Boundary Handling

The velocity (for PSO algorithms) and position for particles for all algorithms were clamped to prevent them from overshooting optima and moving out of the search space boundary. Velocity was set to the maximum velocity, v_{max}, or minimum velocity, v_{min}, when the particle's velocity was greater or less than the set velocity limits [27]. A particle's velocity was modified before any position update [27]:

$$v_i(t) = \begin{cases} \acute{v_i}(t), & \text{if } v_{min} \leq \acute{v_i}(t) \leq v_{max} \quad (3.13a) \\ v_{min}, & \text{if } \acute{v_i}(t) < v_{min} \quad (3.13b) \\ v_{max}, & \text{if } \acute{v_i}(t) > v_{max} \quad (3.13c) \end{cases}$$

where $\acute{v_i}(t)$ is the newly calculated velocity. The minimum and maximum velocities of particles were determined by:

$$v_{max} = \delta(D_{max} - D_{min}) \quad (3.14)$$

$$v_{min} = -(v_{max}) \quad (3.15)$$

where D_{min} and D_{max} denotes the minimum and maximum values of the domain for each TI's parameter, and $\delta \in (0, 1]$ [27].

Furthermore, the position of a particle was modified when the position was outside the domain, as follows:

$$x_i(t) = \begin{cases} \acute{x_i}(t), & \text{if } D_{min} \leq \acute{x_i}(t) \leq D_{max} \quad (3.16a) \\ D_{min}, & \text{if } \acute{x_i}(t) < D_{min} \quad (3.16b) \\ D_{max}, & \text{if } \acute{x_i}(t) > D_{max} \quad (3.16c) \end{cases}$$

where $\acute{x_i}(t)$ is the newly calculated position.

Offspring of DNSGA-II were also restricted according to Equation 3.16.

3.8.3 Discrete Solutions

Computational algorithms are typically used to solve continuous optimization problems. However, the algorithms employed in this study were adapted for discrete problems. To optimize TIs or to find better combinations of parameters for TRs which work on discrete time series of prices, all algorithms used in this study were adapted for the discrete search space. The algorithms were adapted as follows:

PSO particles were randomly initialized with a discrete uniform distribution, unlike the sPSO where particles are randomly initialized with a continuous uniform distribution. The velocity for PSO algorithms was still calculated with Equation 2.8. However, the

result was passed through a hyperbolic tangent function and rounded up to the nearest integer to attain a new discrete velocity value, $v_{i,d}(t)$. A particle's position at time step, t, was therefore given by [48]:

$$\acute{x_i}(t) = x_i(t) + round(tanh(v_i(t))) \tag{3.17}$$

where $\acute{x_i}(t)$ is the newly calculated position, i.e. the position at time t.

The mutant vector and offspring of DNSGA-II were rounded up to the nearest integer.

3.8.4 Objective Function

The fitness of a SOO PSO's particle was defined as the net profit, $netPF$, over a trading period, which is the sum of returns (losses and gains). The net profit determined which particle was used in testing or for trading as depicted in Figure 3.1.
For MOO, total transaction cost, T_{Cost}, was employed to have conflicting objectives functions. Therefore, the following objectives were used:

- **Net profit ($netPF$)**: It is the sum of returns (losses and gains) after deduction of the cost of transaction. The objective is to maximize the $netPF$, given by:

$$netPF = \sum_{t=2}^{T} pF_t \tag{3.18}$$

- **Total transaction cost (T_{Cost})**: The second objective is to minimize the T_{Cost}. An investor or trader is charged a percentage per returns. Since we are always in the market as discussed in Section 3.7.1, the charges are applied to all returns (either gains or losses). The system needs to make profitable trades in order to cover T_{Cost} and make profit. A percentage of 2.5 was charged as $tCost$ per return in this study. T_{Cost} is given by:

$$T_{Cost} = \sum_{t=2}^{T} tCost_t \tag{3.19}$$

where T_{Cost} and $netPF$ are the net profit and total transaction cost respectively. pF_t and $tCost_t$ are the traders' returns per trade (on gain or loss) and transaction cost at time t, respectively, which are calculated based on Algorithm 6.

Note: T_{Cost} is negated to turn it into a maximization objective function.

3.9 Summary

This chapter provided background information on the financial market. Section 3.1 emphasized the fundamental and technical analysis and how they are used to forecast the financial market. Section 3.2 further explained the most commonly used technical indicators, which are used to forecast the financial market. Moreover, the respective trading rules associated with technical indicators to make trades were highlighted in Section 3.3.

A review of work done on the application of computational intelligence algorithms to the financial market was also presented in this Chapter. Section 3.4 reviewed SOAs applied to the financial market, of which most literature assumed the financial market as a static environment. Since the financial market is not a SOP, Section 3.5 discussed work that applied MOAs to the financial market. Moreover, Section 3.6 highlighted how some studies made decisions from the POS to make a trade in the financial market.

Moreover, this chapter provided details on the experimental setup for this study. Section 3.7 discussed the simulated trading system for the experiment. Section 3.7 also provided details of the sliding window mechanism used and the evaluation model employed for this experiment. Section 3.8 discussed the general experimental setup for this study.

Chapter 4

Dynamic Particle Swarm Optimization for Financial Markets

Most PSO applications for the stock or Forex market assume that the search space or the environment is static. However, the financial market is one of the most complex and dynamic environments in the world. This chapter focusses on exploring the performance of various dynamic PSOs when maximizing profit in a complex dynamic environment, such as the Forex market. Algorithms, such as the qPSO and cPSO, are employed and compared with the performance of the sPSO algorithm and the tPSO algorithm, which are usually used when optimizing TIs for the stock and Forex markets [58, 16].

The rest of the chapter is organized as follows: The description of the conducted experiment is detailed in Section 4.1. Section 4.2 then discusses the results obtained from the experiments.

4.1 Experimental Setup

This section describes the experimental setup of the experiments discussed in this chapter. The data used for this experiment were USDJPY and USDZAR currency pairs discussed in Section 3.7.2. Moreover a sliding window mechanism was used as depicted in Figure 3.6 and as discussed in Section 3.7.3. The rest of this section's layout is: Section 4.1.1 presents the representation used for the particles. The objective function

is discussed in Section 4.1.2 and parameter configuration is discussed in Section 4.1.3. Finally, performance measures are highlighted in Section 4.1.4.

4.1.1 Particle representation

The trading system used a PSO to select good parameters for each TI. Particles were encoded as depicted in Table 3.2. There are 10 dimensions, each representing a parameter for all 4 TIs to be optimized.

4.1.2 Objective Function

The fitness of a particle was defined as the net profit, $netPF$, over a trading period, which is the sum of returns (losses and gains). The net profit determined which particle should be used in testing or for trading as depicted Figure 3.1(a). Therefore, the objective function used was the net profit, $netPF$, defined in Equation 3.18.

4.1.3 Parameter Configuration

The parameters' values in Table 4.1 were selected according to [10, 9, 50]. The performance of a PSO is largely influenced by a good balance between exploration and exploitation. To be able to get a good trade off, the inertia weight, ω_{min}, was linearly reduced as suggested by Shi and Eberhart [25], to enable exploration at the initial stage and exploitation at the later stage. Furthermore, the acceleration coefficient, c_1, decreased linearly overtime, while the acceleration coefficient, c_2, increased linearly [50]. The values of $c_1(t)$, $c_2(t)$ and ω_t at time step t were calculated as:

$$c_1(t) = \left(c_{1,min} - c_{1,max}\right)\frac{t}{n_t} + c_{1,max} \tag{4.1}$$

$$c_2(t) = \left(c_{2,max} - c_{2,min}\right)\frac{t}{n_t} + c_{2,min} \tag{4.2}$$

$$\omega(t) = \left(\omega_{min} - \omega_{max}\right)\frac{t}{n_t} + \omega_{max} \tag{4.3}$$

where n_t is the maximum number of iterations. t, min and max denotes the current iteration, minimum value and maximum values respectively. The parameters' respective values can be found in Table 4.1.

Table 4.1: Parameter settings of PSO and the boundaries of technical indicators

PSO	Values	Technical Indicators	Domain,D
ω_{min}	0.4	n	[1,250]
ω_{max}	0.9	lim_b	[1,50]
$c_{1,min}$	0.5	lim_s	[51,100]
$c_{1,max}$	2.5	$n3$	[1,9]
$c_{2,min}$	0.5	$n1, n2$	[10,50]
$c_{2,max}$	2.5		
Q, r_{cloud}	$0.15 * (D_{max})$		
δ, R_c, R_p	0.1, 1, $\sqrt{3D_{max}}$		

Note that linearly increasing c_2 and decreasing ω and c_1 was only applied to the tPSO, cPSO and qPSO. Parameters of sPSO remained unchanged after every iteration. tPSO is the same as sPSO, except that tPSO linearly increased c_2 and decreased ω and c_1 after every iteration.

Velocity and boundary constraints were handled as explained in Section 3.8.2

4.1.4 Evaluation measures

Since a sliding window was employed, returns were aggregated over all windows within a run. An average of the net profit, Avg_{netPF}, over all 30 independent runs was calculated. However, in order to measure the consistency of the algorithms' performance, the coefficient of variation (CV) was used. CV is a statistical measure to compare the degree of variation from one data series to another, no matter how different the averages are from one another[2].

The results from the study are discussed in the following section.

4.2 Experimental Results

This section discusses the results obtained from the study. Sections 4.2.1 and 4.2.2 discuss the performance of all algorithms over the USDJPY and USDZAR currency pairs

without transaction cost, respectively. Sections 4.2.3 and 4.2.4 discuss the performance of all algorithms over the USDJPY and USDZAR currency pairs with transaction cost, respectively. Moreover, Section 4.2.5 outlines the general observations made throughout the experiment.

4.2.1 USDJPY Without Transaction Cost

Tables 4.2 and 4.3 summarize the outcomes of the study on training and testing data for the USDJPY currency pair without transaction cost respectively. Tables 4.2 and 4.3 show the average net profit, Avg_{netPF}, and CV values for the various TIs and all four algorithms. Moreover, Figure 4.1 presents the accumulated sum returns for various TIs by all four algorithms over the USDJPY currency pair without transaction cost. qPSO showed good performance over time, while all other algorithms depreciated in performance over time.

From Table 4.2's training results, it can be seen that qPSO yielded very good results for both net profit and CV on all TIs, except for MACD, for which tPSO outperformed all other algorithms. qPSO consistently was the second best performer on all TIs, except for its worse performance for CV on RSI. Furthermore, sPSO yielded the least net profit on MACD, EMA and RSI. However, it performed well on the SMA. All algorithms generally performed well on the training data. However, the trends were different for the testing data.

As shown in Table 4.2's testing results, on SMA qPSO was the only algorithm that yielded positive returns for both net profit and CV. qPSO continued to produce superior profit on both EMA and RSI, and was followed by qPSO. However, tPSO and sPSO made the least profit on EMA and RSI respectively. As depicted in Figures 4.1(a), 4.1(b) and 4.1(d), qPSO was more superior than other algorithms on both the training and testing datasets. However, Table 4.3's testing results and Figure 4.1(c) indicate that all algorithms performed better on MACD, with qPSO producing the highest profit, followed by qPSO.

Table 4.2: Algorithms' evaluation without cost on USDJPY training dataset

TIs	Measures	sPSO	tPSO	cPSO	qPSO
SMA	Avg_{netPF}	36.697	35.838	37.391	**46.991**
	CV	0.141	0.116	0.135	**0.114**
EMA	Avg_{netPF}	34.161	33.956	34.054	**37.114**
	CV	0.049	0.057	0.046	**0.018**
RSI	Avg_{netPF}	30.462	30.249	31.605	**37.567**
	CV	0.221	0.213	0.224	**0.181**
MACD	Avg_{netPF}	33.866	**34.115**	34.019	33.877
	CV	0.022	**0.011**	0.014	0.021

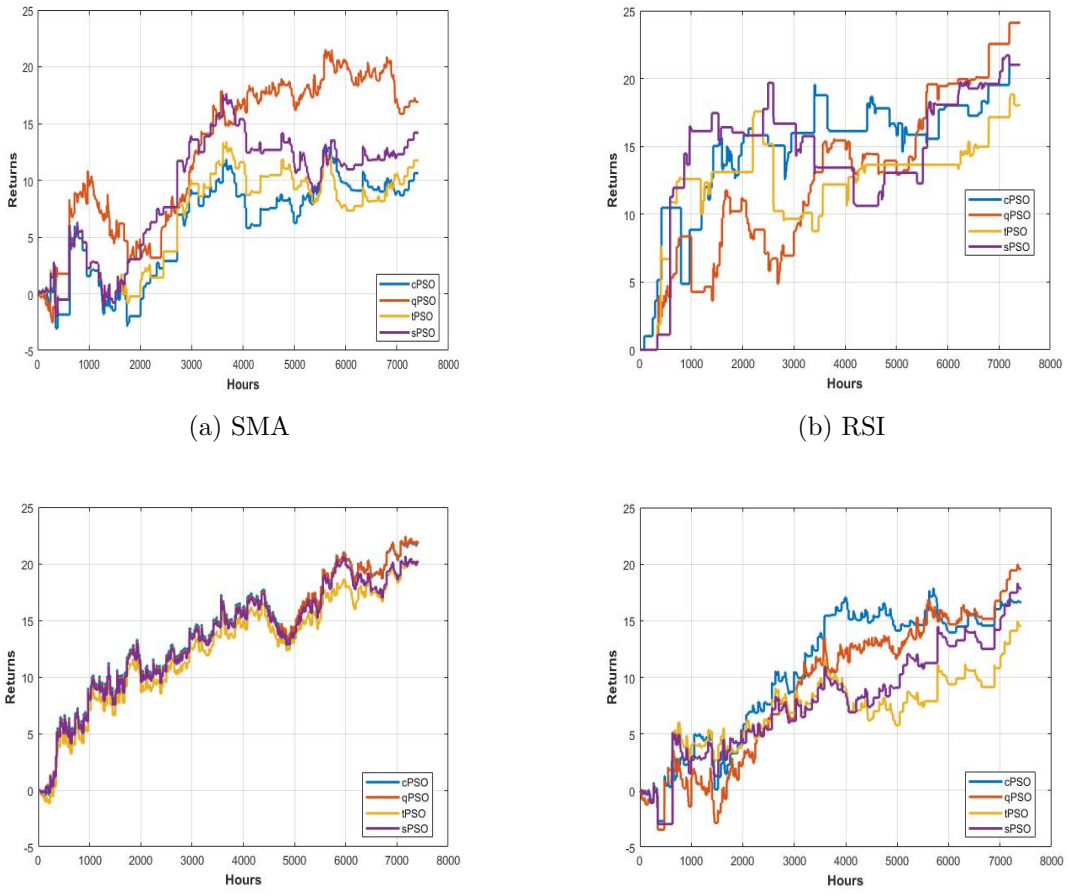
Table 4.3: Algorithms' evaluation without cost on USDJPY testing dataset

TIs	Measures	sPSO	tPSO	cPSO	qPSO
SMA	Avg_{netPF}	-1.022	-1.338	-0.672	**10.038**
	CV	-6.589	-5.442	-11.001	**0.428**
EMA	Avg_{netPF}	2.783	1.951	4.091	**13.133**
	CV	2.317	3.715	1.607	**0.337**
RSI	Avg_{netPF}	-2.203	0.741	1.071	**2.880**
	CV	-4.471	10.507	9.259	**2.794**
MACD	Avg_{netPF}	17.712	17.624	**19.378**	18.924
	CV	0.090	0.080	**0.052**	0.083

4.2.2 USDZAR Without Transaction Cost

Tables 4.4 and 4.5 summarize the results for sPSO, tPSO, qPSO and qPSO for the Avg_{netPF} and CV performance measures for the training and testing data respectively.

From Table 4.4's training results, it can be seen that all algorithms recorded a positive profit with SMA, with qPSO recording the highest profit, and sPSO recording the least profit. However, sPSO was the most stable algorithm with the lowest CV value. The profit trend continued for both EMA and RSI, with qPSO recording the highest profit. qPSO on the other hand recorded the highest profit with MACD. The training results in Table 4.4 confirm the good performance of qPSO, also in comparison to the static



(a) SMA (b) RSI

(c) MACD (d) EMA

Figure 4.1: Accumulated returns for highest returns on USDJPY testing dataset

algorithms. However, the Forex market training results do not necessarily translate into good results during testing.

Table 4.5, which also summarizes the test data results, shows that qPSO performed better than the other algorithms with SMA and RSI, as also depicted in Figures 4.2(a) and 4.2(b). tPSO recorded the best profit with EMA and MACD, which is also depicted in Figures 4.2(c) and 4.2(d). However, all algorithms recorded negative profit with EMA, with qPSO being the worst performer.

In terms of consistency and stability measured by CV, qPSO was more stable with the SMA and MACD TIs, while qPSO was more stable with the EMA and RSI TIs. This

Table 4.4: Algorithms' evaluation without cost on USDZAR training dataset

TIs	Measures	sPSO	tPSO	cPSO	qPSO
SMA	Avg_{netPF}	6.169	6.203	6.353	**6.490**
	CV	**0.060**	0.069	0.067	0.065
EMA	Avg_{netPF}	9.229	9.192	9.337	**9.572**
	CV	0.114	0.057	0.101	**0.009**
RSI	Avg_{netPF}	9.166	9.088	9.032	**9.829**
	CV	0.134	0.130	**0.092**	0.098
MACD	Avg_{netPF}	6.512	6.462	**7.108**	6.308
	CV	0.117	0.124	**0.007**	0.132

Table 4.5: Algorithms' evaluation without cost on USDZAR testing dataset

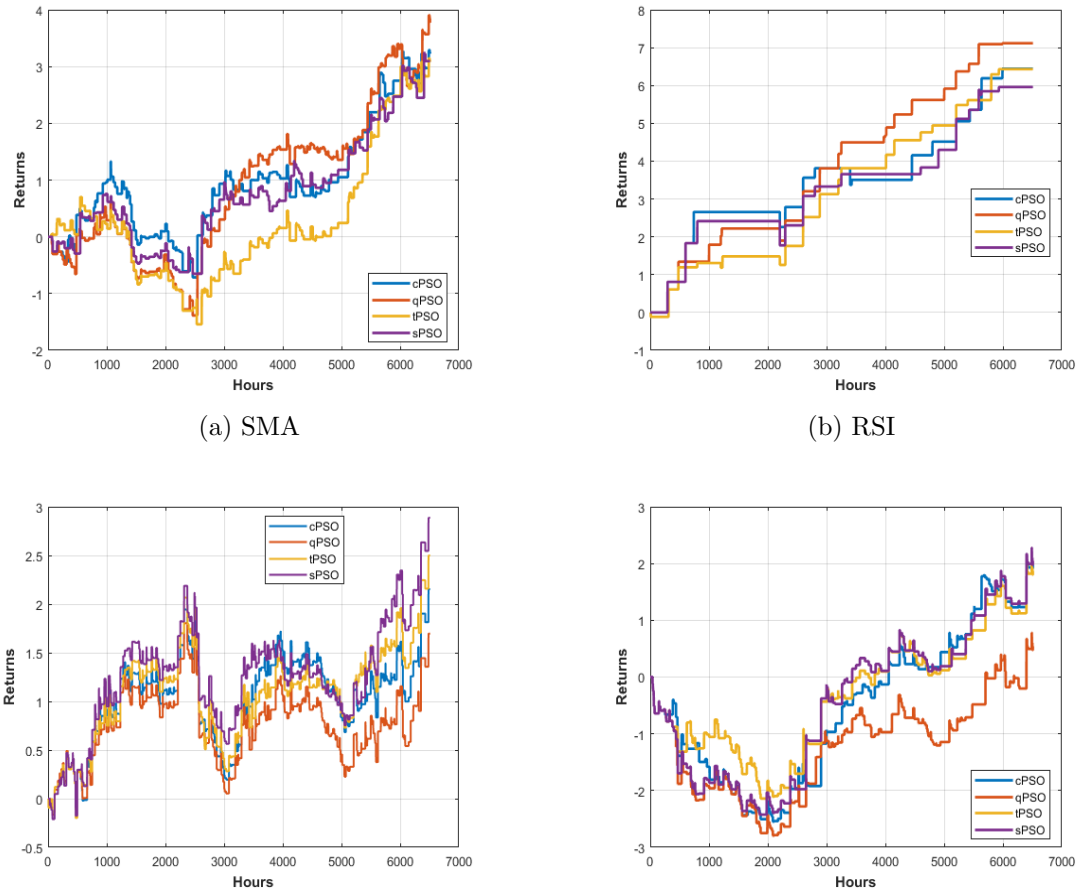
TIs	Measures	sPSO	tPSO	cPSO	qPSO
SMA	Avg_{netPF}	1.309	0.991	1.469	**1.834**
	CV	0.765	1.261	**0.737**	0.807
EMA	Avg_{netPF}	-0.046	**-0.018**	-0.110	-0.628
	CV	-20.280	-46.700	-7.422	**-1.004**
RSI	Avg_{netPF}	2.816	2.975	2.062	**4.277**
	CV	1.809	**1.864**	1.776	1.445
MACD	Avg_{netPF}	1.809	**1.864**	1.776	1.445
	CV	0.307	0.300	**0.077**	0.233

trend also shows that the dynamic PSO algorithms were more stable in performance compared to traditional PSO algorithms.

The observed trends changed completely when algorithms had to maximize profit while also considering the transaction cost, discussed next.

4.2.3 USDJPY With Transaction Cost

This section analyses the performance of all algorithms when introducing a transaction cost of 2.5 percent per return (profit or loss) for both USDZAR and USDJPY.



(a) SMA (b) RSI

(c) MACD (d) EMA

Figure 4.2: Accumulated returns for highest returns on USDZAR testing dataset

Tables 4.6 and 4.7 summarize the results of all algorithms on USDJPY and USDZAR test data. The performance of all algorithms was measured according to the Avg_{netPF} and t_{Cost}, while NO_{trades} was used to assist with analyzing t_{Cost}.

From Table 4.6 it can be observed that no algorithmed perform well when compared to their performance depicted in Table 4.3, which did not include the transaction cost. All algorithms recorded negative profit with SMA. However, qPSO still outperformed the other algorithms by making more trades and paying more for transactions. qPSO continued with its good performance with EMA, by trading more and paying more for transactions. sPSO was able to make a profit with the least transaction cost and trades.

Table 4.6: Algorithms' evaluation with cost on USDJPY testing dataset

TIs	Measures	sPSO	tPSO	cPSO	qPSO
SMA	Avg_{netPF}	-8.877	-9.684	-8.733	**-1.396**
	NO_{trades}	162	169	180	581
	t_{Cost}	**2.214**	2.273	2.343	3.785
EMA	Avg_{netPF}	0.254	-0.014	-0.127	**1.623**
	NO_{trades}	217	223	230	515
	t_{Cost}	**2.401**	2.427	2.446	3.333
RSI	Avg_{netPF}	**2.35**	-1.064	-0.536	-4.419
	NO_{trades}	42	44	60	103
	t_{Cost}	1.136	1.143	**1.323**	1.692
MACD	Avg_{netPF}	14.643	14.069	15.289	**15.320**
	NO_{trades}	629	630	631	624
	t_{Cost}	4.780	4.788	4.797	**4.765**

A different trend was observed when algorithms used RSI. With less transactions and cost, sPSO was able to outperform the other algorithms, which recorded negative profit. These results are completely different from what is depicted in Table 4.3 with RSI. Although qPSO traded more, more trades did not lead to more profit. An algorithm has to make profit trades with good profit margins to make a profit. With MACD, all algorithms made good profit, with qPSO making the most profit, followed by cPSO.

4.2.4 USDZAR With Transaction Cost

The performance of all algorithms on USDZAR with transaction cost did not differ much from USDZAR without cost, except that profits reduced due to the deductions of transaction cost.

From Table 4.7 it can be seen that qPSO still maintained its dominant performance over the other algorithms with SMA, while sPSO performed better than tPSO and qPSO by trading less and making more profit. Again, qPSO outperformed other algorithms with RSI and MACD, followed by tPSO. However, tPSO and qPSO recorded less trans-

Table 4.7: Algorithms' evaluation with cost on USDZAR testing dataset

TIs	Measures	sPSO	tPSO	cPSO	qPSO
SMA	Avg_{netPF}	0.619	0.523	0.540	**1.204**
	NO_{trades}	145	152	176	218
	t_{Cost}	**0.457**	0.465	0.498	0.542
EMA	Avg_{netPF}	-0.824	-0.996	**-0.759**	-1.016
	NO_{trades}	158	160	153	167
	t_{Cost}	0.426	0.428	**0.419**	0.440
RSI	Avg_{netPF}	2.557	2.738	1.729	**4.339**
	NO_{trades}	16	16	14	19
	t_{Cost}	0.134	0.145	**0.131**	0.889
MACD	Avg_{netPF}	1.141	1.176	1.169	**0.178**
	NO_{trades}	316	316	312	317
	t_{Cost}	0.717	**0.715**	0.716	0.722

action cost with RSI and MACD.

All algorithms recorded negative profit with EMA. This followed the same trend as the results obtained without transaction cost (refer to Table 4.5), with qPSO recording the worst profit.

4.2.5 General Observations

- The performance of qPSO was worsened when transaction cost was introduced, while sPSO's performance was improved.
- qPSO traded more than all the other algorithms, while sPSO made less trades.
- All algorithms did not make use of EMA to record profit over the USDZAR currency pair.
- More trades did not necessarily translate into profit. However, more profit trades with a good profit margin could translate into profit.
- All algorithms made good profit with MACD, irrespective of the currency pair.

- The better performance of qPSO and cPSO can be attributed to the algorithms' characteristics. Both qPSO and cPSO dynamically maintain the balance between exploration and exploitation throughout the search process. The randomized repositioning of qPSO charged particles improve the chance to land on a good solution and to influence the normal particles.

- tPSO, on the other hand, could only maintain diversity during the early stage of the search when particles did not yet converge. When particles converged at a later stage or got stuck in local optima, it was difficult to explore for new solutions. In a dynamic environment, like the Forex market where both the search space and the objective function keep changing, temporal exploration algorithms, like tPSO, do not yield good results throughout the search.

- Although qPSO performed well, its CV values were still high. This is due to optima that are far away from the radius making it difficult for qPSO to track. This can be addressed by incorperating a technique to dynamically adapt the radius of qPSO.

4.3 Summary

This chapter explored the performance of different variants of the dynamic particle swarm optimization (PSO) algorithm, namely the quantum PSO (qPSO) and charge PSO (cPSO), on optimizing technical indicators (TIs) in the foreign exchange (Forex) market to maximize profit. The results obtained compared from both the standard particle swarm optimization (sPSO) algorithm and a time-series particle swarm optimization (tPSO) algorithm on the USDJPY and USDZAR currency pairs were compared against one another.

All algorithms showed good performance on profit returns for all TIs during training. qPSO was the best performing algorithm on both the training and testing datasets, and was followed by qPSO. However, the performance of all algorithms deteriorated when transaction cost was introduced. It was difficult for the algorithms to find a trade-off between profit and transaction cost.

Dynamic variants of PSO were able to find optimal parameters for TIs in the Forex

market. Finer tuning of the parameters of the algorithms will result in improved performance. Furthermore, the profit returns of TIs will also depend on the type of currency pairs and trading rules.

Further areas for research include defining the problem as a multi-objective optimization problem (MOP), because the financial market do not only aim to maximize profit, but also to minimize risk and the cost of transaction. Furthermore, adapting a co-operative PSO for TI parameter optimizing might yield good results. These areas of research are investigated in the next two chapters.

Chapter 5

Effects of Decision Models on Dynamic Multi-Objective Optimization Algorithms for Financial markets

Maximizing profit in a financial time series, like foreign exchange, with computational intelligence techniques is very challenging. It is even more challenging to make a decision for a MOP, like automated Forex trading. This chapter investigates the effects of five decision models on three state-of-the-art the dynamic multi-objective algorithms (MOAs) namely, the dynamic vector-evaluated particle swarm optimization (DVEPSO) algorithm, the multi-objective particle swarm optimization algorithm with crowded distance (MOPSO-CD) and dynamic non-dominated sorting genetic algorithm II (DNSGA-II).

5.1 Experimental Setup

The data used for this study is the USDZAR currency pair discussed in Section 3.7.2. Moreover a sliding window mechanism was used as depicted in Section 3.7.3. The next sections discuss the particle representation, objective functions, decision models (DMs)

for multi-objective optimization (MOO) and the parameter configuration.

5.1.1 Particle representation

The trading system used PSO to select good parameters for each TI. Particles were encoded as depicted in Table 3.2, with 10 dimensions, where each dimension represented a parameter of all 4 TIs to be optimized.

5.1.2 Objective Functions

Two conflicting objective functions were employed: the net profit, $netPF$ (refer to Equation 3.18) over a trading period, and T_{Cost} (refer to Equation 3.19). An investor or trader was charged a percentage per returns. The net profit and transaction cost objective functions determine which solution should be part of the POS after training as depicted Figure 3.1(b).

Note: T_{Cost} is negated to turn it into a maximization objective function.

5.1.3 Decision making models for MOO

The decision models used for this experiment were the Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS), simple additive weighting (SAW), gray relational analysis (GRA), objective sum (SUM) and highest profit (HPF), as discussed in Section 2.11.

5.1.4 Parameter Configuration

The parameters and configuration for the algorithms used in this study are as follows:

- The population size was set to 100 for all algorithms and the maximum number of iterations was set to 100. The star topology was used for MOPSO-CD and DVEPSO throughout the study. The values of the inertia weight ω, c_2, and c_1 used for MOPSO-CD and DVEPSO, listed in Table 5.1, guarantee convergence [24].
- DVEPSO used the random knowledge sharing topology and a sub-swarm size of 50.

Table 5.1: Parameter settings of DVEPSO and MOPSO-CD and the boundaries of technical indicators

Parameters	**Values**	**TIs**	**Domain,D**
ω	0.729844	$n1, n2$	[30,200]
c_1	1.496180	lim_s, n	[51,100],[1,200]
c_2	1.496180	$n3$, lim_b	[1,9], [1,50]

- The probability of crossover and mutation for DNSGA-II were set to 0.9 and 0.1667 respectively, which showed good returns [39].

5.1.5 Velocity and Boundary Handling for PSO Algorithms

The velocity and position of particles were clamped to prevent them from overshooting optima and moving out of the search space boundary, as discussed in Section 3.8.2.

5.1.6 Evaluation measures

Three evaluation measures were used for this study, and each one was aggregated over all windows (in sliding window) within a run. The averages of all evaluation measures, namely the net profit, Avg_{netPF}, and winning trades, $gTrade$, over all 30 independent runs were calculated and the coefficient of variation, CV [2] of net profit was also calculated over 30 runs. The evaluation measures were calculated as follows:

$$Avg_{netPF} = \frac{\sum_{i=1}^{R}(\frac{netPF}{I}) * 100}{R} \tag{5.1}$$

where I is the initial investment. In this experiment I is the first closing price from the training and test data. R is the total number of runs, namely 30.

$$gTrade = \frac{\sum_{i=1}^{R}(\frac{Tw}{Tn}) * 100}{R} \tag{5.2}$$

where Tw, Tn are the total number of wins and total number of trades respectively.

5.2 Experimental Results

This section discusses the results obtained from the study. Figure 3.3 depicts the actual price movement for both the training and testing datasets.

Tables 5.2 to 5.4 summarize the outcomes of the study on testing data for DVEPSO, MOPSO-CD and DNSGA-II respectively. The tables show the average net profit, Avg_{netPF}, percentage of profit trades, $gTrade$, and coefficient of variation (CV) values for the relative strength index (RSI) and moving average convergence divergence (MACD) TIs for all DMs on each respective algorithm. Figures 5.1 to 5.3 show regions selected by DMs from the POF on training data.

5.2.1 Results of DVEPSO

From Table 5.2, it is clear that GRA consistently made higher profit than all other DMs on all TIs, except with EMA of which GRA recorded negative profit. HPF recorded the highest profit with EMA from the Pareto-optimal set (POS) of $DVEPSO$, followed by SAW across all measures. $TOPSIS$ on the other hand, performed poorly on all measures.

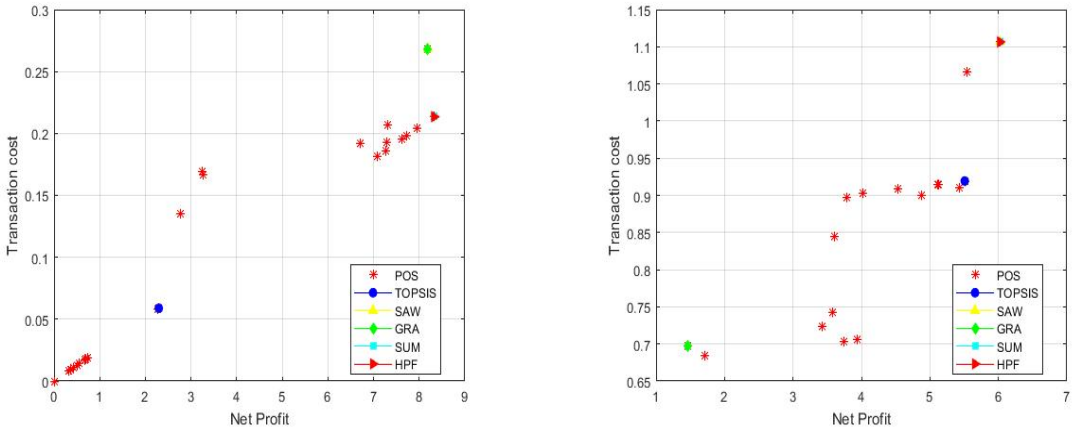
From Figure 5.1, HPF, SUM and SAW selected the same solution from RSI's POS with $DVEPSO$ most of the time, which also can be seen in Table 5.2, i.e. how similar their net profits were. Figure 5.1 shows DMs selecting solutions from different regions of the POS, but HPF, SUM and SAW still followed the same trend. GRA was able to select profitable solutions irrespective of the TI.

5.2.2 Results of MOPSO-CD

Table 5.3 shows that SUM made the best decision for RSI from the $MOPSO-CD$ POS, which resulted in good profit, more profit trades, and a lower CV respectively, followed by HPF. Optimizing MACD TI by MOPSO-CD produced a different trend. GRA outperformed other DMs with higher profit, but with lesser profit trades. SUM and $TOPSIS$ on the other hand, had the highest profit trades and the lowest CV respectively. Moreover, SAW performed well with SMA and EMA TIs by recording the highest and a positive profit.

Table 5.2: Algorithms Evaluation DVEPSO on USDZAR Test dataset

TIs	Measures	TOPSIS	SAW	GRA	SUM	HPF
SMA	Avg_{netPF}	-8.33	-3.26	**2.14**	-7.06	-7.56
	$gTrade$	**47.74**	40.75	40.84	42.66	40.55
	CV	-0.67	-3.590	**5.51**	-0.99	-1.02
EMA	Avg_{netPF}	-0.44	2.72	-2.32	2.84	**3.27**
	$gTrade$	**33.46**	32.22	32.18	32.75	32.52
	CV	-17.658	2.671	-5.435	2.982	**2.425**
RSI	Avg_{netPF}	0.84	2.94	**9.00**	2.29	2.85
	$gTrade$	35.41	61.05	**64.32**	58.54	59.27
	CV	1.166	0.560	**0.213**	0.545	0.473
MACD	Avg_{netPF}	6.35	8.78	**13.37**	7.34	7.40
	$gTrade$	31.38	30.70	**32.83**	30.93	30.91
	CV	0.108	0.085	**0.073**	0.092	0.092



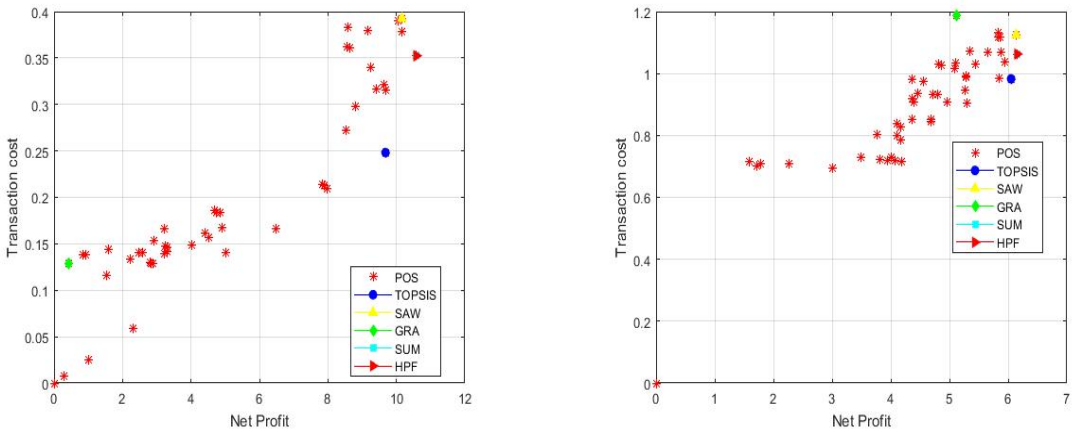
(a) RSI (b) MACD

Figure 5.1: DM points from DVEPSO POF on training dataset

What can be seen from optimizing MACD is that higher profit trades, though important, did not guarantee higher profits. Higher profit trades with high profit margins could guarantee higher profit. In general, MOPSO-CD better optimized MACD in comparison to all other TIs.

Table 5.3: Algorithms Evaluation MOPSO-CD on USDZAR Test dataset

TIs	Measures	TOPSIS	SAW	GRA	SUM	HPF
SMA	Avg_{netPF}	-11.87	**3.45**	-1.17	-7.90	-6.37
	$gTrade$	**51.05**	42.17	42.20	42.70	42.21
	CV	-0.625	**1.934**	-7.959	-1.190	-1.698
EMA	Avg_{netPF}	-6.96	**5.18**	-14.85	-2.05	-1.71
	$gTrade$	30.16	31.02	27.54	31.07	**31.29**
	CV	-1.2108	**0.922**	-0.463	-3.200	-4.148
RSI	Avg_{netPF}	2.03	3.95	8.29	**8.88**	8.83
	$gTrade$	49.50	61.25	63.85	**69.30**	68.05
	CV	0.218	0.318	0.163	**0.103**	0.106
MACD	Avg_{netPF}	10.16	10.63	**15.98**	11.52	11.74
	$gTrade$	31.42	28.67	28.52	**30.71**	30.70
	CV	**0.044**	0.063	0.056	0.045	0.047



(a) RSI (b) MACD

Figure 5.2: DM points from MOPSO-CD POF on training dataset

Figure 5.2 illustrates that the decision trend was different from the decision made from DVEPSO's POS. SUM and HPF followed almost the same trend in both the POS of $MOPSO-CD$ RSI and $MACD$.

Table 5.4: Algorithms Evaluation DNSGA-II on USDZAR Test dataset

TIs	Measures	TOPSIS	SAW	GRA	SUM	HPF
SMA	Avg_{netPF}	-12.84	-5.62	-2.47	-9.78	**5.62**
	$gTrade$	**52.465**	41.732	40.671	41.938	41.732
	CV	**-0.366**	-1.538	-3.752	-0.594	-1.538
EMA	Avg_{netPF}	-3.20	-5.29	**2.21**	-5.58	-5.29
	$gTrade$	31.114	28.927	**38.241**	28.853	28.927
	CV	-1.507	-0.797	2.416	**0.782**	-0.797
RSI	Avg_{netPF}	0.00	10.86	7.11	**11.28**	10.86
	$gTrade$	0.00	71.15	66.05	**74.36**	71.15
	CV	0	0.106	0.177	**0.094**	0.106
MACD	Avg_{netPF}	11.76	8.19	**21.00**	9.58	8.19
	$gTrade$	31.50	29.93	**36.62**	31.52	29.93
	CV	**0.037**	0.063	**0.037**	0.054	0.063

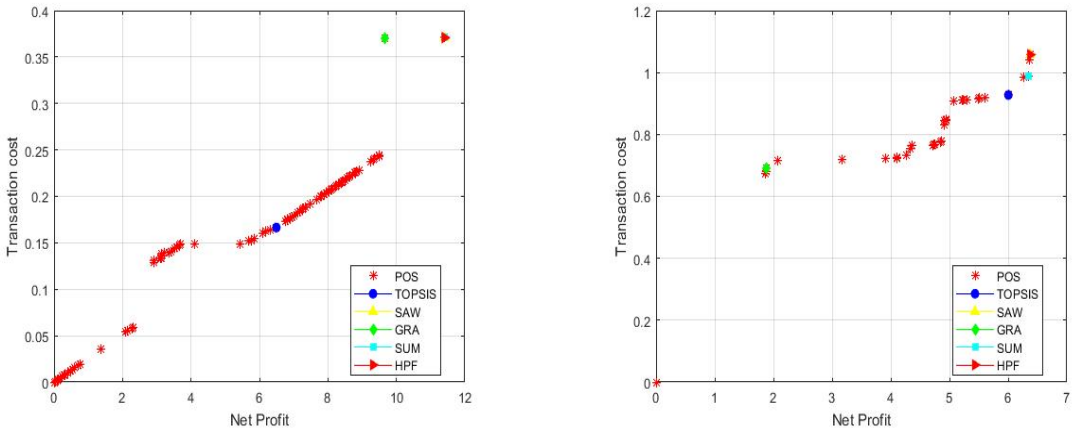
5.2.3 Results of DNSGA-II

From Table 5.4 it can be seen that, $TOPSIS$ did not make any trade with RSI from the DNSGA-II POS. However, SUM outperformed all DMs for RSI across all measures, followed by SAW and PF, while GRA outperformed other DMs for $MACD$ and EMA and was followed by $TOPSIS$. However, HPF did perform better than other DMs with SMA.

Figure 5.3 and Table 5.4 depict that HPF and SAW selected the same solution and produced the same net profit for both RSI and $MACD$.

5.2.4 General Observations

- GRA in general performed better than other DMs across all algorithms with MACD.
- SUM also was a better performer across all algorithms with RSI.
- In terms of which algorithm had the most consistence performance, measured by



(a) RSI (b) MACD

Figure 5.3: DM points from DNSGA-II POS for MACD on training dataset

CV, $DNSGA-II$ was the best algorithm with $DVEPSO$ being the worst.

- All algorithms were able to optimize $MACD$. Except for RSI, of which algorithms struggled to produce profit.
- Moreover, Table 5.5 shows that more trades were recorded with $MACD$, irrespective of the DMs.
- From Figures 5.1 to 5.3, it can be seen that the higher the cost of the transaction in RSI, the higher the returns in test data. However, lower net profit solutions from the training data set generated higher profit in the testing data for $MACD$.
- Almost all DMs struggled to produce positive returns with SMA and EMA for all algorithms' POS.

Table 5.5: Number of profit trades for all

TIs	Measures	TOPSIS	SAW	GRA	SUM	PF
RSI	$DVEPSO$	2	18	15	3	5
	$MOPSO-CD$	2	20	17	7	7
	$DNSGA-II$	0	9	3	8	9
MACD	$DVEPSO$	35	41	37	37	38
	$MOPSO-CD$	41	51	42	44	45
	$DNSGA-II$	40	46	31	44	45

5.3 Summary

The focus of this chapter was to explore the effect of decision models (DMs) on the net profit of automated foreign exchange (Forex) trading with dynamic multi-objective optimization algorithms (DMOAs).

Three state-of-the-art dynamic multi-objective algorithms (DMOAs) were used, namely the dynamic vector-evaluated particle swarm optimization (DVEPSO) algorithm, the multi-objective particle swarm optimization algorithm with crowded distance (MOPSO-CD) and the dynamic non-dominated sorting genetic algorithm II (DNSGA-II). Additionally, five DMs were employed.

The results showed that each region of the Pareto front (POF) generated a different net profit out of the sample data set. However, gray relational analysis (GRA) and objective sum (SUM) were able to consistently find good points across all DMOAs and technical indicators (TIs) used.

Since each region of the multi-objective algorithm (MOA)'s POF generated different profit, it cannot be categorically stated that one DMOA performed better than the other in the Forex market. However, DNSGA-II was the most stable algorithm with the lowest coefficient of variation (CV) values.

More work needs to be done in the area of automated decision making for dynamic multi-objective optimization (DMOO), especially for the Financial market. Moreover, hybrid or combined decision models for DMOAs and the Financial market should be explored.

Chapter 6

Dynamic Multi-Swarm Multi-Objective PSO

This chapter discusses a new algorithm proposed in this dissertation, namely the dynamic multi-swarm multi-objective particle swarm optimization (DMS-MOPSO) algorithm.

6.1 Dynamic multi-swarm multi-objective particle swarm optimization

This section discusses the details of DMS-MOPSO. The DMS-MOPSO algorithm is presented in Algorithm 7. DMS-MOPSO is a dynamic multi-swarm multi-objective particle swarm optimization (MOPSO) algorithm inspired by the work of Blackwell and Branke [9]. Blackwell and Branke [9] proposed a multi-swarm particle swarm optimization (PSO) algorithm for dynamic environments by incorporating exclusion, anti-convergence, quantum and charged particles. Their approach is adapted for multi-objective optimization problems (MOPs). DMS-MOPSO makes use of only the exclusion technique without anti-convergence. The components of DMS-MOPSO are discussed in the following sections.

6.1.1 Multi swarms

The whole population of particles are divided into n multiple sub-swarms, S_n. The purpose of the multi swarm algorithm is to try and position groups (sub-swarms) on different promising peaks (optima). Moreover, MOPs are multi-modal in nature and have more than one optima. Therefore, the aim of a multi-objective optimizer is to find as many optima (set of solutions or POS) as possible [27]. To make the multi-swarms algorithm very effective and not just a division of swarms, there should be a mechanism so that sub-swarms can share knowledge or information. Two forms of communication approaches are employed, namely exclusion [9] and a knowledge transfer topology [33].

Exclusion

Since the aim of a multi-swarm is to position sub-swarms on different peaks in the search space, exclusion prevents more than one sub-swarm from settling on a peak, which is a very common problem in multi-swarm algorithms [9]. In order to prevent this and encourage swarm-diversity, sub-swarms can repel each other. However, this approach might also prevent sub-swarms from reaching the peak. In order to address the aforementioned problems in multi-objective optimization (MOO), the exclusion approach in [9] is modified as follows:

- A crowded-distance comparison is made between the selected gBest (leader) of all sub-swarms. One sub-swarm is randomly marked for exclusion if the crowding distance between the $gBest$ of sub-swarm, S_a, and the $gBest$ of the next sub-swarm, S_{a+1}, is the same or falls within a predefined radius [9]. Since the Forex market is a discrete problem, a radius is not used. Parameters of TIs can have different values, but the same objective values.

- Any density measure can also be used to measure the closeness of the gBest of the sub-swarms.

The reaction of DMS-MOPSO after a sub-swarm has been marked for exclusion is outlined in Section 6.1.2.

Selecting a Leader

Selecting a global guide for MOPSO is not as simple as for a single-objective optimization (SOO) PSO where only the best solution is used. A set of solutions, namely the Pareto-optimal set (POS), are all good solutions. For this reason there should be a mechanism to select a gBest. Studies have proposed different approaches to select a gBest for MOPSO. The approach adapted for DMS-MOPSO is as defined in [49]. Each sub-swarm maintains its own local archive, lA_n, with non-dominated solutions, which is then sorted by crowded distance. A gBest is selected from the top 10 percent of the solutions in lA_n for each sub-swarm.

A leader, ld, which will serve as the global guide for each sub-swarm, is selected according to a knowledge sharing topology as discussed in [33] and depicted in Figure 6.1. The two knowledge sharing topologies are:

- Ring topology: this is the traditional topology used in the vector-evaluated particle swarm optimization (VEPSO) algorithm [53]. As depicted in Figure 6.1, the leader for a sub-swarm, S_a, is the $gBest$ of the next sub-swarm, S_{a+1}, which aids in information sharing among sub-swarms.

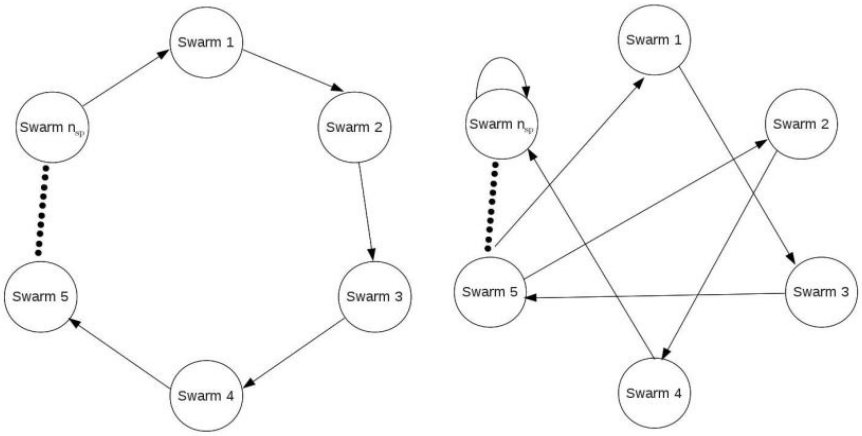
- Random topology: with this approach, on the other hand, which was proposed by Helbig and Engelbrecht [32], a sub-swarm's gBest or the gBest of any sub-swarm can be selected as the leader.

.

6.1.2 DMS-MOPSO procedure

Algorithm 7 presents the stages in DMS-MOPSO. The algorithm initializes the population, pop, and each sub-swarm, S_a. During initialization, the $pBest$ is set to the particle's current position. $gBest$ is updated according to Section 6.1.2.

After initialization, the algorithm goes through the following stages:



(a) Ring Topology (b) Random Topology

Figure 6.1: Knowledge transfer topologies [33]

Check for exclusion

The algorithm checks whether more than one sub-swarm have settled on a solution or have converged to the same solution. This is done by following the procedure in section 6.1.1.

Check for change

Each sub-swarm checks for a change in the environment. A random number of particles (called sentry particles) are selected and re-evaluated before every iteration. If the current objective vector dominates the previously stored objective vector or vice visa, the sub-swarm is marked for change ($change =$ "$True$").

React to change and exclusion

If a sub-swarm is marked "$True$" for both change and exclusion:

- "b" percent of the sub swarm or the whole subs swarm is re-initialized and re-evaluated. "b" is a decreasing linear equation per iteration and is defined as follows:

$$b = \left(\frac{1 - (cI - 1)}{(mI - 1)} \right)^{(1/80)} \tag{6.1}$$

where cI and mI denote the current iteration and maximum number of iterations respectively.

- Moreover, all non-dominated solutions are added to the local archive, lA (only associated with the sub-swarm). The local archive, lA, is sorted by crowded distance.
- The $gBest$ for the sub-swarm is randomly selected from the n percent bottom of the local archive, lA.
- Change and exclusion are then set to "$False$".

However, if a sub-swarm is marked for only change detection:

- The $gBest$ remains unchanged, but "b" percent of the sub swarm or the whole subs swarm is re-initialized and re-evaluated.
- Change are set to "$False$".

Moreover, if a sub swarm is marked "$True$" for only exclusion:

- All non-dominated solutions are added to the local archive, lA.
- The local archive, lA, is sorted by crowded distance.
- The $gBest$ for the sub swarm is randomly selected from the n percent bottom of the local archive, lA and exclusion is set to "$False$".

The $pBest$ values of re-initialized particles are reset to the current position to prevent biased movement towards the particles' previous $pBest$ [33].

Archive Management

Non-dominated solutions from the population, pop, are added to the external archive gA, and all solutions in the archive dominated by new solutions are removed from the archive. If gA exceeds the limit, gA is truncated by sorting the solutions by crowded distance in descending order to removing the bottom solutions (solutions with the least crowded distance).

pBest Update

The $pBest$ is set to the current position if the current position dominates the $pBest$. The current $pBest$ or current position is randomly selected as the $pBest$ if none dominates the other. Moreover, the $pBest$ of reinitialized particles after change is set to the current position to avoid bias towards old optima.

gBest Update

The $gBest$ of the current sub-swarm is set to the $pBest$ if the $pBest$ dominates the $gBest$.

Algorithm 7 Pseudo code for DMS-MOPSO

DMS-MOPSO$(parameters, problemdefinition)$
$pop \leftarrow Initialization$
For $EACHsubSwarm, S_a$
  $exclusion \leftarrow$ "$False$"
EndFor
$t \leftarrow 0$
While$(t \leq MAXevaluation)$
  Check for Exclusion
  For$(EACHsubSswarm, S_a)$
    Check for change
    Respond to change and exclusion
    $exclusion, change \leftarrow$ "$False$"
    For$(EACHparticle, i, ofsubSwarm, a\ )$
      Select leader
      Update velocity with equation 2.8
      Apply velocity limit
      Update position with equation 3.17 for discrete values
      Apply position limit
      Evaluate particle
      Update pBest
      Update gBest
    EndFor
  EndFor
$A \leftarrow$ *Assign non-dominated solution (pop)*
if $A > A_{limit}$ **then**
  $Truncate$
  (Remove particles with
  the least crowded distance)
end if
EndWhile
return A *as* POF , POS
EndProcedure

6.2 Experimental Setup

This section discusses the experimental setup used for experiments conducted in this Chapter. Section 6.2.1 discusses the evaluation measures used to evaluate the performance of the algorithms. Sections 6.2.2 and 6.2.3 present the DMs and parameter configurations used, respectively.

6.2.1 Evaluation Measures

Three evaluation measures were used for this study, and each one was aggregated over all windows (in sliding window) within a run. The averages of all evaluation measures, namely the net profit, Avg_{netPF}, and winning trades, $gTrade$, over all 30 independent runs were calculated. The coefficient of variation, CV [2] of net profit was also calculated over 30 runs. The evaluation measures were calculated as discussed in Section 5.1.6.

6.2.2 Decision making models in MOO

The decision models used for these experiment were Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS), simple additive weighting (SAW), GRA, SUM and highest profit (HPF) as discussed in Section 2.11.

6.2.3 Parameter Configuration

The parameters and configuration of the algorithms used in this study are:

- The sub-swarm size was set to 50 for all algorithms and the maximum number of iterations was set to 100. The star topology was used the study. The values listed in Table 6.1 for ω, C_1 and C_2 guarantees convergence [24]. The buy, lim_a, and sell, lim_c, limits for relative strength index (RSI) are listed in Table 6.1.
- DMS_{r2}: the number of sub-swarms was set to two, and 30 percentage of the sub-swarm was re-initialized during change and exclusion reaction was used.
- DMS_{r4}: the number of sub-swarms was set to four, and 30 percentage of the sub-swarm was re-initialized during a change and exclusion reaction.

Table 6.1: Parameter settings of DVEPSO and MOPSO-CD and the boundaries of technical indicators

Parameters	**Values**	**TIs**	**Domain,D**
ω	0.729844	$n1, n2$	[30,200]
c_1	1.496180	lim_a, n	[51,100],[1,200]
c_2	1.496180	$n3$, lim_c	[1,9], [1,50]
Q, r_{cloud}	$0.15 * (D_{max})$		
δ, R_c, R_p	0.1, 1, $\sqrt{3D_{max}}$		

- DMS_{q2}: the number of sub-swarms was set to 2, and 30 percentage of the sub-swarm was re-initialized during a change and exclusion reaction. 50 percent of the swarm was quantum particles.
- DMS_{q4}: the number of sub-swarms was set to 4, and 30 percentage of the sub-swarm was re-initialized during a change and exclusion reaction. 50 percent of the swarm was quantum particles.
- A random knowledge sharing topology was used for all algorithms.

Note: Velocity and boundary constraints were handled as explained in Section 3.8.2.

6.3 Results of the performance of different variants of DMS-MOPSO on EURGBP dataset

This section discusses the results obtain from comparing the different varaints of DMS-MOPSO. Section 6.3.1 discusses the results obtained with $TOPSIS$ DM for all algorithms over $EURGBP$ currency pair. Section 6.3.2 discusses the results obtained with SAW DM for all algorithms over $EURGBP$ currency pair.

Table 6.2: Algorithms Evaluation with TOPSIS on EURGBP Test dataset

TIs	Measures	DMS_{r2}	DMS_{r4}	DMS_{q2}	DMS_{q4}
SMA	Avg_{netPF}	**1.32**	-0.08	1.17	0.39
	$gTrade$	33.45	32.01	**33.69**	32.63
	CV	**1.41**	-28.90	1.92	5.60
EMA	Avg_{netPF}	-1.85	-2.38	**-1.56**	-1.62
	$gTrade$	25.52	25.26	**26.20**	25.61
	CV	-1.29	**-0.60**	-1.25	-1.30
RSI	Avg_{netPF}	-0.17	-2.14	**0.09**	-1.57
	$gTrade$	44.44	28.06	**52.67**	39.44
	CV	-20.47	-1.65	**50.48**	-2.49
MACD	Avg_{netPF}	-12.75	-13.04	**-12.32**	-12.74
	$gTrade$	29.66	29.64	30.72	**30.94**
	CV	-0.11	**-0.10**	-0.12	-0.13

6.3.1 TOPSIS on EURGBP

From Table 6.2, DMS_{r2} recorded the highest profit and a lower CV value, followed by DMS_{q2} with SMA. However, DMS_{r4} recorded a negative profit. All algorithms recorded negative a profit with EMA DMS_{q2}, however, recorded the least negative profit.

The observed trends were slightly different with RSI, where only DMS_{q2} recorded a positive profit. All algorithms were very unstable as indicated by their respective CV values. All algorithms failed to maximize profit with $MACD$ by using $TOPSIS$ DM. The worse negative profit was recorded with $MACD$.

6.3.2 SAW on EURGBP

From Table 6.3, it can be seen that in general, algorithms making a decision with the SAW DM performed better than when using the $TOPSIS$ DM.

DMS_{q2} was the best performer with a higher profit and a lower CV, followed by DMS_{r2}. DMS_{q4} recorded a negative profit. All algorithms again recorded negative

Table 6.3: Algorithms Evaluation with SAW on EURGBP Test dataset

TIs	Measures	DMS_{r2}	DMS_{r4}	DMS_{q2}	DMS_{q4}
SMA	Avg_{netPF}	2.27	0.38	**2.93**	-0.29
	$gTrade$	42.89	**43.04**	42.84	42.85
	CV	1.96	9.74	**1.01**	-14.81
EMA	Avg_{netPF}	**-7.08**	-10.04	-8.60	-9.41
	$gTrade$	**18.73**	18.21	18.16	17.93
	CV	-0.52	-0.45	**-0.43**	-0.57
RSI	Avg_{netPF}	**6.44**	4.35	5.19	5.92
	$gTrade$	**66.78**	63.39	66.32	65.04
	CV	**0.96**	1.31	1.12	1.03
MACD	Avg_{netPF}	-11.50	**-10.19**	-12.07	-10.58
	$gTrade$	28.42	28.81	**29.22**	29.14
	CV	**-0.14**	-0.20	-0.18	-0.17

profits with EMA. Moreover, the performance of the algorithms was worse with EMA by using the SAW DM rather than the $TOPSIS$ DM.

All algorithms performed much better with RSI, with DMS_{r2} recording the highest profit and $gTrade$, and the lowest CV, followed by DMS_{q4}. Maximizing profit with RSI is where the SAW DM gave algorithms a boost. All algorithms performed poorly well with $MACD$. Both SAW and $TOPSIS$ did not make a good decision from the POS of algorithms with $MACD$. However, SAW performed slightly better than $TOPSIS$.

6.4 Performance of DMS-MOPSO against other algorithms

This section compares the performance of DMS-MOPSO against the performance of some of the state-of-the-art MOAs, namely the DNSGA-II, the DVEPSO and MOPSO-CD. The parameter configuration for the algorithms are discussed in Section 5.1.4. The USDZAR currency pairs dataset was used for both training and testing. DMS_{r2} was

used for the comparison.

6.4.1 Results

This section discusses the results obtained from the experiment. The performance measures used for the study were the average net profit, Avg_{netPF}, profit trades, $gTrade$, and CV for average profit. Tables 6.4 to 6.6 summarise the obtained results.

6.4.2 Performance of algorithms on USDZAR with GRA

From Table 6.4 it can be seen that $DVEPSO$ made a profit, followed by DMS_{r2} with SMA. However, the other algorithms recorded a negative profit. The CV values for all algorithms were very volatile, as depicted in Figure 6.2(a). This means the probability of achieving consistent positive profit is very low. $DNSGA-II$ outperformed the other algorithms in all performance measures with EMA. $DEVPSO$ and $MOPSO-CD$, however, recorded negative profits. All algorithms still recorded higher CV values, making all algorithms very unpredictable in terms of performance with EMA.

Performance with RSI and $MACD$ did follow a different trend. $DVEPSO$ recorded higher profit and was followed by $MOPSO-CD$. Although $DNSGA-II$ recorded the highest profit trades, $gTrade$, it recorded the lowest profit, which means the profit margins were small. All algorithms recorded lower CV values as compared to SMA and EMA, with $MOPSO-CD$ recording the least profit. As depicted in Figure 6.2(c), the algorithms are much stable with RSI. $DNSGA-II$ recorded the highest profit and lowest CV with $MACD$, followed by DMS_{r2}. As depicted in Figure 6.2(d), the algorithms with $MACD$ were the most stable, with the lowest CV.

6.4.3 Performance of algorithms on USDZAR with SUM

Table 6.5 indicates that DMS_{r2} was the only algorithm which recorded a positive profit with SMA. Moreover, $DVEPSO$ was the only algorithm which recorded a positive profit with EMA, while $DNSGA-II$ recorded the lowest CV. Figures 6.3(a) and 6.3(b) depict how unstable and how high the volatility of the algorithms with SMA and EMA were, respectively.

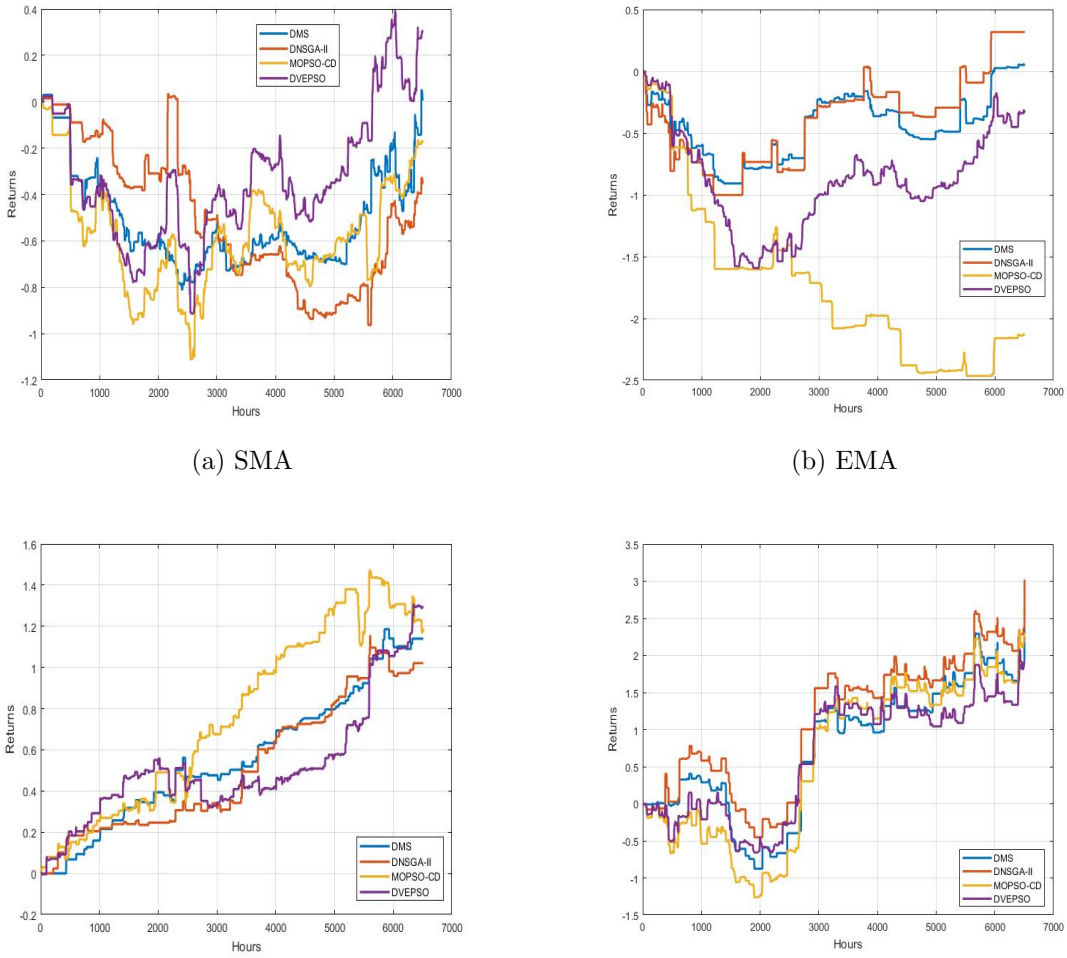
Table 6.4: Algorithms Evaluation with GRA on USDZAR Test dataset

TIs	Measures	$DNSGA-II$	$DVEPSO$	$MOPSO-CD$	DMS_{r2}
SMA	Avg_{netPF}	-2.46	**2.14**	-1.17	0.04
	$gTrade$	40.67	40.84	**42.20**	40.02
	CV	-3.75	**5.51**	-7.96	32.47
EMA	Avg_{netPF}	**2.21**	-2.32	-14.85	0.34
	$gTrade$	**38.24**	32.18	27.54	35.60
	CV	**2.42**	-5.45	-0.46	-33.22
RSI	Avg_{netPF}	7.11	**9.00**	8.29	7.94
	$gTrade$	**66.04**	64.32	63.85	64.55
	CV	1.23	1.49	**1.14**	1.40
MACD	Avg_{netPF}	**20.99**	13.37	15.98	18.49
	$gTrade$	36.62	32.83	28.52	**37.68**
	CV	**0.26**	0.518	0.39	0.34

Table 6.4 also depicted that all algorithms recorded a positive profit with RSI and $MACD$. DMS_{r2} outperformed all algorithms with a higher profit and a lower CV value, and was followed by $DNSGA-II$ with a higher $gTrade$ with RSI. Figure 6.3(c) depicts a smoother performance of all algorithms as compared to Figures 6.3(a) and 6.3(b). $MOPSO-CD$ performed the best with $MACD$ by obtaining a higher profit, followed by DMS_{r2} with the lowest CV value. As depicted in Figure 6.3(d), all algorithms gained momentum after 2000 hours of trade.

6.4.4 Performance of algorithms on USDZAR with HPF

DMS_{r2} outperformed all algorithms with SMA, RSI and $MACD$ with the highest and the lowest profit and CV values respectively. All algorithms recorded negative profits with SMA. $DVEPSO$ was the best performer with EMA, with a higher profit and $gTrade$, and a lower CV. From Figure 6.4, it can be seen that all algorithms took longer to make profit. The exception is visible in Figure 6.4(c), where the algorithms made profit at the early stage of trading.



(a) SMA

(b) EMA

(c) RSI

(d) MACD

Figure 6.2: Accumulated average profit with GRA

6.4.5 General observations

- All algorithms were more stable with lower CV values with $MACD$.
- As depicted in Figures 6.2 to 6.4, it takes longer for an algorithm to adapt to the dynamics of the dataset or environment.
- All algorithms performed better with RSI and $MACD$, across all DMs.
- DMS_{r2} was more stable and performed better with the SUM and HPF DMs, but

Table 6.5: Algorithms Evaluation with SUM on USDZAR Test dataset

TIs	**Measures**	$DNSGA-II$	$DVEPSO$	$MOPSO-CD$	DMS_{r2}
SMA	Avg_{netPF}	-9.78	-7.06	-7.90	**0.93**
	$gTrade$	41.94	42.66	**42.70**	42.21
	CV	-0.60	-0.99	-1.19	**6.41**
EMA	Avg_{netPF}	-5.58	**2.84**	-2.05	-4.73
	$gTrade$	28.85	**32.75**	31.07	29.54
	CV	**0.78**	2.98	-3.20	-1.03
RSI	Avg_{netPF}	11.28	2.29	8.88	**13.27**
	$gTrade$	**74.36**	58.54	69.30	69.74
	CV	0.66	3.805	0.72	**0.55**
MACD	Avg_{netPF}	9.58	7.34	**11.52**	11.26
	$gTrade$	**31.52**	30.93	30.71	31.43
	CV	0.37	0.64	0.31	**0.30**

Table 6.6: Algorithms Evaluation with HPF on USDZAR Test dataset

TIs	**Measures**	$DNSGA-II$	$DVEPSO$	$MOPSO-CD$	DMS_{r2}
SMA	Avg_{netPF}	-5.62	-7.56	-6.37	**2.93**
	$gTrade$	41.73	40.55	**42.21**	41.16
	CV	-1.54	-1.02	-1.70	**2.44**
EMA	Avg_{netPF}	-5.29	**3.27**	-1.71	-7.13
	$gTrade$	28.93	**32.52**	31.29	29.46
	CV	-0.80	**2.43**	-4.15	-0.66
RSI	Avg_{netPF}	10.86	2.85	8.83	**12.64**
	$gTrade$	**71.15**	59.27	68.05	68.37
	CV	0.74	3.30	0.74	**0.71**
MACD	Avg_{netPF}	8.19	7.40	11.74	**12.10**
	$gTrade$	29.93	30.91	30.79	**31.00**
	CV	0.44	0.64	0.33	**0.30**

(a) SMA

(b) EMA

(c) RSI

(d) MACD

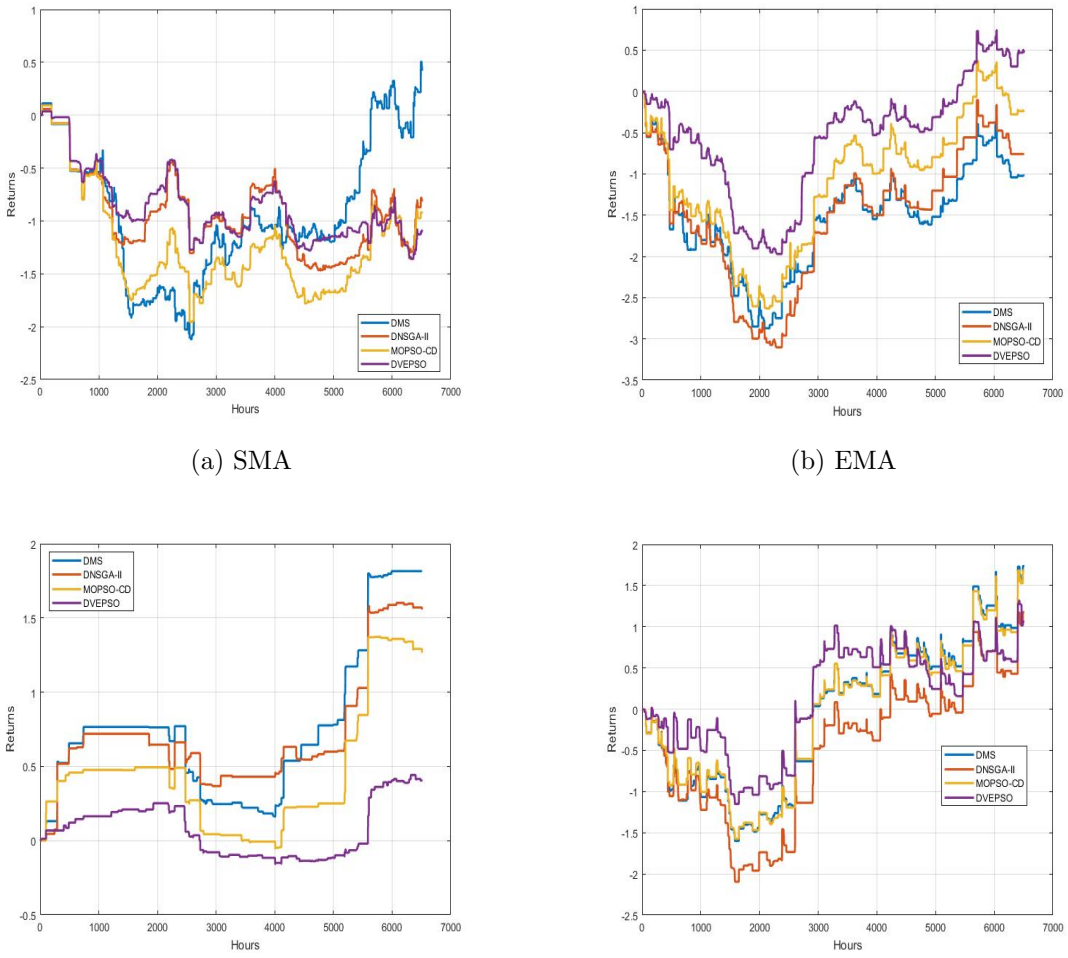
Figure 6.3: Accumulated average profit with SUM

performed worse with *GRA*.

- The performance of each algorithm differed with respect to the DM and the TI.

6.5 Summary

This chapter discussed the proposed algorithm for dynamic multi-objective optimization (MOO), namely the dynamic multi-swarm multi-objective particle swarm optimization



(a) SMA

(b) EMA

(c) RSI

(d) MACD

Figure 6.4: Accumulated average profit with HPF

(DMS-MOPSO). Section 6.1 discussed all the stages and procedures of DMS-MOPSO, including how the archive was managed. An experiment was conducted and the results were discussed in Sections 6.3 and 6.4. Section 6.3 compared the performance of different variants of DMS-MOPSO. The results showed that DMS_{r2} and DMS_{q2} performed better in general. Moreover the performances differed slightly with respect to the decision model (DM) used. However, DMS_{q2} is quantum particles showed a better performance and stability.

Moreover, the results obtained from Section 6.4 shows the performance of DMS_{r2}

against other state-of-the-art multi-objective algorithms (MOAs) when using three DMs. DMS_{q2} performed better than the other algorithms with objective sum (SUM) and highest profit (HPF). On the other hand, $DNSGA-II$ perform well with GRA.

Chapter 7

Conclusions

This chapter summarises the conclusions of the studies conducted in this dissertation. Section 7.1 summarises the various conclusions and observations made from the studies. Section 7.2 discusses possible future work eminating from this study.

7.1 Summary of Conclusions

In this dissertation a new dynamic multi-swarm multi-objective particle swarm optimization (DMS-MOPSO) was proposed. The purpose of a muti-swarm is to enable the algorithm to track different promising solutions (or trends). DMS-MOPSO was used to optimize four technical indicators (TIs). To make the application of metaheuristics to real world problems (foreign exchange (Forex)) more justifiable, two conflicting objectives were optimized: the net profit and transaction cost were maximized and minimized simultaneously. DMS-MOPSO was compared against other nature inspired state-of-the-art dynamic multi-objective algorithms (DMOAs).

The main objective of this dissertation was to develop a particle swarm optimization (PSO) based multi-objective algorithm (MOA) to solve dynamic multi-objective optimization problems (MOPs), namely DMS-MOPSO. Moreover, this study aimed to explore the effects on decision models (DMs) for the Financial market. Four experimental studies were conducted.

Chapter 4 explored the performance of various dynamic PSOs when maximizing profit

in a complex dynamic environment, such as the Forex market. Quantum PSO (qPSO) and charge PSO (cPSO) algorithms were employed and compared against the standard particle swarm optimization (sPSO) algorithm and the time-series particle swarm optimization (tPSO) algorithm, which are usually used when optimizing TIs for the stock and Forex markets [16, 58]. The results showed that no algorithm was able to perform well when cost of transaction was applied. All algorithms recorded negative profit with the simple moving average (SMA). However, qPSO still outperformed the other algorithms by making more trades and paying more for transactions. qPSO continued with its good performance with exponential moving average (EMA), by trading more and paying more for transactions. sPSO was able to make a profit with the least transaction cost and trades.

A different trend was observed when algorithms used the relative strength index (RSI). With less transactions and cost, sPSO was able to perform better than the other algorithms, which recorded negative profit. These results were are completely different from what was depicted in Table 4.3 with RSI. Although qPSO traded more, it did not lead to more profit, because an algorithm has to make profit trades with good profit margins to make a profit. With the moving average convergence divergence (MACD), all algorithms produced good profit, with qPSO making the most profit, followed by qPSO.

Chapter 5 explored the effect of DMs on the net profit of automated Forex trading with dynamic MOAs. Three state-of-the-art DMOAs were used, namely the dynamic vector-evaluated particle swarm optimization (DVEPSO) algorithm, the multi-objective particle swarm optimization algorithm with crowded distance (MOPSO-CD) and the dynamic non-dominated sorting genetic algorithm II (DNSGA-II). Additionally, five DMs were employed. The results showed that each region of the Pareto front (POF) generated a different net profit out of the sample data set. However, gray relational analysis (GRA) and objective sum (SUM) were able to consistently find good points across all DMOAs and TIs used. Since each region of the MOA's POF generated different profit, it cannot be categorically stated that one DMOA performed better than the other in the Forex market. However, DNSGA-II was the most stable algorithm with the lowest coefficient of variation (CV) values.

In Chapter 6, two experiments were conducted and the obtained results analyzed. Different variants of DMS-MOPSO were evaluated. DMS_{r2} (without quantum particles) and DMS_{q2} (with quantum particles) variants of DMS-MOPSO showed to be the better performers. However, DMs really influenced the performance of the algorithms. When DMS_{r2} was compared against four state-of-the-art algorithms, the results showed that DMS_{r2} in general performed better and was more stable as the other algorithms.

The obtained results showed that a multi-swarm approach for multi-objective optimization (MOO) can solve dynamic MOPs.

7.2 Future Work

- More work needs to be done in the area of automated decision making for dynamic multi-objective optimization (DMOO), especially for the Financial market. Moreover, a hybrid or combined decision model for DMOAs and the Financial market should be explored.

- Incremental Learning - one of the most challenging aspects of financial markets is to trade in real time. The financial market is a complex and a dynamic environment. An algorithm must be able to track any change and make good trades within a specific time frame. However, when the dynamics of the dataset change, any previously trained model might perform poorly. In order for an algorithm to keep performing well, it should be re-trained. More research should go into how to quickly re-train a model to make good trades.

- DMS-MOPSO should be explored more by fine tuning parameters, and with different variants for different datasets.

- Incorporation of deep-learning techniques with MOAs to have a more general and robust model, and to use more fundamental data.

- A more realistic financial simulated environment should be used. Most trading simulations used for financial experiments make many assumptions, like latency. More research must go into how to get a near normal trading environment.

Bibliography

[1] The best currency pairs to trade and times to trade them? http://www.learntotradethemarket.com/forex-trading-strategies/what-are-the-best-forex-currency-pairs-to-trade. Accessed: 2018-06-01.

[2] Coefficient of variation (cv). https://www.investopedia.com/terms/c/coefficientofvariation.asp. Accessed: 2018-04-28.

[3] Foreign exchange market. https://en.wikipedia.org/wiki/Foreign_exchange_market. Accessed: 2018-04-28.

[4] How can you calculate volatility in excel? https://www.investopedia.com/ask/answers/021015/how-can-you-calculate-volatility-excel.asp. Accessed: 2018-04-28.

[5] Technical indicators and overlays. http://stockcharts.com/school/doku.php?id=chart_school:technical_indicators. Accessed: 2018-04-28.

[6] Alireza Afshari, Majid Mojahed, and Rosnah Mohd Yusuff. Simple additive weighting approach to personnel selection problem. *International Journal of Innovation, Management and Technology*, 1(5):511, 2010.

[7] Gerald Appel. *Technical analysis: power tools for active investors*. FT Press, 2005.

[8] Frederick Ditliac Atiah and Mardé Helbig. Dynamic particle swarm optimization for financial markets. In *Proceedings of the IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 2337–2344. Bangalore, India, 2018.

[9] Tim Blackwell and Jürgen Branke. Multiswarms, exclusion, and anti-convergence in dynamic environments. *IEEE transactions on evolutionary computation*, 10(4):459–472, 2006.

[10] Tim M Blackwell and Peter J Bentley. Dynamic search with charged swarms. In *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, pages 19–26. Portland, Oregon, USA, 2002.

[11] Diego J Bodas-Sagi, Pablo Fernández, J Ignacio Hidalgo, Francisco J Soltero, and José L Risco-Martín. Multiobjective optimization of technical market indicators. In *Proceedings of the Annual Conference Companion on Genetic and Evolutionary Computation Conference (GECCO): Late Breaking Papers*, pages 1999–2004. ACM, Portland, Oregon, USA.

[12] Jürgen Branke. *Evolutionary optimization in dynamic environments*, volume 3. Springer Science & Business Media, 2012.

[13] Jürgen Branke, Erdem Salihoğlu, and Şima Uyar. Towards an analysis of dynamic environments. In *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, pages 1433–1440. Washington DC, USA, 2005.

[14] Matthew Butler and Dimitar Kazakov. Particle swarm optimization of bollinger bands. In *Proceedings of the con Swarm Intelligence*, pages 504–511. Beijing, China, 2010.

[15] Rodolfo C Cavalcante, Rodrigo C Brasileiro, Victor LF Souza, Jarley P Nobrega, and Adriano LI Oliveira. Computational intelligence and financial markets: A survey and future directions. *Expert Systems with Applications*, 55:194–211, 2016.

[16] An-Pin Chen, Chien-Hsun Huang, and Yu-Chia Hsu. A novel modified particle swarm optimization for forecasting financial time series. In *Proceedings of the International Conference on Intelligent Computing and Intelligent Systems*, volume 1, pages 683–687. Shanghai, China, 2009.

[17] Nai-Fu Chen, Richard Roll, and Stephen A Ross. Economic forces and the stock market. *Journal of business*, pages 383–403, 1986.

[18] Carlos A Coello Coello, Gregorio Toscano Pulido, and M Salazar Lechuga. Handling multiple objectives with particle swarm optimization. *IEEE Transactions on evolutionary computation*, 8(3):256–279, 2004.

[19] Robert W Colby and Thomas A Meyers. *The encyclopedia of technical market indicators*. Dow Jones-Irwin Homewood, IL, 1988.

[20] Ricardo de Almeida, Gilberto Reynoso-Meza, and Maria Teresinha Arns Steiner. Multi-objective optimization approach to stock market technical indicators. In *Proceedings of the Evolutionary Computation (CEC), 2016 IEEE Congress on*, pages 3670–3677. Denver, Colorado, USA, 2016.

[21] Kalyanmoy Deb, S Karthik, et al. Dynamic multi-objective optimization and decision-making using modified NSGA-II: a case study on hydro-thermal power scheduling. In *International conference on evolutionary multi-criterion optimization*, pages 803–817. Springer, 2007.

[22] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[23] Julien L Duhain. *Empirical analysis of particle swarm optimization on dynamic environments*, volume 1. Master's thesis, University of Pretoria, 2012.

[24] Russ C Eberhart and Yuhui Shi. Comparing inertia weights and constriction factors in particle swarm optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, volume 1, pages 84–88. La Jolla, CA, USA, 2000.

[25] Russell Eberhart and James Kennedy. A new optimizer using particle swarm theory. In *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pages 39–43. Nagoya, Japan, 1995.

[26] Russell C Eberhart and Yuhui Shi. Tracking and optimizing dynamic systems with particle swarms. In *Proceedings of the IEEE Congress on Evolutionary Computation (IEEE Cat. No. 01TH8546)*, volume 1, pages 94–100. Seoul, South Korea, 2001.

[27] Andries P Engelbrecht. *Fundamentals of computational swarm intelligence*. John Wiley & Sons, 2006.

[28] Eugene F Fama. Random walks in stock market prices. *Financial analysts journal*, 51(1):75–80, 1995.

[29] Marco Farina, Kalyanmoy Deb, and Paolo Amato. Dynamic multiobjective optimization problems: test cases, approximations, and applications. *IEEE Transactions on evolutionary computation*, 8(5):425–442, 2004.

[30] Nobert Funke and Andrea Goldstein. Financial market volatility. *Intereconomics*, 31(5):215–220, 1996.

[31] Mardé Greeff and Andries P. Engelbrecht. *Dynamic Multi-objective Optimisation Using PSO*, pages 105–123. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[32] Mardé Greeff and Andries P Engelbrecht. Dynamic multi-objective optimisation using pso. In *Multi-Objective Swarm Intelligent Systems*, pages 105–123. Springer, 2010.

[33] Mardé Helbig. Solving dynamic multi-objective optimisation problems using vector evaluated particle swarm optimisation. *University of Pretoria, Faculty of Engineering, Built Environment and Information Technology*, 2012.

[34] Ching-Lai Hwang and Kwangsun Yoon. Methods for multiple attribute decision making. *Multiple attribute decision making*, pages 58–191, 1981.

[35] Razan A Jamous, Essam El Seidy, and Bayoumi Ibrahim Bayoum. A novel efficient forecasting of stock market using particle swarm optimization with center of mass based technique. *International Journal of Advanced Computer Science and Applications*, 7(4):342–347, 2016.

[36] Deb Kalyanmoy. *Multi objective optimization using evolutionary algorithms*. John Wiley and Sons, 2001.

[37] N Kasabov, L Erzegovesi, M Fedrizzi, A Beber, and D Deng. Hybrid intelligent decision support systems and applications for risk analysis and discovery of evolving economic clusters in europe. In *Future Directions for Intelligent Systems and Information Sciences*, pages 347–372. Springer, 2000.

[38] James Kennedy and Rui Mendes. Population structure and particle swarm performance. In *Proceedings of the IEEE on Evolutionary Computation, 2002*, volume 2, pages 1671–1676. olulu, HI, USA, 2002.

[39] Youngmin Kim and David Enke. Developing a rule change trading system for the futures market using rough set analysis. *Expert Systems with Applications*, 59:165–173, 2016.

[40] Hak-Keung Lam. *Computational intelligence and its applications: evolutionary computation, fuzzy logic, neural network and support vector machine techniques*. World Scientific, 2012.

[41] Dome Lohpetch and David Corne. Multiobjective algorithms for financial trading: Multiobjective out-trades single-objective. In *Proceedings of the IEEE on Evolutionary Computation (CEC)*, pages 192–199. New Orleans, LA, USA, 2011.

[42] John J Murphy. *Study Guide for Technical Analysis of the Futures Markets: A Self-training Manual*. New York institute of finance New York, 1987.

[43] Jovita Nenortaite and Rimvydas Simutis. Adapting particle swarm optimization to stock markets. In *Proceedings of the 5th International Conference on Intelligent Systems Design and Applications*, pages 520–525. Warsaw, Poland, 2005.

[44] Azadeh Nikfarjam, Ehsan Emadzadeh, and Saravanan Muthaiyah. Text mining approaches for stock market prediction. In *Proceedings of the 2nd International Conference on Computer and Automation Engineering (ICCAE)*, volume 4, pages 256–260. Singapore, 2010.

[45] Adama Ouattara, Luc Pibouleau, Catherine Azzaro-Pantel, and Serge Domenech. Economic and environmental impacts of the energy source for the utility production system in the hda process. *Energy conversion and management*, 74:129–139, 2013.

[46] Murat Ozturk, Ismail Hakki Toroslu, and Guven Fidan. Heuristic based trading system on forex data using technical indicator rules. *Applied Soft Computing*, 43:170–186, 2016.

[47] Anna Rakitianskaia and Andries P Engelbrecht. Training neural networks with pso in dynamic environments. In *Proceedings on the IEEE Congress on Evolutionary Computation, 2009. CEC'09*, pages 667–673. Trondheim, Norway, 2009.

[48] Milan R Rapaić, Željko Kanović, and Zoran D Jeličić. Discrete particle swarm optimization algorithm for solving optimal sensor deployment problem. *Journal of Automatic Control*, 18(1):9–14, 2008.

[49] Carlo R Raquel and Prospero C Naval Jr. An effective use of crowding distance in multiobjective particle swarm optimization. In *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, pages 257–264. Washington DC, USA, 2005.

[50] Asanga Ratnaweera. Particle swarm optimization with self-adaptive acceleration coefficients. In *Proceedings of the International Conference on Fuzzy Syst. & Knowledge Discovery (FSKD 2002), Singapore, Nov.*, volume 1, pages 264–268, 2002.

[51] Peter J Rousseeuw, Ida Ruts, and John W Tukey. The bagplot: a bivariate boxplot. *The American Statistician*, 53(4):382–387, 1999.

[52] Sepehr Sanaye and Davood Modarrespoor. Thermal-economic multiobjective optimization of heat pipe heat exchanger for energy recovery in hvac applications using genetic algorithm. *Thermal Science*, 18(suppl. 2):375–391, 2014.

[53] J David Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the First International Conference on Genetic Algorithms and Their Applications, 1985*. Pittsburgh, Pennsylvania, USA, 1985.

[54] Arlindo Silva, Ana Neves, and Ernesto Costa. An empirical comparison of particle swarm and predator prey optimisation. In *Proceedings of the Irish Conference on Artificial Intelligence and Cognitive Science*, pages 103–110. Limerick, Ireland, 2002.

[55] Frans Van den Bergh and Andries Petrus Engelbrecht. Cooperative learning in neural networks using particle swarm optimizers. *South African Computer Journal*, 2000(26):84–90, 2000.

[56] ET van Zyl and AP Engelbrecht. Comparison of self-adaptive particle swarm optimizers. In *Proceedings of the IEEE on Swarm Intelligence (SIS)*, pages 1–9. Orlando, FL, USA, 2014.

[57] Fei Wang, LH Philip, and David W Cheung. Complex stock trading strategy based on particle swarm optimization. In *Proceedings of the IEEE Conference on Computational Intelligence for Financial Engineering & Economics (CIFEr)*, pages 1–6. New York, USA, 2012.

[58] Fei Wang, LH Philip, and David W Cheung. Combining technical trading rules using particle swarm optimization. *Expert Systems with applications*, 41(6):3016–3026, 2014.

[59] J Welles Wilder. *New concepts in technical trading systems*. Trend Research, 1978.

Appendix A

Acronyms

This appendix provides an alphabetical listing of all acronyms used in this dissertation. Each acronym is typeset in bold and its meaning is provided alongside

ANN

Artificial Neural Network 43, 46

BB

bollinger bands 44

CI

computational intelligence 1, 2, 12, 47

cPSO

charge PSO 2, 21, 24, 59, 61, 69, 99

CV

coefficient of variation 61, 62, 64, 65, 69, 74, 80, 90, 99

DE

differential evaluation 46, 47

DM

decision model 2–4, 25, 33, 72, 74, 77, 79, 86, 88, 93, 95, 98–100

DMOA

dynamic multi-objective algorithm 2–4, 12, 25, 36, 79, 80, 98–100

DMOO

dynamic multi-objective optimization 12, 16, 17, 25, 80, 100

DMOP

dynamic MOP 2, 16

DMS-MOPSO

dynamic multi-swarm multi-objective particle swarm optimization 2–5, 17, 18, 24, 49, 81, 83, 88, 90, 94, 98, 100

DNSGA-II

dynamic non-dominated sorting genetic algorithm II 2, 5, 25, 29, 30, 56, 57, 71, 73, 74, 77, 79, 80, 90, 99

DVEPSO

dynamic vector-evaluated particle swarm optimization 2, 5, 25, 31, 32, 71–74, 79, 90, 99

EMA

exponential moving average 39, 41, 62, 64, 65, 67, 68, 99

EMH

efficient-market hypothesis 2, 38

Forex

foreign exchange 1–5, 9, 12, 13, 21, 37, 38, 43–47, 49, 50, 54, 59, 64, 69–71, 79, 80, 98, 99

GA

genetic algorithm 43, 44

GP

genetic programming 46, 47

GRA

gray relational analysis 35, 72, 79, 86, 99

GRC

gray relational coefficient 35

HPF

highest profit 36, 72, 77, 86, 95

MA

moving average 38, 39, 41, 44

MACD

moving average convergence divergence 38, 39, 41, 62–65, 67–69, 74, 99

MOA

multi-objective algorithm 3–5, 25, 26, 43, 46–49, 71, 79, 90, 95, 98–100

MOO

multi-objective optimization 6, 9, 10, 26, 27, 33, 47, 49, 53, 57, 72, 82, 94, 100

MOP

multi-objective optimization problem 2, 4, 6, 9–11, 15, 16, 25, 31, 36, 46–48, 70, 71, 81–83, 98, 100

MOPSO

multi-objective particle swarm optimization 26–28, 30, 81, 83

MOPSO-CD

multi-objective particle swarm optimization algorithm with crowded distance 2, 5, 25–28, 36, 46, 71, 72, 74, 76, 79, 90, 99

NSGA-II

non-dominated sorting genetic algorithm II 29, 46

POF

Pareto front 10, 11, 16, 25, 26, 28, 30, 47, 79, 99, 110, 113, 115

POS

Pareto-optimal set 10, 11, 25, 33, 35, 36, 47, 48, 53, 74, 77, 83, 88, 110, 113, 115

PSO

particle swarm optimization 2–5, 12, 18–21, 24–29, 36, 43–45, 56, 57, 59, 60, 65, 69, 70, 72, 81, 83, 98

qPSO

quantum PSO 2, 5, 21, 24, 59, 61–70, 99

RSI

relative strength index 40, 41, 62, 64, 65, 67, 68, 74, 99

SAW

simple additive weighting 35, 72, 86

SMA

simple moving average 39, 41, 62–66, 68, 99

SOA

single-objective optimization algorithm 2–4, 12, 15, 26, 43, 45, 48, 49

SOO

single-objective optimization 4, 17, 18, 25, 27, 36, 47, 49, 57, 83

SOP

single-objective optimization problem 6, 7, 9–12, 16, 25, 48

sPSO

standard particle swarm optimization 5, 43–45, 57, 59, 61–64, 67–69, 99

SUM

objective sum 36, 72, 79, 86, 95, 99

TI

technical indicator 1, 2, 5, 12, 37, 38, 40, 43–47, 49, 50, 56, 57, 59, 60, 62, 65, 69, 70, 72, 74, 76, 79, 93, 98, 99

TOPSIS

Technique for Order of Preference by Similarity to Ideal Solution 33, 72, 86

tPSO

time-series particle swarm optimization 5, 44, 59, 61–64, 68, 69, 99

TR

trade rule 37, 40, 41, 44–46, 50, 57

TRB

trading range break-out 44

VEPSO

vector-evaluated particle swarm optimization 31, 83

Appendix B

Symbols

This appendix lists the mathematical symbols used throughout this dissertation, and their definitions. The symbols used within each chapter are listed under separate sections. Each section lists only newly introduced symbols.

Chapter 2: Background

A	External Achieve
A^+	Ideal solution
A^-	Euclidean distance to the worst solution, A^-
A^-	Worst solution
As	The Simple additive weihting value
C_i	The highest closeness value
f^1, f^2	Objective functions
F_{ij}	Normalised objectives matrix

f_j^{max}	The maximum objective function values
f_j^{min}	The minimum objective function values
I_{ij}	The point difference between two solutions
I_j^{max}	The maximum point difference, j
I_j^{min}	The minimum point difference, j
m	Number of objective function
n	Number of bjective functions
n_s	Number of particles/solutions
nP	New population created from R
P	Population or generation
P_{i+1}	Next Population or generation after each iteration
R	Combined parents and offsprings
S_{i+}	Euclidean distance to the ideal solution, A^+
w_j	Weight for an objective function
wF_{ij}	A weighted normalised matrix
$x_i^{'}$	Individual in population nP
(POF^*, t)	POF found at time step, t
(POS^*, t)	The optimum POS found at time step, t
$\mathbb{R}$	One-dimensional real space
$\mathbb{R}^{n_x}$	n_x dimensional real space
ω	Initia weight

x*(t)	Optimum at time step, t
ζ	Severity of change
a_i	Acceleration term of the charged PSO velocity update equation
c_1	Cognitive coefficient
c_2	Social coefficient
F	Feasible solution from the seach space
f	Objective function
f_k	The k-th objective function
g_i	Inequality constraints
h_i	Equality constraints
i	Index
i	Position of the particle in its population
n_x	Number of decision variable
n_x	Number of decision variables
POF	Pareto front
POS	Pareto-optimal set
Q_i	Charged magnitude of particle i
r_1, r_2	Random number
r_{cloud}	Radius of the quantum cloud
R_c	Core radius
R_p	Perception limit

S	Seach space
t	Time step
$v_i(t)$	Velocity of particle at time step t
x^*	Global optimum
$x_i(t-1)$	Prevoius position of particle at time step t
$x_{gBest,i}(t)$	Global best position of the particle at time step t
$x_i(t)$	Position of particle at time step t
$x_{pBest,i}(t)$	Personal best position of the particle at time step t
gBest	Global best
pBest	Personal best

Chapter 3:Optimization Problems and Simulation for Financial Markets

$\acute{v_i}(t)$	The newly calculated velocity
$\acute{x_i}(t)$	Newly calculated position
δ	Random number between 0 and 1
ω_{max}	Maximum initia weight
ω_{min}	Minimum initia weight
avG	The average gain
$avG(prev)$	The previous average gain
Avg_{netPF}	Average net profit
$avgPF$	Average Net Profit
avL	The average loss

$avL(prev)$	The previous average loss
$c_{1,max}$	Minimum cognitive coefficient
$c_{1,min}$	Maximum cognitive coefficient
$c_{2,max}$	Maximum social coefficient
$c_{2,min}$	Minimum social coefficient
cG	Current gain
cL	Current loss
cP	Current Price
D	Domain of TIs
lim_a	RSI buy limit
lim_b	RSI buy limit
lim_c	RSI sell limit
lim_s	RSI sell limit
$n1, n2, n3$	Time frame 1,2 and 3
n_t	The maximum number of iterations
$netPF$	Net Profit
NO_{trades}	Number of Trades
pF	Profit after a trade
$prevPrice$	Previous closing price
$prevSignal$	Previously generated signal (Buy or Sell)
RS	Relative strength

$rsin$	The look back parameter or time frame
$RSIv$	Relative strength indicator value
T_{Cost}	Total transaction cost
t_{cost}	Transaction cost
T_n	Total number of trades
$tCost$	Transaction cost after a trade
$tGains$	Total Gains
$tLoss$	Total loss
v_i	Velocity of particle, i
v_{max}	Maximum velocity
v_{min}	Minimum velocity
x_i	Position of particle, i
D_{min},D_{max}	The minimum and maximum values of the domain
cP	The closing price
$EMA(prev)$	The exponential moving average of the previous day
$macdL$	MACD line
n	Number of time frames or period
per	The percentage which determines the weight of recent cP
$sigLine$	Signal lines

CV	CV
$gTrade$	Winning trades
I	Initial investment
n_t	The maximum number of iterations
R	Total runs which is 30
T_{cost}	Total transaction cost
$tGains$	Total gains
Tl	Total number of losses
$tLoss$	Total losses
Tn	Total number of trades
Tw	Total number of wins

Chapter 6: Dynamic Multi-Swarm Multi-Objective PSO

cI, mI	Current iteration and maximum iteration
gA	External or Global archive
lA	Local archive
ld	Sub-swarm leader
n	Index of sub-swarm
S_{a+1}	Next sub-swarm
S_a	Sub-swarm n or current sub-swarm

Appendix C

Derived Publications

This appendix lists the publications derived from the work presented in this thesis.

- Frederick Ditliac Atiah and Mardé Helbig. Dynamic particle swarm optimization for Fnancial markets. In *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 2337 - 2344. IEEE, 2018.
- Frederick Ditliac Atiah and Mardé Helbig. Effects of Decision Models on Dynamic Multi-objective Optimization Algorithms for Financial Markets. In *IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2019.
- Frederick Ditliac Atiah and Mardé Helbig (in press). Dynamic particle swarm optimization for a fnancial time series. In *Journal of Banking and Financial Technology (JBFT)*, 2019.