Vector-Evaluated Particle Swarm Optimization using Co-operative Swarms

Justin Maltese

Department of Computer Science

Submitted in partial fulfillment
of the requirements of:

Cosc 4F90 - Honors Bachelor of Science Thesis

Faculty of Mathematics and Science, Brock University
St. Catharines, Ontario

To my parents, whose love and support I will always cherish.

Abstract

Many real-world optimization problems contain multiple (often conflicting) goals to be optimized concurrently. Vector evaluated particle swarm optimization (VEPSO) is a variant of the traditional particle swarm optimization (PSO) algorithm which employs multiple swarms to combat multi-objective problems (MOPs). Each swarm optimizes a single objective and information is passed between swarms using a knowledge transfer strategy (KTS). Traditionally, the swarms perform optimization using the original PSO algorithm. However, recently three powerful variants of PSO which utilize co-operative principles were shown to improve PSO performance in single-objective environments. This thesis investigates the feasibility of using these co-operative PSO variants within VEPSO swarms to improve performance along with exploring the impacts on KTS efficiency. A comparison of the highest performing co-operative algorithm-KTS combinations is also made against well-known multi-objective algorithms. The results indicate that co-operation is a powerful tool which enhances hypervolume/solution distribution and allows VEPSO to successfully compete with top multi-objective optimization algorithms.

Acknowledgements

I would like to extend my gratitude to the following people, for without them this work would not have been possible:

- Prof. Beatrice M. Ombuki-Berman for her excellent guidance and supervision throughout the entire duration of this work.
- Prof. Andries P. Engelbrecht for his collaboration during the research process.
- The Computer Science Department at Brock University for their exceptional level of education provided to me.
- The Computational Intelligence Library (CIlib) team for providing a means to perform research via their public codebase.

J.P.M

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

Every day, humans face a multitude of difficult problems which demand solutions. Some of the more challenging problems require concurrent optimization of multiple distinct objectives. These types of problems are referred to as multi-objective problems (MOPs). It is common for objectives within these problems to conflict, making it difficult (and often impossible) for all objectives to be fully optimized simultaneously. MOPs are commonly encountered in fields such as engineering [58], business [5] , mathematics [33] and physics [14]

Particle swarm optimization (PSO) is an optimization algorithm which simulates flocking behaviour in nature (such as a bird flock or fish school). Through it's powerful explorative capabilities, PSO is able to find good solutions for a variety of problems [31]. While the original PSO was mainly used for single-objective problems, it has also been applied to multi-objective problems [45, 9].

The first application of PSO to multi-objective problems was introduced in [44] as vector evaluated particle swarm optimization (VEPSO). VEPSO utilizes multiple swarms to perform optimization by allocating a single swarm to each objective. Each swarm attempts to optimize it's respective objective while simultaneously passing information to other swarms. The method by which information is passed between swarms is referred to as a knowledge transfer strategy (KTS). Using a KTS ensures

that the problem as a whole is optimized due to each swarm being influenced by the successes of other swarms (while also maintaining a bias to their assigned objective).

Although the original VEPSO algorithm performs well [45], there is potential for improvement. Various researchers have experimented with modifying different VEPSO mechanics in an attempt to improve overall performance of the algorithm. Harrison et al. [23] explored the use of different knowledge transfer strategies while Lim et al. [32] proposed an improvement to the algorithm in which non-dominated solution guides were used. One currently unexplored area is the optimization algorithm used by the individual swarms. Traditionally, each swarm uses the original PSO algorithm to perform optimization. However, recently the performance of PSO has been improved by applying concepts inspired from cooperative systems [16]. Utilizing this co-operative variant (known as CPSO-S) in place of traditional PSO holds great potential for improving the overall performance of the VEPSO algorithm.

1.1 Objectives

This thesis explores the effects on VEPSO performance when swarms optimize using the CPSO-S algorithm as opposed to regular PSO. In addition to the original CPSO-S algorithm, co-operative variants such as CPSO-S_K and CPSO-H_K are also tested. The effects of using CPSO-S (and it's variants) within VEPSO on KTS performance is also analyzed. The traditional ring and random KTSs are experimented on along with two powerful new hybrid strategies recently introduced in [23], the PCX Archive and PCX GBest KTS. A comparison of the two best algorithm-KTS combinations is also performed against well-known multi-objective optimization algorithms. The performance metrics used are solution distribution [20] and hypervolume [63].

To fulfill the proposed objectives, this work aims to achieve the following goals:

- Perform a direct performance comparison between using CPSO-S/CPSO-S_K/CPSO-

H_K swarms in VEPSO against using regular PSO swarms

- Establish the benefit, if any, of using CPSO-S/CPSO-S_K/CPSO-H_K swarms within VEPSO
- Provide specific problem scenarios, including function shapes and modalities, of when it is advantageous for swarms to utilize co-operative principles
- Observe the performance differences between CPSO-S, CPSO-S_K and CPSO-H_K in the context of VEPSO and determine which variant performs best, if any
- Analyze KTS performance when co-operation is present in VEPSO
- Observe whether the conclusions regarding KTS performance in [23] hold when co-operation is present in VEPSO
- Evaluate whether the highest performing algorithm-KTS combinations are able to compete with well-known multi-objective optimization algorithms
- Provide KTS/algorithm recommendations for scenarios involving solution distribution prioritization and hypervolume prioritization

1.2 Thesis Organization

The remainder of this thesis is organizes as follows:

Chapter 2 contains all relevant background information of the study. This includes an overview of multi-objective optimization along with algorithms such as PSO, VEPSO and CPSO. An introduction to the various test suites used is also given.

Chapter 3 provides an explanation of how VEPSO with CPSO swarms is implemented. This includes detailing their interactions with each other.

Chapter 4 describes the experimental setup used in this study. Parameters for PSO/CPSO and VEPSO are listed. An overview of the benchmark functions used

in the study is provided. The two performance measures used (hypervolume and solution distribution) are also introduced. Finally, the statistical methods of analysis are described.

Chapter 5 presents the results of all experiments performed. Analysis and discussion of the results is also performed.

Chapter 6 discusses conclusions drawn and future work.

Chapter 2

Background

This chapter provides an overview of all background information relevant to this thesis. Topics covered include multi-objective optimization and pareto optimality, PSO, VEPSO, CPSO and the WFG/ZDT toolkits.

2.1 Multi-objective Optimization

Multi-objective optimization refers to the concurrent optimization of two or more objectives. Conflicting objectives reveal the true difficulty of these optimizations, as trade-offs must be made in some objective(s) in order to further optimize another objective. For non-trivial problems this makes it nearly impossible to fully optimize all objectives simultaneously. An example of this would be the classic multi-objective problem (MOP) of minimizing fuel consumption while maximizing vehicle performance, seen in previous literature [27]. A tradeoff would be required in this scenario, as an improvement in one objective would likely degrade the other. Multi-objective optimization problems[1] can be formulated as follows:

$$min(f_1(x), f_2(x), ..., f_n(x)), x \in X \tag{2.1}$$

[1]Minimization problems are assumed for this section. When dealing with maximization problems, all objectives would be maximized as opposed to minimized.

where n is an integer representing the number of objectives and X is the feasible set of decision vectors, commonly defined by constraint functions. The objective function is defined as:

$$f : X \to \mathbb{R}^n, f(x) = (f_1(x), ..., f_n(x))^T \tag{2.2}$$

A vector $z^* := f(x^*) \in \mathbb{R}^n$ is referred to as an objective vector, subject to $x^* \in X$. Due to the conflicting nature of objectives commonly seen in MOPs, it is very rare for a feasible solution which optimizes all functions simultaneously to exist. Instead, the focus of multi-objective optimization is on finding feasible solutions which cannot improve any objective further without worsening any other objective. These solutions are referred to as Pareto optimal. The concept of Pareto optimality is further explored in Section 2.1.1.

As a consequence of the trade-offs which occur during multi-objective optimization, a decision maker is required to give optimization of certain objectives priority over others. Determining which objectives are the most desirable to optimize often requires prior knowledge in the field. This is known as an *a priori* approach in which the algorithm would make trade-offs during the optimization process. *A priori* approaches deal with assigning objective bias before optimization occurs, allowing the algorithm to make trade-offs during the optimization. This approach generally give the decision maker a small yet highly specialized set of solutions.

The opposite of an *a priori* approach would be an *a posteriori* approach in which tradeoffs would be made manually by the decision maker after the optimization occurs. *A posteriori* approaches give a large set of generalized solutions to the decision maker. A balanced approach which yields solution sets that are not highly specialized or generalized is known as *interactive*. In this approach, the decision maker would dynamically guide the optimizer into more desirable areas of the search space during the optimization.

2.1.1 Pareto Optimality

Pareto optimality (also referred to as Pareto efficiency) in it's most general form refers to a state in which resources are distributed such that a single individual cannot improve without negatively affecting at least one other individual. Pareto optimality originated from economic observations by Vilfredo Pareto (1848 - 1923) and continues to be applied to the field of economics [51]. The concept of pareto optimality is extremely important to multi-objective optimization. Within the context of a MOP, a Pareto optimal solution is defined to be a solution in which no further improvements can be made for any objective without worsening some other objective. Solutions which are not Pareto optimal are considered undesirable, thus the end goal of any MOP optimization is to obtain the set of Pareto optimal solutions, referred to as the Pareto Front.

A formal definition [59] of the Pareto Front is as follows:

$$PF^* = \{\vec{y^*} \in A | \nexists \vec{y} \in S : \vec{y} \prec \vec{y^*}\} \tag{2.3}$$

where A represents the objective space and $\prec$ is a strict dominance relation such that $\vec{y} \prec \vec{y^*}$ if $\vec{y_i} \leq \vec{y_i^*}$ for all i and $\vec{y_i} < \vec{y_i^*}$ for some i. A sample Pareto front is shown in Figure 2.1[2]. The front itself is highlighted in red and represents all solutions that are not dominated by any other solutions. Note that solution K is not part of the Pareto front as it is dominated by both solution D and E.

2.2 Particle Swarm Optimization

Particle swarm optimization (PSO) is a nature-inspired metaheuristic optimization algorithm developed by Kennedy and Eberhart [31]. It mimics the flocking behavior

[2]Original author is Noel Rosario. Image is used under the Creative Commons Attribution-Share Alike 3.0 Unported License.

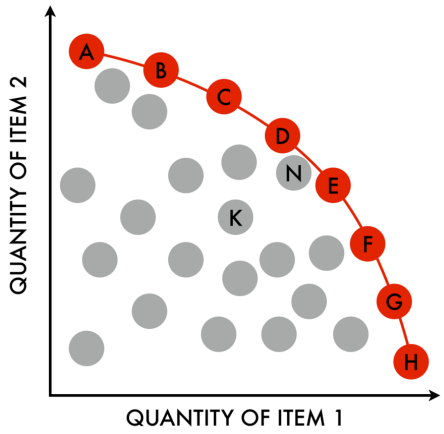

Figure 2.1: A Pareto front for a maximization problem with two objectives.

of birds and the schooling behaviour of fish. The original intent of the PSO algorithm was to observe these behaviours as they would be found in nature. However, it was soon realized that PSO could be used as a powerful optimization algorithm for non-linear functions. PSO has been used in many real world applications including the optimization of power plants [24], antennas [37], electromagnetic devices [25] and security identification systems [30].

The PSO algorithm contains a population of simple entities known as *particles*. Particles move through the n-dimensional search space using a flocking behaviour influenced by other swarm members. The position of each particle represents a candidate solution to the problem which is being optimized by the swarm. The desirability of a particle position is determined by a fitness function relating to the problem at hand. The swarm flocks to more desirable positions iteratively, with the end goal being convergence around the most desirable (globally optimal) position in the search space.

Each particle is responsible for keeping track of it's current position, personal best position and current movement velocity. These are all required to determine the movement of the particle at each iteration. Particle movement is primarily influenced by these three factors:

- The current velocity of the particle
- The personal best position of the particle
- The global best position of the swarm

The above factors are not necessarily of equal importance, as weight values are assigned to each of them. The *inertial* weight controls particle velocity influence, the *cognitive* weight directs particles to the personal best particle position and the *social* weight guides particles to the best position of the swarm. In addition to this, random variables to help overcome local minima are also included in the weighting of the last two factors in the list. The entire PSO algorithm is shown in Algorithm 1.

Algorithm 1 PSO

```
1: Initialize particles randomly
2: while current iteration < max iterations do
3:     for each particle j in the swarm do
4:         if f(j.x⃗) > f(j.y⃗) then
5:             j.y⃗ = j.x⃗
6:         end if
7:         if f(j.y⃗) > f(ŷ) then
8:             ŷ = j.y⃗
9:         end if
10:    end for
11:    for all particles in the swarm do
12:        Calculate velocity update using eqn (2.4)
13:        Update particle position using eqn (2.5)
14:    end for
15: end while
```

The above algorithm makes use of two update equations:

$$\vec{v} = \alpha\vec{v} + \omega C_1(\vec{y} - \vec{x}) + \lambda C_2(\hat{y} - \vec{x}) \tag{2.4}$$

$$\vec{x} = \vec{x} + \vec{v} \tag{2.5}$$

Where C_1 and C_2 are randomly generated numbers in the range [0,1], α is the inertial weight, ω is the cognitive weight, λ is the social weight, $\vec{v}$ is the particle

velocity, $\vec{y}$ is the particle personal best position, $\hat{y}$ is the swarm global best position and $\vec{x}$ is the current particle position. Since these update equations are meant for continuous values, it can be very inefficient to use the original PSO algorithm for problems with discrete values. However, previous literature has introduced effective modifications of the update equations to accomodate discrete values [26, 49, 47]. The weights in Equation 2.4 are commonly set to values derived from Clerc's velocity update equation modifications [7] which are:

$$\vec{v} = K(\vec{v} + \omega C_1(\vec{y} - \vec{x}) + \lambda C_2(\hat{y} - \vec{x})) \tag{2.6}$$

Note that K is a constriction factor calculated as:

$$K = \frac{2}{|2 - \phi - \sqrt{\phi^2 - 4\phi}|} \tag{2.7}$$

In Equation 2.7, ϕ is calculated as $(\lambda + \omega)$ and must be greater than 4. By distributing K in Equation 2.6, the missing inertial weight can be included. Choosing $\phi = 4.1$ to satisfy Clerc's formula has been shown to empirically perform well [3], which corresponds to the values $\alpha = 0.729$, $\omega = 1.494$ and $\lambda = 1.494$. These parameter values are very popular in PSO literature, however other values have also been proposed [50, 52].

2.3 Co-operative PSO

The original PSO algorithm, along with many other stochastic optimization algorithms, suffers from a problem called the "curse of dimensionality". In general terms, the curse of dimensionality implies that algorithmic performance wanes as problem dimensionality increases. This can cause the algorithm to quickly become stuck in local minima. Due to the curse of dimensionality, PSO essentially takes one step

forward and two steps backward [55].

To illustrate this phenomena, consider the problem of minimizing $\sum_{i=0}^{n} ln(|x_i|)$ where x is a three-dimensional vector and n represents the length of the vector (three). Observe that the fitness of the problem is optimized[3] as the dimension values move closer to 0. If a particle is located at the position (10,10,10), its fitness value would be approximately 6.907. If this particle moved to the position (13,6,12), its new fitness value would be 6.842 approximately. This would mean that the new position would be regarded as more desirable than the original position, despite the particle moving away from the optimal values in two of the three dimensions. This is a severe problem and can lead to major performance loss in the PSO algorithm.

In an attempt to overcome the issue, Van den bergh and Engelbrecht [54] proposed a new variant of PSO based on concepts observed in co-operative systems. This variant later became known as CPSO-S. CPSO-S attempts to improve the original algorithm by partitioning the search space into lower dimensional subspaces. However, a requirement for this method is that the optimization algorithm must be able to search every possible region of the search space. The search space partioning is performed by splitting the single swarm which is attempting to optimize a vector of n dimensions into n swarms of each attempting to optimize a single dimension. Note that the number of particles in each sub-swarm remains the same as the number of particles used in the original swarm[4]. The following dilemmas [55] arise as a direct result of these modifications:

- Selection: In order to evaluate fitness, re-construction of the original n dimensional vector must be possible. Therefore for each swarm, there must be a way to select which particle will be selected for representation of the vector dimension that the respective swarm is optimizing. The obvious solution is to choose

[3]One should note that the problem does not have a single value that can be regarded as most optimal due to the fact that ln(0) is undefined.

[4]Essentially, the number of particles used is the original number of particles multiplied by the number of swarms.

the best particle via some metric. However, this is not necessarily optimal as it could lead to greedy behaviour.

- Credit Assignment: How much credit should each swarm be awarded when they contribute to a better solution? In practice, each dimension may not carry equal weighting and thus it may be incorrect to assign equal credit to each swarm.

As shown in the selection problem presented above, CPSO-S must be able to evaluate the fitness of particles. Since each particle is now one-dimensional, a method which converts a 1-D solution into a complete candidate solution is required. CPSO-S accomplishes this by maintaining a *context vector* - a vector whose components consist of the global bests of each swarm, respectively[5] [53]. Note that initially the context vector consists of random dimensions selected from each swarm. Fitness evaluation consists of substituting particle positions into the context vector, detailed further later on in this section.

Algorithm 2 CPSO-S

```
1: Initialize n 1-D PSO swarms
2: while current iteration < max iterations do
3:     for each swarm i do
4:         for each particle j in i do
5:             if f(i, j.x⃗) > f(i, j.y⃗) then
6:                 j.y⃗ = j.x⃗
7:             end if
8:             if f(i, j.y⃗) > f(i, i.ŷ) then
9:                 i.ŷ = j.y⃗
10:            end if
11:            Do regular PSO position updates on j according to (2.4) and (2.5)
12:        end for
13:    end for
14: end while
```

Algorithm 2 provides an overview of the CPSO-S algorithm. Note that the current position of a particle is represented as $\vec{x}$, the personal best position of a particle is

[5]This implies that dimensionality of the vector is identical to the original problem.

represented as $\vec{y}$, and the global best position of a swarm is represented as $\hat{y}$. Fitness calculation of particle position x in swarm y, denoted *f(y,x)*, is performed as follows:

1. Create a clone of the context vector, denoted $\vec{S}$.
2. Take $\vec{S}$ and replace the value at index y with the 1-D positional value x.
3. Use a fitness measure to evaluate the modified $\vec{S}$ vector. The fitness of particle position x corresponds to this value.

The largest advantage of using CPSO-S over standard PSO is that only one component is modified at a time. The fitness of a solution is calculated after each vector component is updated as opposed to when the entire vector has changed. This yields the desired fine-grained search which essentially prevents the "two steps forward, one step back" scenario [55]. Solution diversity is also significantly increased due to the large amount of combinations [54]. However, using CPSO over PSO results in the number of function evaluations per iteration increasing[6] significantly.

2.3.1 Variants

Although CPSO-S was empirically determined to perform well [55], it was found that the algorithm suffered performance degradation in certain scenarios [54]. Specifically, when components of the solution were related to each other CPSO-S performed unnecessary additional work over regular PSO. An obvious solution to this problem would be to group correlated variables together, however in practice variable relationships are not always known *a priori*. Van den Bergh and Engelbrecht [55] introduced a variant of CPSO-S named CPSO-S_K which addressed the variable correlation problem. CPSO-S_K is similar to CPSO-S except that it splits the n-dimensional search

[6] Increase can be mitigated by lowering the number of particles used. When comparing PSO and CPSO, an equal number of particles across both algorithms is recommended to ensure unbiased comparisons.

space into k parts arbitrarily. Each swarm therefore optimizes $\frac{n}{k}$ dimensions with the hope that related dimensions are optimized together.

The dimension vector split of CPSO-S and CPSO-S_K introduced a stagnation problem that is not present in the original PSO algorithm. Occasionally, these algorithms are deceived into becoming stuck in locations which are not locally/globally optimal. This problem only occurs when deception is present in the optimization function. An additional variant, CPSO-H_K designed to address this problem was included in [54]. CPSO-H_K serves as a hybrid strategy in which CPSO-S_K and regular PSO are executed concurrently. The addition of regular PSO is ideal, since it has the ability to overcome pseudo-optimal locations whereas CPSO-S_K does not. Algorithm 3 presents an overview of the CPSO-H_K algorithm.

Note that CPSO-H_K uses a knowledge-transfer strategy to exchange information between the CPSO-S_K and PSO algorithm. This information exchange is a form of co-operation described in [6]. Knowledge transfer is done by simply injecting the best solution from CPSO-S_K into the PSO swarm and vice-versa. Injection requires overwriting a particle, however constraints[54] as to which particles can be overwritten exist to ensure a diverse set of solutions. These constraints are as follows:

1. The particle chosen cannot be the best particle.

2. The particle chosen must be a legal candidate. Only half of the total particles are deemed candidates. These are chosen *a priori*.

It is worth noting that other knowledge-transfer strategies other than the one outlined above are also possible since the knowledge flow of CPSO-H_K is flexible. It is also possible to structure CPSO-H_K as an algorithm which executes CPSO-S_K until it has become trapped, and then switches to PSO [54]. However, for real-world problems it would be difficult to design heuristics capable of detecting when a switch is appriopriate.

Yao and Li have built upon the success of these co-operative PSO variants by adapting them for large-scale problems. Their CCPSO [36] and CCPSO2 [35] algorithms were created to tackle problems with high dimensionalities (100-2000 variables) by using a variable grouping technique called *random grouping*. Both algorithms performed favourably against well-known PSO strategies for high-dimensional problems [35]. Additional work on incorporating co-operative principles into PSO has also been done in [17, 16, 46].

Many other co-operative models exist due to the flexibility that co-operation strategies exhibit. One potential advantage of breaking a problem up into smaller dimensions and solving them co-operatively is the ability to utilize different algorithms when optimizing each sub-problem. This combines the various useful search characteristics of the different algorithms with the intent of harnessing their individual strengths to improve optimization. The heterogeneous co-operative algorithm (C_5-Hetero), introduced by Engelbrecht and Olorunda in [42], is one such algorithm which takes advantage of this property. C_5-Hetero incorporates each of the GA, *DE/rand/1/bin*, *DE/rand-to-best/1/bin*, bare-bones PSO and GCPSO algorithms. Solution fitness is evaluated via a context vector, similar to how it would be performed for CPSO-S and it's variants. The heterogeneous algorithm was shown to perform more consistently than several non-heterogeneous models [42].

2.4 Vector Evaluated PSO

As the original PSO algorithm is designed for problems with only one objective, it cannot be applied for multi-objective optimization without modification. An obvious albeit inefficient way to adapt the algorithm for multi-objective problems is to simply sum the objective functions together with no regard for how important each objective is. This technique was improved via assigning a multiplicative weighting to each

Algorithm 3 CPSO-H_k

Initialize an n-dimensional PSO swarm
Generate a random integer k **S.T** k mod $n = 0$, $0<k<n$
Initialize k CPSO-H_K swarms with $[n/k]$ dimensions each
while *current iteration* $<$ *max iterations* **do**
    for each swarm i **do**
        for each particle j in i **do**
            if f(i, j.$\vec{x}$) $>$ f(i, j.$\vec{y}$) **then**
                j.$\vec{y}$ = j.$\vec{x}$
            end if
            if f(i, j.$\vec{y}$) $>$ f(i, i.$\hat{y}$) **then**
                i.$\hat{y}$ = j.$\vec{y}$
            end if
            Do regular PSO position updates on j according to (2.4) and (2.5)
        end for
    end for
    Transfer knowledge from CPSO-H_K to PSO
    for each particle j in the n-dimensional PSO swarm **do**
        if f(j.$\vec{x}$) $>$ f(j.$\vec{y}$) **then**
            j.$\vec{y}$ = j.$\vec{x}$
        end if
        if f(j.$\vec{y}$) $>$ f($\hat{y}$) **then**
            $\hat{y}$ = j.$\vec{y}$
        end if
    end for
    for all particles in the n-dimensional PSO swarm **do**
        Calculate velocity update using eqn (2.4)
        Update particle position using eqn (2.5)
    end for
    Transfer knowledge from PSO to CPSO-H_K
end while

objective function before summation, formalized via [45, 44]. This essentially defines the overall fitness of a solution as a summation of all objectives, with certain objectives having higher priority than others if desired. The weighting assigned to each objective can either be static or dynamically updated during optimization. Both approaches were shown to be successful [45].

Another strategy was proposed by Coello and Lechuga [8] which did not utilize the traditional weighted summation approach. The proposed strategy, referred to as Multi-objective PSO (MOPSO), utilized the concept of Pareto dominance to establish the most desirable positions of the swarm. MOPSO splits the search space into hypercubes, with each having a fitness value inversely proportional to the number of particles within. The roulette wheel selection strategy is then used to select a hypercube. The leader of this hypercube is then used within the particle velocity update equation, which is

$$\vec{v} = \alpha\vec{v} + \omega C_1(\vec{y} - \vec{x}) + \lambda C_2(\vec{S} - \vec{x}) \tag{2.8}$$

where C_1 and C_2 are randomly generated numbers in the range [0,1], α is the inertial weight, ω is the cognitive weight, λ is the social weight, $\vec{v}$ is the particle velocity, $\vec{y}$ is the particle personal best position, $\vec{S}$ is the leader of the selected hypercube and $\vec{x}$ is the current particle position. At each iteration, the best position p_i is updated with respect to the domination relation between the previous best position and new best position of the particle. MOPSO uses a fixed-size repository to store all non-dominated solutions found during optimization. If the repository becomes full, solutions located in crowded areas of the objectice space are removed whenever a new solution is inserted. MOPSO was tested on various benchmark problems [8] and found to be an effective multi-objective optimization algorithm.

An alternative strategy for multi-objective PSO optimization was proposed by Parsopoulos and Vrahatis [45], referred to as Vector-Evaluated Particle Swarm Opti-

mization (VEPSO). VEPSO was originally inspired by the Vector-Evaluated Genetic Algorithm (VEGA) introduced in [48], however VEPSO has been shown to outperform VEGA [44]. The algorithm holds some similarities to MOPSO as it utilizes the concept of pareto domination and implements a structure to hold non-dominated solutions (similar to MOPSOs repository). One significant difference between the two algorithms is that particle fitness is now evaluated as a vector rather than a single scalar value, with each component of the vector corresponding to the fitness of an objective.

VEPSO utilizes multiple swarms where each swarm is assigned it's own objective to optimize. To ensure that the problem as a whole is optimized, information is passed between swarms via the use of a knowledge transfer strategy (KTS). A KTS defines how swarms will interact with each other, allowing each swarm to contain influence from other swarms. The VEPSO algorithm consists of four main phases:

1. Evaluation of fitness
2. Transfer of knowledge using the chosen KTS
3. Archive maintenance
4. Updating particle velocities

Phase one consists of calculating the fitness value of each individual particle with respect to the objective of the swarm containing that particle. The personal best and swarm global best are also updated in this phase. Phase two consists of applying the KTS to transfer information between swarms. Section 4.6.1. discusses various KTSs used to transfer knowledge in unique ways. The next phase performs maintenance of the current archive. Solutions which are non-dominated are added into the archive and solutions which become non-dominated as a result of these additions are removed. In the final phase, particle velocities are updated according to the specified velocity

update equation. The traditional PSO velocity update is commonly used[7], however other variants are also possible. These phases repeat iteratively until the max number of iterations is reached or some predefined stopping criteria is satisfied. Algorithm 4 describes the entire VEPSO algorithm. Note that $\vec{x}$ denotes current particle position and $\vec{y}$ denotes the personal best position of the current particle.

Algorithm 4 VEPSO

```
Initialize particles randomly
Initialize empty archive
while current iteration < max iterations do
    for each swarm i do
        for each particle j in i do
            if f(i, j.x⃗) > f(i, j.y⃗) then
                j.y⃗ = j.x⃗
            end if
        end for
    end for
    Utilize KTS to update global guide each swarm
    Perform archive maintenance
    for each swarm i do
        for each particle j in i do
            Calculate velocity of j using given update equation
            Update particle position of j according to eqn (2.5)
        end for
    end for
end while
```

2.4.1 Knowledge Transfer Strategies

Knowledge transfer strategies used within VEPSO play a crucial role in avoiding stagnation. Engelbrecht, Matthysen and Malan [40] show that the KTS must be carefully selected, as choosing strategies which are prone to stagnation degrade the performance of VEPSO considerably. The following are knowledge transfer strategies introduced in previous literature:

[7]However, the global best position used in the equation is commonly defined by the KTS.

- **Ring KTS** [45]- The global guide for a swarm is set to the global best particle from the neighbouring swarm. This is performed in a circular ring fashion.
- **Random KTS** [21] - The global guide for a swarm is set to the global best particle from a randomly selected swarm.
- **Random Personal Best KTS** [23] - The global guide for a swarm is set to a randomly selected personal best position from a randomly selected swarm
- **Roulette Wheel Personal Best KTS** [23] - The global guide for a swarm is set to a personal best position from a randomly selected swarm using roulette wheel selection. Selection probability is calculated as a proportion of the overall fitness.
- **Tournament Personal Best KTS** [23] - The global guide for a swarm is set to a personal best position from a randomly selected swarm using tournament selection. The tournament size is 10% of the selected swarm size.
- **Rank-based Personal Best KTS** [23] - The global guide for a swarm is set to a personal best position from a randomly selected swarm using rank-based selection. This consists of selecting a random number c within [1,n] where n is the number of particles. The best c particles are taken and from this group a random particle is selected.
- **PCX GBest KTS** [23] - The global guide for a swarm is set to the result of the Parent Centric Recombination [13] operator applied to the position of three archive solutions, selected randomly. The selected positions must be unique. If this criteria cannot be satisfied, randomly selected personal bests from randomly selected swarms are used to fill the remaining parent positions.
- **PCX GBest KTS** [23] - The global guide for a swarm is set to the result of the Parent Centric Recombination [13] operator applied to the global best position

of three swarms, selected randomly. The selected global best positions must be unique. If this criteria cannot be satisfied, a randomly selected personal best from a randomly selected swarm is used as the final parent.

2.5 Multi-objective Test Suites

Various suites exist for constructing benchmark functions suitable for multi-objective optimization. Outlined in [28], the general goals of any multi-objective test suite should be:

1. Unimodal test problems should be included. These are used to test convergence velocity relative to different Pareto optimal geometries and/or bias conditions.
2. The test suite must cover degenerate Pareto optimal fronts, disconnected Pareto optimal fronts and disconnected Pareto optimal sets.
3. Most test problems in the suite should be multimodal, however a few deceptive problems should also be present.
4. Problems should mainly nonseparable.
5. Problems which are both nonseparable and multimodal should be present to help simulate real-world problems.
6. Problems should be resistant to hill-climbing strategies.

Two very well known toolkits which aim to accomplish the above objectives are presented in Sections 2.5.1 and 2.5.2. These are the WFG and ZDT toolkits.

2.5.1 ZDT Toolkit

The ZDT Toolkit, originally presented by in [62], was originally created based on a pair of multi-objective optimization difficulties proposed by Deb [12]. The first

difficulty is the convergence to a Pareto-optimal front. This convergence is affected by multimodality, deception and isolated optima. If any of these phenomena are present in a problem, it is likely that the optimizer will have more difficulty converging to a front that is Pareto-optimal.

The second difficulty concerns itself with maintaining a diverse population. Diversity is important in achieving a well distributed non-dominated front [62]. However, certain properties of the Pareto-optimal front may restrict Pareto-optimal solution diversity. These are: discreteness, convexity and uniformness.

In total, there are 6 functions in the ZDT toolkit, commonly denoted ZDT1,ZDT2,...,ZDT6. Each of these functions include one of the difficulties presented by Deb [12], discussed above. These functions are restricted to two objectives only, as this is sufficent to accurately portray essential aspects of multi-objective optimization [62]. However, the number of decision space variables varies with the problem. The general format of these functions is shown below using three functions, f_1, g and h, where m represents the number of decision variables.

$$\text{Minimize } T(\vec{x}) = (f_1(x_1), f_2(\vec{x}))$$
$$\text{subject to } f_2(\vec{x}) = g(x_2, ..., x_m)h(f_1(x_1), g(x_2, ..., x_m))$$
$$\text{where } \vec{x} = (x_1, ..., x_m)$$

A complete definition of each individual ZDT function can be seen in the original work by Zitzler *et al.* [62].

2.5.2 WFG Toolkit

Huband *et al.* explore various limitations of the DLTZ toolkit in [29] with the intent of creating a new test suite devoid of these faults. The WFG toolkit is then proposed as a completely customizable toolkit, something which had not been done previously at that time. WFG's customizability stems from it's flexible problem attributes,

which are fully definable by the user. The problems themselves are defined in terms of a vector of parameters. This vector is derived through a series of transition vectors which each add complexity to the problem. Problem parameters are considered either distance-related or position-related. WFG problems are formulated as follows:

$$
\begin{aligned}
\text{Given } z =& \{z_1, ..., z_k, z_{k+1}, ..., z_n\} \\
\text{Minimize } f_{m=1:M}(x) =& x_M + S_m h_m(x_1, ..., x_{M-1}) \\
\text{where } x =& \{x_1 ... x_M\} = \{\max(t^p_M, A_1)(t^p_1 - 0.5) + 0.5, ..., \\
& \max(t^p_M, A_{M-1})(t^p_{M-1} - 0.5) + 0.5, t^p_M\} \\
t^p =& \{t^p_1, ..., t^p_M\} \leftarrow\!\shortmid t^{p-1} \leftarrow\!\shortmid ... \leftarrow\!\shortmid t^1 \leftarrow\!\shortmid z_{[0,1]} \\
z_{[0,1]} =& \{z_{1,[0,1]}, ..., z_{n,[0,1]}\} = \{z_1/z_{1,\max}, ..., z_n/z_{n,\max}\}
\end{aligned}
$$

where x is a set of M parameters with domain [0,1], M is the dimensionality of the objective space, z is a set of $k + l = n \geq M$ working parameters, $h_{1:M}$ are shape functions, $S_{1:M} > 0$ are scaling constants, $A_{1:M-1} \in \{0, 1\}$ are degeneracy constants and $t^{1:p}$ are transition vectors where "$\leftarrow\!\shortmid$" is used to indicate that the transition vectors are formed from another vector using a transformation function. The domain[8] of all $z_i \in$ z is [0, $z_{i,max}$] with all $z_{i,max} > 0$. Individual details of each WFG test function can be viewed in [28, 29].

The efficiency of the WFG toolkit is further explored in [28]. Huband *et al.* demonstrate WFG's effectiveness by showing the various problems present in existing toolkits which sre absent in the WFG toolkit. These shortcomings are listed below:

1. Deb's Toolkit

   (a) Limited to constructing two objective problems.

   (b) No suggested functions which facilitate problems with flat regions.

[8]Note that the lower bound is always zero for convenience.

(c) No real valued deceptive functions.

(d) Suggested functions don't facilitate problems with degenerate or mixed Pareto optimal front geometries.

(e) Only one function is nonseparable.

(f) Position and distance parameters are independent of each other, which is not representative of most real-world problems.

2. ZDT Toolkit

(a) Limited to constructing two objective problems.

(b) No suggested functions which facilitate problems with flat regions.

(c) It's only deceptive problem is binary encoded.

(d) Does not contain any non-separable problems.

(e) Only the number of distance parameters is scalable.

(f) All distance parameters are extremal except in ZDT4.

(g) No degenerate front problems.

3. DLTZ Toolkit

(a) No suggested functions which facilitate problems with flat regions.

(b) No deceptive problems.

(c) No nonseparable problems.

(d) The number of position parameters is always fixed with respect to the number of objectives.

Chapter 3

VEPSO using Co-operative PSO

This chapter discusses implementation of the VEPSO algorithm when co-operative PSO swarms are used instead of traditional PSO swarms. In Section 3.1, structural changes that arise as a result of these modifications are presented. In Section 3.2, an analysis of knowledge transfer strategy implications is presented. Section 3.3 and Section 3.4 discuss personal/global best updating and archive maintenance, respectively. Lastly, additional considerations which one should be mindful of when using co-operative principles in VEPSO are discussed in Section 3.5.

3.1 Structural Modifications

Since the PSO algorithm used for each swarm is independent of VEPSO, incorporating co-operative principles only requires modification of the individual swarm structures. Specifically, the n k-dimensional swarms within VEPSO are split into m subswarms, where m depends on the variant of CPSO being used and $m > n$. The m value for each CPSO variant is covered below:

- *CPSO-S* - since each k-d swarm will be reduced to k 1-d swarms, m will be equivalent to $k \cdot n$.

- *CPSO-S_K* - m can be calculated according to the equation below:

$$m = \sum_{i=0}^{n} \frac{k}{s_i} \tag{3.1}$$

  where s_i is the i'th random split value.

- *CPSO-H_K* - as hybridization is present, additional subswarms are required. m can be calculated according to the following:

$$m = \sum_{i=0}^{n} (\frac{k}{s_i}) + n \tag{3.2}$$

  where s_i is the random split value for the i'th CPSO-S_K swarm.

Figure 3.1 presents VEPSO with traditional PSO swarms while Figure 3.2 demonstrates VEPSO with CPSO-S swarms. These figures illustrate the structural differences present in VEPSO when co-operative principles are utilized within swarms. Observe that the total number of swarms increases in Figure 3.2 due to the dimensional split of CPSO-S.

3.2 Knowledge Transfer

The inclusion of a CPSO variant does not require any changes to the knowledge transfer strategy used. Knowledge flow between swarms (with respect to the chosen KTS) remains the same irrespective of whether PSO or a CPSO variant is used. However, some knowledge transfer strategies[1] require a "global best" particle to transfer knowledge, which CPSO variants do not maintain. To overcome this, these variants instead utilize their context vector for knowledge transfer purposes when required.

Figure 3.3 exemplifies knowledge flow of the random KTS via utilization of the context vector. To aid in visualization, consider the following scenario: Suppose

[1]Within this thesis, these are the ring, random and PCX GBest strategies

during the knowledge transfer phase, swarm S^2 transfers knowledge to swarm S^1. Next, during the update phase, particle x of subswarm S_1^1 is about to be updated. Particle x will then utilize the context vector of swarm S^2 (in place of the global best particle) within it's update equation. By extension, the global best particle of S_1^2 is used since the context vector is made up of individual subswarm bests.

3.3 Personal/Global Best Update

Criteria for updating the personal/global bests do not change when co-operative swarm variants are used within VEPSO. This is because CPSO-S, CPSO-S_K and CPSO-H_K update personal and global bests in the same manner as traditional PSO. Personal and global bests are updated based on fitness values in best algorithms, thus particles with the highest fitness scores are regarded as best. One should note that it is also possible to update personal/global bests based on non-domination criteria, however this is not utilized within this thesis.

3.4 Archive Maintenance

To account for the subswarm split of CPSO variants, missing dimensions of particles are replaced with context vector dimensions when being evaluated for archive addition. If a particle then results in a newly non-dominated vector, the entire vector (including context vector dimensions) is added into the archive.

3.5 Additional Considerations

Using co-operative PSO swarms within VEPSO presents some considerations which one should be mindful of when performing comparisons. CPSO increases the number of function evaluations per iteration due to the swarm split, which is not present in

traditional PSO. To account for this, one should lower the number of particles used by each CPSO swarm. For experimentation purposes of VEPSO, the total number of particles used for CPSO swarms should be equal to the number of particles used in PSO swarms.

When using CPSO within VEPSO swarms, one should also ensure that a variety of diverse functions are used. Since CPSO is a step-wise, methodical approach, it tends to perform exceptionally well for functions with simple shapes and modalities in comparison to other algorithms. The inclusion of function properties such as multimodality, deception, disconnection and degeneracy allow for a much better real-world environment when testing VEPSO with CPSO swarms.

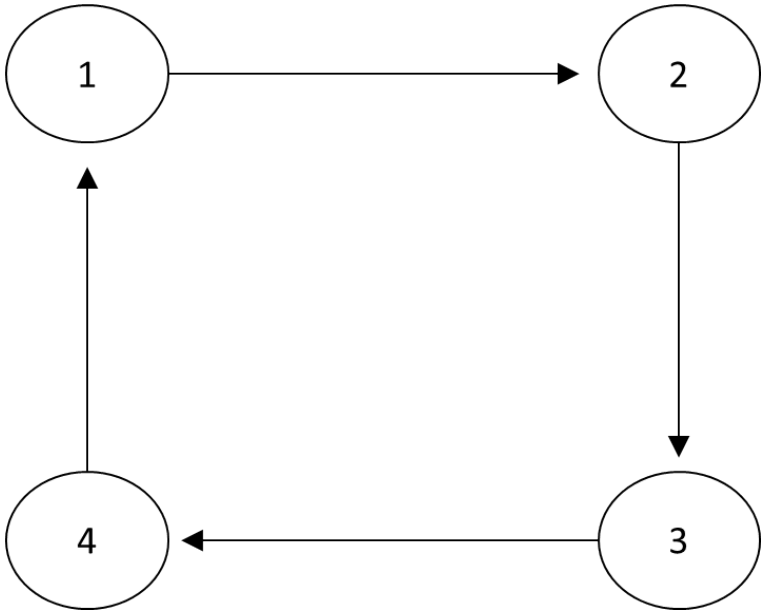

Figure 3.1: Illustration depicting the structural representation of VEPSO when traditional PSO swarms are used for a problem with 4 objectives. Knowledge flow is shown according to the ring KTS.

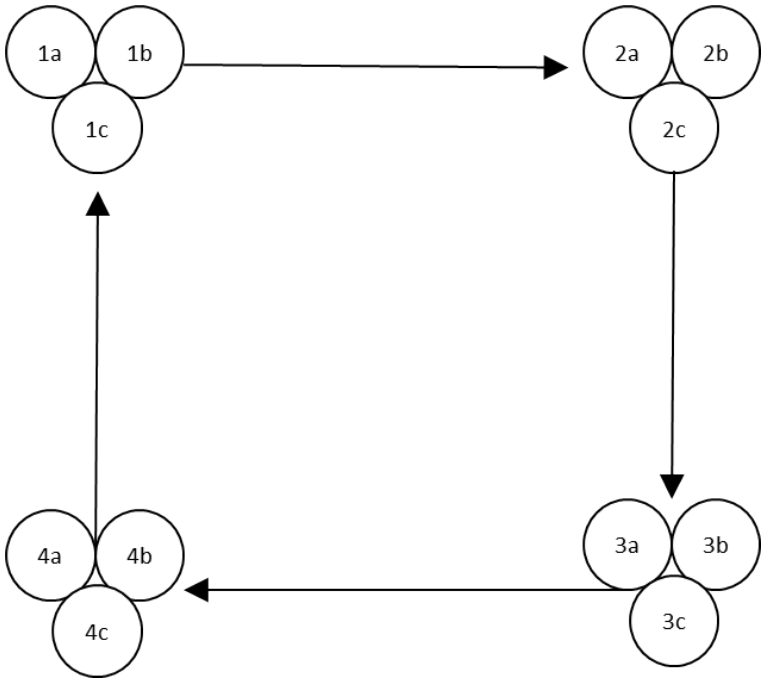

Figure 3.2: Illustration depicting the structural representation of VEPSO when co-operative PSO swarms are used for a problem with 4 objectives. The number of subswarms per swarm are 3. Knowledge flow is shown according to the ring KTS.

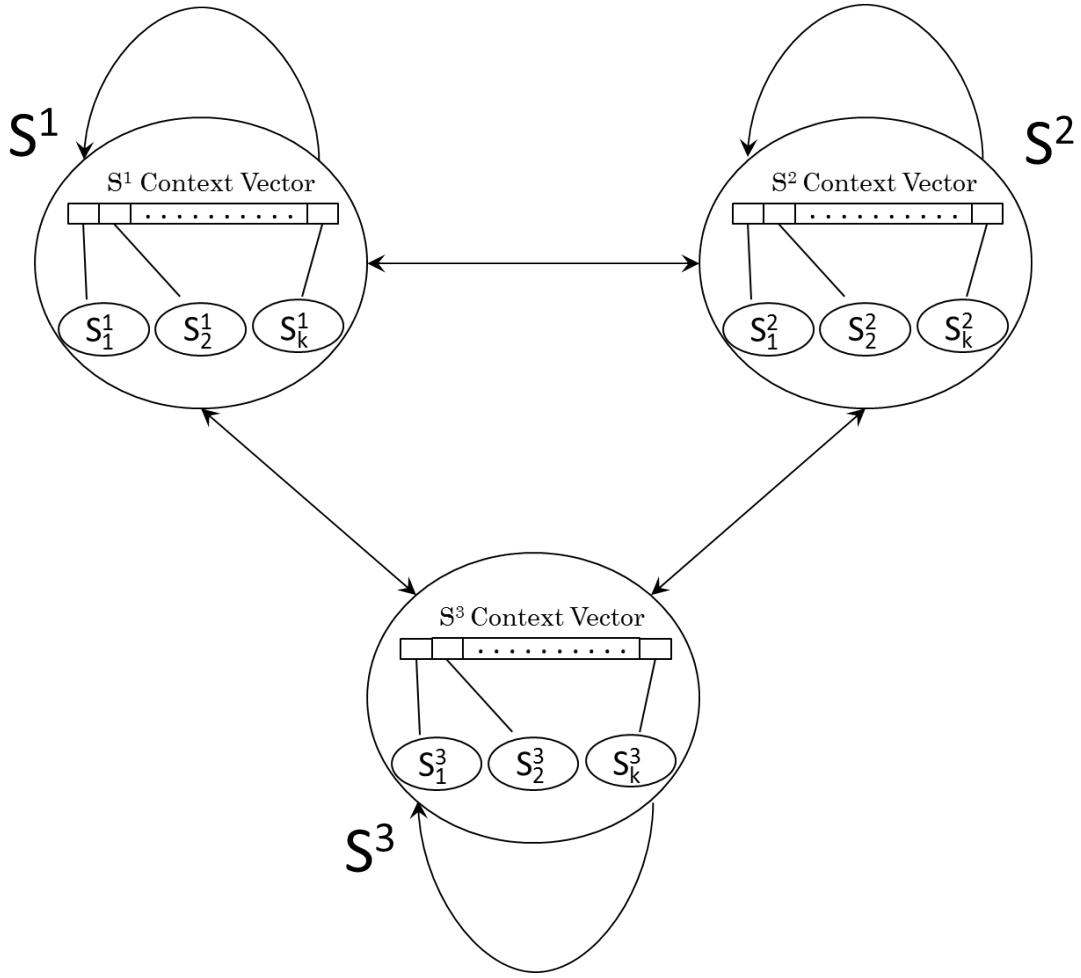

Figure 3.3: Visualization of the random KTS knowledge transfer flow using context vectors. The problem at hand contains a 3-dimensional objective space.

Chapter 4

Experimental Setup

This chapter details the methods and components used in performing experimentation. Section 4.1 presents the various performance measures used for algorithm comparison. In Section 4.2, an overview of the statistical analysis methods are covered. Parameters of PSO and CPSO are detailed in Section 4.3 and 4.4 respectively while VEPSO parameters are shown in Section 4.5. Lastly, Section 4.6 explains the various benchmark functions chosen.

Experiments were performed via an altered version of the Computational Intelligence Library [43]. Several elements from the jMetal framework [15] were also incorporated as well. All pseudo-random number generation was performed via an implementation of the Mersenne Twister [39].

4.1 Performance Measures

In real-world multi-objective optimization knowing the Pareto optimal front is extremely rare. This make measuring the performance of multi-objective optimizers a non-trivial task. Many performance measures tend to analyze the obtained approximation front using various criteria. These criteria can be categorized as follows:

- *Distribution* - the overall distribution of the non-dominated solution set

- *Variance* - non-dominated solution spread with respect to the minimum/maximum objective values
- *Diversity* - amount of different non-dominated solutions
- *Precision* - similarity of the approximation front to a best known estimation

The performance measures used in this work aim to incorporate a blend of these criteria without actually knowing the true Pareto optimal front.

4.1.1 Hypervolume Indicator

The hypervolume indicator, introduced in [60], is a metric which measures both the spread of solutions along the Pareto front in addition to the closeness of solutions to the Pareto-optimal front. Hypervolume is represented as a single scalar value, calculated as the sum of sub-cuboids created by solution points. Since hypervolume calculation is an NP-hard problem [2], it takes exponential time with respect to the number of objectives. This is a serious drawback, as it may be infeasible to use the hypervolume metric when the objective space dimensionality is large.

One extremely important aspect of the Hypervolume metric is that it is maximized if, and only if, the solution set consists entirely of all Pareto-optimal points [18]. Therefore, hypervolume can be used to indicate how well a multi-objective optimization algorithm is at producing the true Pareto-optimal front. In addition to this, hypervolume is capable of proving that a solution set is not worse than some other solution set for all solution pairs [61].

An overview of various hypervolume calculation algorithms can be found in [1]. The implementation used in this study is the LebMeasure algorithm [19]. This algorithm removes rectangular polytopes which do not intersect with any other region concerning the space covered by non-dominated points. The hypervolumes of these polytopes are recorded and new points are spawned upon removing each region. This

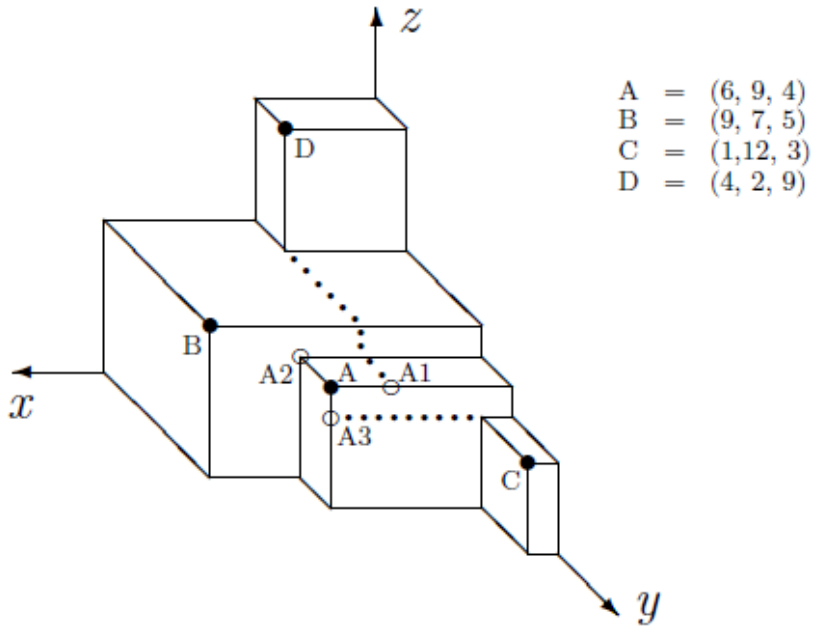

Figure 4.1: Spawning in LebMeasure. The blocks denote the hypervolume dominated by A, B, C, D in a maximisation problem relative to the origin. The filled circles represent the four points, and the empty circles represent the three potential spawns of A. Each spawn is generated by reducing one objective to the largest smaller value from the other points. It is clear that A2 is dominated by B, so A is replaced by A1 and A3. Image and caption borrowed from [57].

process repeats iteratively until the polytope which remain do not dominate any region of space. The total hypervolume is then returned as the accumulated hypervolume from the recorded polytopes. Figure 4.1 presents a visualization of hypervolume and the LebMeasure algorithm.

4.1.2 Solution Distribution

The solution distribution metric [20] is used to measure the distribution and spacing density of the produced non-dominated solution set. This is calculated via the following equation:

$$S = \frac{1}{n}\sqrt{\frac{1}{n}\sum_{i=1}^{n}(d_i - \vec{d})^2},\ \vec{d} = \frac{1}{n}\sum_{i=1}^{n} \tag{4.1}$$

In the equation above, n represents the number of non-dominated solutions and d_i is the Euclidean distance between solution i and it's closest objective-space neighbour. Minimization of this metric corresponds to a more desirable distribution of solutions.

4.2 Statistical Methods

Statistical methods are divided into two main categories according to what is known about the population at hand. The popular *parametric* category consists of tests which make assumptions about the underlying distribution, often assuming that the distribution is normal. Conversely, *non-parametric* tests do not rely on the assumption that data has been drawn from a given distribution. Thus, these methods are considered to be *distribution free* since they have no dependence on the given population. Since normal variances cannot be assumed[1] for a stochastic optimizer [34], the methods used in this work are non-parametric. This helps to ensure that type I and II errors are reduced as much as possible since the data is not guaranteed to be normally distributed [11].

The non-parametric version of the classic t-test, known as the Mann-Whitney-Wilcoxon rank sum test [38], was used in a pairwise fashion to test for statistical significance in all experiments. If a statistically significant difference between a pair of algorithms existed, the algorithm with the higher mean was given a point and the algorithm with the lower mean was deducted a point. Each Mann-Whitney-Wilcoxon rank sum test was performed with a 95% confidence level. All experiments were recorded using 35 runs.

[1]However, it is still possible for a stochastic optimizer to possess a normal distribution

4.2.1 Mann-Whitney-Wilcoxon Rank Sum Test

The Mann-Whitney-Wilcoxon rank sum test [38], also known as the Mann-Whitney U-test, is used to determine whether two independent samples of observations are drawn from identical distributions. The test is built around the idea that arranging random variables together in increasing order of magnitude will help draw conclusions about the relationship between the parent populations. If sufficient evidence of random mixing is not present, the null hypothesis of identical distribution is rejected. The Mann-Whitney-Wilcoxon rank sum test makes a few assumptions beforehand:

- The two samples are random.
- The two samples are independent of each other.
- Observations are arranged by rank.

To calculate this test. first find the individual sum of the ranks R_1 and R_2. Next, determine the U statistic for each sample using the following equations:

$$U_1 = n_1 n_2 + \frac{n_1(n_1+1)}{2} - R_1 \tag{4.2a}$$

$$U_2 = n_2 n_1 + \frac{n_2(n_2+1)}{2} - R_2 \tag{4.2b}$$

$$U = min\{U_1, U_2\} \tag{4.2c}$$

where n_1 and n_2 represent the sizes of sample 1 and 2, respectively. A z score can then be calculated as:

$$\mu = \frac{n_1 n_2}{2} \tag{4.3a}$$

$$\sigma = \sqrt{\frac{n_1 n_2 (n_1 + n_2 + 10)}{12}} \tag{4.3b}$$

$$z = \frac{U - \mu}{\sigma} \tag{4.3c}$$

The z score calculated above is then compared with a critical value associated with a pre-determined confidence level, α. The null hypothesis which states that the two populations have identical distribution functions is then either rejected or accepted based on this comparison. One should note that the Mann-Whitney-Wilcoxon rank sum test is the non-parametric equivalent of the t-test.

4.3 PSO Parameters

Each PSO swarm is created with 100 particles. The initial velocity of these particles are set to zero. Concerning the weight values, α is set to 0.729844, ω is set to 1.496180 and λ is set to 1.496180. Note that these values are not set arbitrarily; they are chosen due to the fact that they ensure convergence [56]. Boundary clamping is used to ensure that particles do not move into illegal positions with respect to the given problem. Updating is done asynchronously as described in [4].

4.4 CPSO Parameters

To ensure an unbiased comparison between traditional PSO and the CPSO-S/ CPSO-S_K variants, each swarm is assigned the same number of particles (100) as traditional PSO before performing splitting into sub-swarms. When the sub-swarm split is performed based on the CPSO variant, these particles are divided as evenly as possible

Table 4.1: Overview of Functions Used

Function	Position Parameters	Distance Parameters	Shape	Modality
1	20	4	Convex	Uni
2	20	4	Convex, Disconnected	Uni, Multi
3	20	4	Linear, Degenerate	Uni
4	20	4	Concave	Multi
5	20	4	Concave	Deceptive
6	20	4	Concave	Uni
7	20	4	Concave	Uni
8	20	4	Concave	Uni
9	20	4	Concave	Multi, Deceptive

across all sub-swarms. The CPSO-H_K variant is also used with 100 total particles, allocating 50 particles to it's CPSO-S_K algorithm and 50 particles to it's traditional PSO implementation. Since the total number of particles used for all CPSO variants and traditional PSO is equal, the number of fitness function evaluations per iteration will be identical for each algorithm. The weight values, initial velocity values, boundary clamping techniques and aysnchronous update methodologies are identical to those of traditional PSO, described in Section 4.3.

4.5 VEPSO Parameters

The number of swarms used in the VEPSO algorithm was equal to the number of objectives for the problem at hand. The archive was restricted to 250 solutions, with solutions being removed when this size limit was reached. Selection of the solution to remove was chosen randomly.

4.6 Benchmark Functions

Various benchmark functions taken from the WFG toolkit are used for experimentation purposes in this work. These functions, selected due to their wide array of shapes and modalities, ensure a diverse and complete testing environment. They incorporate

unique difficulties to help simulate unpredictable real-world problems. An overview of the WFG toolkit and it's parameters can be found in Section 2.5.2. Each function uses the exact same parameter set, consisting of 3 objectives and a total of 24 decision parameters. The decision parameters contain 20 position-related parameters and 4 distance-related parameters. A summary of all functions used along with their properties can be observed in Table 4.1.

Chapter 5

Experimental Results and Discussion

This chapter presents and discusses results obtained from the experiments performed. The experiments were split into three main categories. The first category, covered in Section 5.1, consists of experiments measuring algorithm performance. The second category, presented in Section 5.2, contains experiments designed to test the performance of all knowledge transfer strategies for each variant of CPSO. KTS performance was not tested for VEPSO with traditional PSO swarms as previous work on it by Harrison et al. [23] exists. Section 5.3 discusses experiments which compare the performance of the two best co-operative algorithm-KTS combinations against other well known multi-objective algorithms.

Within this chapter, it is assumed that references to each algorithm are within the context of using them in VEPSO swarms. All experiments reference *difference*, which is simply the difference between pairwise wins and losses (described in Section 4.2) for each algorithm. In addition to this, the *rank* of each is shown which denotes the algorithm placing in comparison to all other algorithms with respect to WFG function performance. In the event of an identical placing, algorithms were compared in a head-to-head fashion over all 6 functions from the ZDT toolkit. If an algorithm performed better, it was deemed the winner and thus was given the higher placing.

If the tie could not be resolved, algorithms were given identical placings.

5.1 Algorithm Performance

This section compares the performance of each algorithm over various WFG functions. The objective of this comparison is to establish which algorithms, if any, performed better over each of the functions. To ensure unbiased comparisons, all algorithms were tested using each of the 4 different knowledge transfer strategies. A results summary is presented in Table 5.5 which displays the highest ranking algorithm for each group. If the highest ranking algorithms were tied, the algorithm with the highest placing for the metric not being considered was shown.

5.1.1 Ring KTS

Performance of PSO, CPSO, CPSO-H_K and CPSO-S_K swarms were compared over the nine WFG functions using the ring KTS. Table 5.1 presents the results of these experiments for both the hypervolume and distribution metric. Examining this table for each WFG function presents some interesting results. One observation is that PSO yielded extremely poor hypervolume performance in comparison to the other algorithms, as PSO recorded one or more losses over every single function. Thus, it can be deduced that at least one co-operative variant produces higher hypervolume over PSO for every single WFG function when the ring KTS is used. Although PSO also distributed poorly as well, the previous conclusion does not apply for the distribution metric as no losses occured for WFG1 and WFG3.

Concerning the performance of the CPSO variants, CPSO-S_K outperformed CPSO-H_K for the hypervolume metric. However, both algorithms placed first overall very rarely, suggesting that neither variant has hypervolume performance advantages over CPSO-S for the ring strategy. This observation also holds for the distribution metric.

CPSO-H_K and CPSO-S_K obtained similar distribution placings on average, performing slightly worse than PSO and considerably worse than CPSO-S. CPSO-S was undoubtedly the top performing algorithm overall, placing first for 8 of 9 functions with respect to both metrics.

5.1.2 Random KTS

The experiments in Section 5.1.1 were repeated using the random KTS, yielding a similar set of results along with a few additional observations. These results are displayed in Table 5.2. PSO again produced subpar performance in comparison to the CPSO variants, with respect to hypervolume. However, it distributed solutions more consistently than CPSO-H_K and CPSO-S_K.

An interesting observation present in Table 5.5 is that CPSO-s again placed first for every function except WFG1. Performance degradation was severe for this function as the algorithm did not earn a single win for both metrics. It is observed that WFG1 is the only function with a strictly convex shape, suggesting that CPSO-S has severe difficulty when presented with this landscape.

The CPSO-S_K algorithm placed higher than CPSO-H_K and PSO on average, ranking first overall for 3 out of 9 functions and second for the remaining functions. The algorithm did not perform as well with regards to the distribution metric, earning losses from all other algorithms for WFG4 and WFG6. Both CPSO-H_K and CPSO-S_K yielded the lowest distribution placings on average. It is interesting to note that CPSO-H_K performed worse than the other co-operative algorithms for the deceptive WFG5 and WFG9 functions. This suggests that the CPSO-H_K may offer no performance benefits over CPSO-S and CPSO-S_K in multi-objective environments, since it fails to accomplish it's sole purpose of improving performance in deceptive environments.

5.1.3 PCX GBest KTS

Table 5.3 contains results for the algorithms using the PCX GBest transfer strategy.Observations drawn from the results of these experiments are nearly identical to those of the ring and random strategies. In terms of both hypervolume, and distribution, CPSO-S was dominant over all other algorithms. The algorithm consistently placed first, struggling only with WFG1. In terms of hypervolume, CPSO-S_K earned the second highest placings. However, CPSO-S_K was able to outperform CPSO-S for WFG1, WFG2 and WFG7. It can thus be asserted that CPSO-S_K does increase the performance of CPSO-S in some scenarios. The only drawback of CPSO-S_K is that distribution tends to be very poor in general, often much worse than the other algorithms.

CPSO-H_K and PSO both yielded the lowest hypervolume metrics on average, with CPSO-H_K earning slightly more wins over PSO. PSO struggled severely, earning only a single win on the WFG9 function. However, PSO distributed it's solutions better than CPSO-S_K and CPSO-H_K. An intriguing observation is that CPSO-H_K earned its highest placings on the convex WFG1 function, outperforming both CPSO-S and CPSO-S_K.

5.1.4 PCX Archive KTS

Performance while using the PCX Archive KTS was tested for each algorithm, detailed in Table 5.4. One noticeable difference from previous results can be observed via the CPSO-H_K distribution performances. The algorithm yielded a very high solution distribution for all functions, placing first and second consistently. From this, it can be confidently stated that CPSO-H_K works very well with the PCXArchive strategy. It is hypothesized that this can be attributed to the solution injection mechanism of the CPSO-H_K algorithm and the transfer mechanism of PCXArchive. As time passes, half of the particles of the swarm become filled with global best particles

and thus the archive contains many solutions based around slight variations of these global bests. These particles are selected as swarm leaders due to the knowledge transfer mechanism of PCXArchive, which increases explorative potential and leads to a desirable solution distribution.

Table 5.4 also reveals that PSO again experienced subpar hypervolume performance in comparison to the other algorithms. The algorithm had a negative difference score for all functions, consistently yielding more losses than wins. Since this observation was present for all other strategies thus far, it is concluded that CPSO and it's variants yield greater hypervolume performance over PSO in general, independent of the KTS used. However, as CPSO-S_K and CPSO-H_K experienced equal or worse distribution in comparison to PSO, the utilization of these algorithms is seen as more of a trade-off than a strict improvement. This does not apply for CPSO-S, as it's consistent domination of both metrics over PSO allows one to conclude that it is a definite improvement over traditional PSO.

5.2 KTS Performance

This section measures the performance of each knowledge transfer strategy for each co-operative PSO swarm variant.This was done to observe the strengths/weaknesses of each KTS when co-operative PSO swarm variants are used. A results summary is presented in Table 5.9 which displays the highest ranking strategy for each group. If the highest ranking algorithms were tied, the algorithm with the highest placing for the metric not being considered was shown.

5.2.1 CPSO-S

Performance using various knowledge transfer strategies is observed for CPSO-S. A summary of the results obtained is shown in Table 5.6. Concerning the non-hybrid

strategies, the random strategy was outperformed by the ring strategy for the hypervolume metric, earning more losses than wins on every function other than WFG9. However, the random strategy distributed solutions better than the ring strategy on average. It is theorized that the randomness produced adds enhanced explorability, thus improving the distribution. Therefore in the context of CPSO-S, the random KTS should be utilized when distribution is a priority and the ring KTS should be used when hypervolume is a priority.

The hybrid strategies experienced better performance than the non-hybrid strategies for both the hypervolume and distribution metric. It is worth noting that a greater disparity was seen for the former metric rather than the latter. The PCXGBest strategy yielded incredibly well distribution metric results, placing first for all functions except WFG9 and recording only 2 losses overall. Although PCXArchive had the second best distribution performance overall, it was still severely outclassed by PCXGBest with regards to the metric. Hypervolume between the two strategies was very comparable as both experienced similar results (which were still much higher than the non-hybrid strategies). Due to the superior distribution and non-existant hypervolume degradation[1], it can be confidently stated that the PCX GBest strategy was the overall best performing KTS for the CPSO-S algorithm.

5.2.2 CPSO-S_K

KTS performance is experimented on for the CPSO-S_K algorithm, presented in Table 5.7. The same conclusions as Section 5.2.1 can be formulated with respect to the non-hybrid strategies. With respect to CPSO-S_K, the random KTS should be when one desires higher distribution and the ring KTS should be used when one desires higher hypervolume. This is due to the ring KTS experiencing better hypervolume performance in general over the random strategy, ranking higher for all functions

[1] In comparison to the PCXArchive strategy.

except WFG6 and WFG9. The random KTS outperformed the ring strategy with regards to the distribution metric, further supporting the drawn conclusion.

The hybrid strategies presented a different set of observations than was previously found. WFG functions 2 to 7 possessed the highest hypervolume when the PCXGBest strategy was used, while WFG functions 2,4,5,6,7 and 8 had the highest distribution metric when the PCXArchive strategy was used. Therefore, PCXGBest was dominant for hypervolume while PCXArchive dominated the solution distribution metric over the WFG functions. It is worth noting that the observations found in this section are nearly identical to those for PSO found in [23]. The only differences between the two are:

1. The random KTS yields better distribution than the ring for CPSO-S_K and not for PSO.

2. The author of [23] noted for PSO that deceptive problems "respond well to the ring strategy when hypervolume is being measured as the ring strategy seems to be more explicitly optimizing single objectives than the random-based strategies". This phenomenon did not occur for CPSO-S_K, as the ring strategy did not respond more favourably to deceptive problems[2] in comparison to the random strategy.

Therefore in regards to KTS performance, PSO and CPSO-S_K yield a similar set of conclusions with only minor differences as presented above.

5.2.3 CPSO-H_K

In Table 5.8, KTS results for CPSO-H_K are displayed. The first observation present is that the ring strategy and random strategy performed much more similarly than was previously seen for the CPSO-S and CPSO-S_K. The only difference between the two

[2]This references the WFG5 and WFG9 functions.

was that the ring strategy recorded slightly higher hypervolume placings on average. Distribution metric was relatively equal, with both algorithms experiencing a similar amount of wins and losses. Since CPSO-H_K consists of both PSO and CPSO-S_K, it is likely that the non-hybrid observations formed are a combination of the individual observations seen in both of these algorithms. PSO was found to have superior performance in terms of both metrics for the ring strategy in [23], while CPSO-S_K favored the ring strategy for hypervolume and the random strategy for distribution. Thus it is likely that since the ring strategy was superior in terms of hypervolume for both algorithms, it is also superior for CPSO-H_K. This may also explain the equal distribution, as no consensus could be reached concerning which KTS performed best for both PSO and CPSO-S_K.

Table 5.8 presents another intriguing observation. The ring strategy completely dominated the random strategy for WFG2, WFG5 and WFG9, obtaining a higher ranking for both hypervolume and distribution. WFG2 possess a disconnected landscape, WFG5 is deceptive and WFG9 is both deceptive and contains a non-separable reduction. It is highly probable that the superiority of the ring strategy over the random strategy for these functions is due to the presence of PSO within CPSO-H_K, as PSO also yields these observations [23].

Hybrid strategy observations were identical to those of CPSO-H_K and PSO. The PCX GBest strategy performed best with respect to the hypervolume metric while the PCX Archive strategy had a superior distribution of solutions in comparison to all other tested strategies.

5.3 Comparison to other Algorithms

This section compares the performance of the best performing co-operative algorithm-KTS combinations against two well known multi-objective optimization algorithms,

oMOPSO [10] and SMPSO [41]. As it has been concluded in Section 5.1 that CPSO-S and it's variants improve VEPSO performance, comparisons[3] must now be made against other algorithms which are known to perform well with the intent of observing whether co-operation allows the VEPSO algorithm to compete. The two highest performing algorithm-KTS combinations were CPSO-S_K PCXGBest and CPSO-S PCXGBest.

Previous work [22] indicates that VEPSO with traditional PSO is outperformed by both SMPSO and oMOPSO on nearly every WFG function for both the distribution and hypervolume metric in 3-D objective space. The PCX GBest strategy was unable to earn a single first place finish for both metrics. It is also shown in [22] that VEPSO typically scales poorly, with no KTS able to outperform oMOPSO or SMPSO on both metrics when more than 3 objectives were used. To help strengthen conclusions, performance is compared in both 3-D and 5-D objective space within this section. A results summary is presented in Table 5.12 which displays the highest ranking algorithm for each group. If the highest ranking algorithms were tied, the algorithm with the highest placing for the metric not being considered was shown.

5.3.1 Performance in 3-D Objective Space

Table 5.10 presents the results for each algorithm when the number of objectives is equal to three. Both oMOPSO and SMPSO exhibited better hypervolume performance on average over the co-operative algorithms. However, CPSO-S performed best for WFG4, WFG5 and WFG6, indicating that the algorithm does still outperform both oMOPSO and SMPSO in certain scenarios. It is interesting to note that these functions together contain all modalities present within the WFG toolkit, which are unimodal, multimodal and deceptive. This suggests that CPSO-S has the potential to outperform these algorithms over all types of function modalities.

[3] An equal amount of function iterations are used across all algorithms.

CPSO-S_K experienced the worst performance overall with respect to both metrics. It's distribution performance was especially bad, placing last for every function other than WFG1. This is not surprising, as CPSO-S_K exhibited poor distribution placings for nearly every KTS in Section 5.1. oMOPSO performed very well, yielding the highest hypervolume values followed by SMPSO. However, with regards to the distribution metric, both oMOPSO and SMPSO were outperformed by CPSO-S for 5 functions. CPSO-S still distributed slightly worse than oMOPSO on average but better than SMPSO.

These observations lead to the conclusion that CPSO-S can undoubtedly compete with both oMOPSO and SMPSO in 3-D objective space, especially with regards to the distribution metric in which it often outperforms both algorithms. This conclusion is not valid for CPSO-S_K, as it is completely dominated by both algorithms.

5.3.2 Performance in 5-D Objective Space

In Table 5.10, the performance of each algorithm is shown for 5-dimensional objective space. This table presents an intriguing set of observations. First, the hypervolume performance of CPSO-S_K is greatly improved from Section 5.3.1, when only three objectives were used. This suggests that CPSO-S_K scales very well with respect to the hypervolume metric. Both CPSO-S_K and CPSO-S yielded better scalability than SMPSO and oMOPSO. oMOPSO scaled worst, placing in the top half for only 3 out of 9 functions for hypervolume. SMPSO scaled slightly better than MOPSO, however both algorithms performed nearly evenly in terms of distribution.

CPSO-S_K's dominant hypervolume performance was offset by it's poor distribution placings. This indicates that the utilization of CPSO-S_K can still be seen as a tradeoff. Situations in which one prioritizes hypervolume over solution distribution would be the most efficient use of CPSO-S_K. However, further research as to whether the solution distribution of CPSO-S_K improves when the number of objectives is

extremely large is needed.

CPSO-S scaled very well in comparison to the non co-operative algorithms. One should be aware that it's fairly large amount of second placings are due to the increased hypervolume performance of CPSO-S_K. It was only outperformed by oMOPSO and SMPSO for WFG1, WFG2 and WFG3. This is likely attributable due to the performance degradation of CPSO-S in the presence of complex function shapes, as WFG1-3 are the only functions which are not strictly concave. Concerning solution distribution, CPSO-S exhibited the highest performance on average. The algorithm earned more wins than losses on 7 out of 9 functions.

Table 5.1: Mann-Whitney Wins and Losses For Ring KTS

			WFG Function								
Algorithm	**Metric**	**Result**	1	2	3	4	5	6	7	8	9
PSO	Hypervolume	Wins	0	0	0	0	0	0	0	0	1
		Losses	3	3	3	3	2	3	3	3	1
		Difference	-3	-3	-3	-3	-2	-3	-3	-3	0
		Rank	4	4	4	4	3	4	4	4	2
	Distribution	Wins	3	1	2	1	2	1	0	2	0
		Losses	0	1	0	1	1	1	1	1	1
		Difference	+3	0	+2	0	+1	0	-1	+1	-1
		Rank	1	2	1	2	2	2	2	2	3
CPSO-S	Hypervolume	Wins	1	2	3	3	3	3	3	3	3
		Losses	2	0	0	0	0	0	0	0	0
		Difference	-1	+2	+3	+3	+3	+3	+3	+3	+3
		Rank	3	1	1	1	1	1	1	1	1
	Distribution	Wins	0	3	2	3	3	3	3	3	2
		Losses	3	0	0	0	0	0	0	0	0
		Difference	-3	+3	+2	+3	+3	+3	+3	+3	+2
		Rank	4	1	1	1	1	1	1	1	1
CPSO-S_K	Hypervolume	Wins	2	2	2	2	2	2	2	2	1
		Losses	0	0	1	1	1	1	1	1	1
		Difference	+2	+2	+1	+1	+1	+1	+1	+1	0
		Rank	1	1	2	2	2	2	2	2	2
	Distribution	Wins	1	1	0	0	1	0	0	0	0
		Losses	2	1	3	3	2	3	1	2	2
		Difference	-1	0	-3	-3	-1	-3	-1	-2	-2
		Rank	3	2	4	4	3	4	2	3	4
CPSO-H_K	Hypervolume	Wins	2	1	1	1	0	1	1	1	0
		Losses	0	2	2	2	2	2	2	2	3
		Difference	+2	-1	-1	-1	-2	-1	-1	-1	-3
		Rank	1	3	3	3	3	3	3	3	4
	Distribution	Wins	2	0	1	1	0	1	0	0	1
		Losses	1	3	2	1	3	1	1	2	0
		Difference	+1	-3	-1	0	-3	0	-1	-2	+1
		Rank	2	4	3	2	4	2	2	3	2

Table 5.2: Mann-Whitney Wins and Losses For Random KTS

Algorithm	**Metric**	**Result**	**WFG Function** 1	2	3	4	5	6	7	8	9
PSO	Hypervolume	Wins	1	0	0	0	0	0	0	0	1
		Losses	2	3	3	3	3	3	3	2	2
		Difference	-1	-3	-3	-3	-3	-3	-3	-2	-1
		Rank	3	4	4	4	4	4	4	3	3
	Distribution	Wins	1	1	2	1	1	1	0	2	0
		Losses	1	2	1	1	1	1	1	1	0
		Difference	0	-1	+1	0	0	0	-1	+1	0
		Rank	3	3	2	2	2	2	2	2	2
CPSO-S	Hypervolume	Wins	0	2	3	3	3	3	2	3	3
		Losses	3	0	0	0	0	0	0	0	0
		Difference	-3	+2	+3	+3	+3	+3	+2	+3	+3
		Rank	4	1	1	1	1	1	1	1	1
	Distribution	Wins	0	3	3	3	3	3	3	3	2
		Losses	3	0	0	0	0	0	0	0	0
		Difference	-3	+3	+3	+3	+3	+3	+3	+3	+2
		Rank	4	1	1	1	1	1	1	1	1
CPSO-S_K	Hypervolume	Wins	3	2	2	2	2	2	2	2	2
		Losses	0	0	1	1	1	1	0	1	1
		Difference	+3	+2	+1	+1	+1	+1	+2	+1	+1
		Rank	1	1	2	2	2	2	1	2	2
	Distribution	Wins	1	2	0	0	1	0	0	0	0
		Losses	0	1	2	3	1	3	1	2	1
		Difference	+1	+1	-2	-3	0	-3	-1	-2	-1
		Rank	2	2	3	4	2	4	2	3	3
CPSO-H_K	Hypervolume	Wins	2	1	1	1	1	1	1	0	0
		Losses	1	2	2	2	2	2	2	2	3
		Difference	+1	-1	-1	-1	-1	-1	-1	-2	-3
		Rank	2	3	3	3	3	3	3	3	4
	Distribution	Wins	2	0	0	1	0	1	0	0	0
		Losses	0	3	2	1	3	1	1	2	1
		Difference	+2	-3	-2	0	-3	0	-1	-2	-1
		Rank	1	4	3	2	4	2	2	3	3

Table 5.3: Mann-Whitney Wins and Losses For PCXGBest KTS

			WFG Function								
Algorithm	**Metric**	**Result**	1	2	3	4	5	6	7	8	9
PSO	Hypervolume	Wins	0	0	0	0	0	0	0	0	1
		Losses	2	3	3	3	3	3	3	3	2
		Difference	-2	-3	-3	-3	-3	-3	-3	-3	-1
		Rank	3	4	4	4	4	4	4	4	3
	Distribution	Wins	2	1	2	0	2	0	1	2	0
		Losses	0	2	1	1	1	1	1	1	0
		Difference	+2	-1	+1	-1	+1	-1	0	+1	0
		Rank	1	3	2	3	2	2	2	2	2
CPSO-S	Hypervolume	Wins	0	2	3	3	3	3	2	3	3
		Losses	2	1	0	0	0	0	0	0	0
		Difference	-2	+1	+3	+3	+3	+3	+2	+3	+3
		Rank	3	2	1	1	1	1	1	1	1
	Distribution	Wins	0	3	3	3	3	3	3	3	1
		Losses	3	0	0	0	0	0	0	0	0
		Difference	-3	+3	+3	+3	+3	+3	+3	+3	+1
		Rank	4	1	1	1	1	1	1	1	1
CPSO-S_K	Hypervolume	Wins	3	3	2	2	2	2	2	2	2
		Losses	0	0	1	1	1	1	0	1	1
		Difference	+3	+3	+1	+1	+1	+1	+2	+1	+1
		Rank	1	1	2	2	2	2	1	2	2
	Distribution	Wins	1	2	0	0	1	0	0	0	0
		Losses	2	1	2	2	2	1	1	3	1
		Difference	-1	+1	-2	-2	-1	-1	-1	-3	-1
		Rank	3	2	3	4	3	2	3	4	4
CPSO-H_K	Hypervolume	Wins	2	1	1	1	1	1	1	1	0
		Losses	1	2	2	2	2	2	2	2	3
		Difference	+1	-1	-1	-1	-1	-1	-1	-1	-3
		Rank	2	3	3	3	3	3	3	3	4
	Distribution	Wins	2	0	0	1	0	0	0	1	0
		Losses	0	3	2	1	3	1	2	2	0
		Difference	+2	-3	-2	0	-3	-1	-2	-1	0
		Rank	1	4	3	2	4	2	4	3	2

Table 5.4: Mann-Whitney Wins and Losses For PCXArchive KTS

			WFG Function								
Algorithm	**Metric**	**Result**	1	2	3	4	5	6	7	8	9
PSO	Hypervolume	Wins	1	0	0	0	0	0	0	0	0
		Losses	2	3	2	3	2	2	3	2	2
		Difference	-1	-3	-2	-3	-2	-2	-3	-2	-2
		Rank	3	4	3	4	3	3	4	3	3
	Distribution	Wins	1	2	1	0	0	1	0	0	1
		Losses	0	0	1	1	1	0	1	1	0
		Difference	+1	+2	0	-1	-1	+1	-1	-1	+1
		Rank	2	1	2	2	2	1	2	2	1
CPSO-S	Hypervolume	Wins	0	2	3	3	3	3	3	3	3
		Losses	3	0	0	0	0	0	0	0	0
		Difference	-3	+2	+3	+3	+3	+3	+3	+3	+3
		Rank	4	1	1	1	1	1	1	1	1
	Distribution	Wins	0	0	3	3	3	1	3	3	1
		Losses	3	0	0	0	0	0	0	0	1
		Difference	-3	0	+3	+3	+3	+1	+3	+3	0
		Rank	4	2	1	1	1	1	1	1	3
CPSO-S_K	Hypervolume	Wins	3	2	2	2	2	2	2	2	2
		Losses	0	0	1	1	1	1	1	1	1
		Difference	+3	+2	+1	+1	+1	+1	+1	+1	+1
		Rank	1	1	2	2	2	2	2	2	2
	Distribution	Wins	1	0	0	0	0	0	0	0	0
		Losses	1	2	3	1	1	3	1	1	2
		Difference	0	-2	-3	-1	-1	-3	-1	-1	-2
		Rank	3	4	4	2	2	4	2	2	4
CPSO-H_K	Hypervolume	Wins	2	1	0	1	0	0	1	0	0
		Losses	1	2	2	2	2	2	2	2	2
		Difference	+1	-1	-2	-1	-2	-2	-1	-2	-2
		Rank	2	3	3	3	3	3	3	3	3
	Distribution	Wins	2	1	1	0	0	1	0	0	1
		Losses	0	1	1	1	1	0	1	1	0
		Difference	+2	0	0	-1	-1	+1	-1	-1	+1
		Rank	1	2	2	2	2	1	2	2	1

Table 5.5: Highest Ranking Algorithm In Each Group

Function	**Group**	**Hypervolume Measure**	**Distribution Measure**
WFG1	Ring	CPSO-S_K	PSO
	Random	CPSO-H_K	CPSO-H_K
	PCX Archive	CPSO-S_K	CPSO-H_K
	PCX GBest	CPSO-S_K	PSO
WFG2	Ring	CPSO-S	CPSO-S
	Random	CPSO-S_K	CPSO-H_K
	PCX Archive	CPSO-S	PSO
	PCX GBest	CPSO-S_K	CPSO-S
WFG3	Ring	CPSO-S	CPSO-S
	Random	CPSO-S	CPSO-S
	PCX Archive	CPSO-S	CPSO-S
	PCX GBest	CPSO-S	CPSO-S
WFG4	Ring	CPSO-S	CPSO-S
	Random	CPSO-S	CPSO-S
	PCX Archive	CPSO-S	CPSO-S
	PCX GBest	CPSO-S	CPSO-S
WFG5	Ring	CPSO-S	CPSO-S
	Random	CPSO-S	CPSO-S
	PCX Archive	CPSO-S	CPSO-S_K
	PCX GBest	CPSO-S	CPSO-S
WFG6	Ring	CPSO-S	CPSO-S
	Random	CPSO-S	CPSO-S
	PCX Archive	CPSO-S	CPSO-S
	PCX GBest	CPSO-S	CPSO-S
WFG7	Ring	CPSO-S	CPSO-S
	Random	CPSO-S	CPSO-S
	PCX Archive	CPSO-S	CPSO-S
	PCX GBest	CPSO-S	CPSO-S
WFG8	Ring	CPSO-S	CPSO-S
	Random	CPSO-S	CPSO-S
	PCX Archive	CPSO-S	CPSO-S
	PCX GBest	CPSO-S	CPSO-S
WFG9	Ring	CPSO-S	CPSO-S
	Random	CPSO-S	CPSO-S
	PCX Archive	CPSO-S	CPSO-H_K
	PCX GBest	CPSO-S	CPSO-S

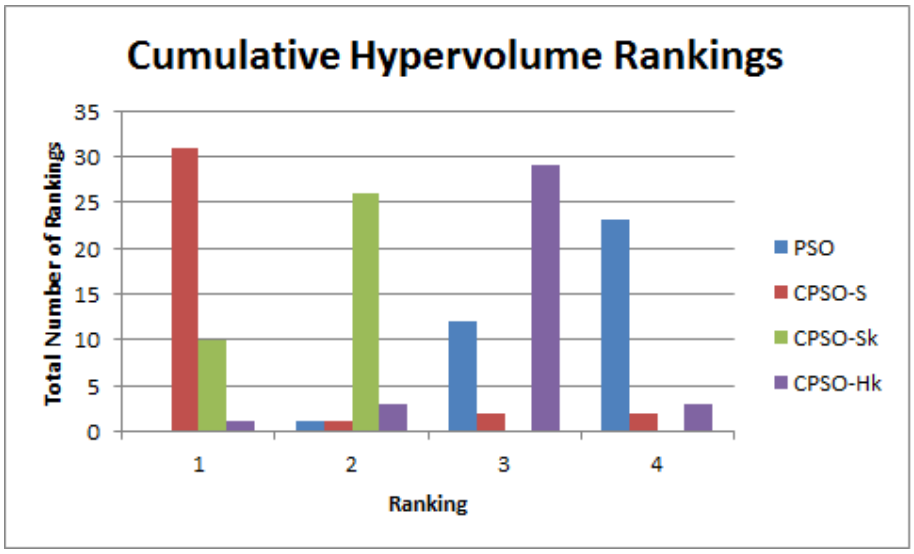

Figure 5.1: Cumulative rankings are shown for each algorithm with respect to the hypervolume metric.

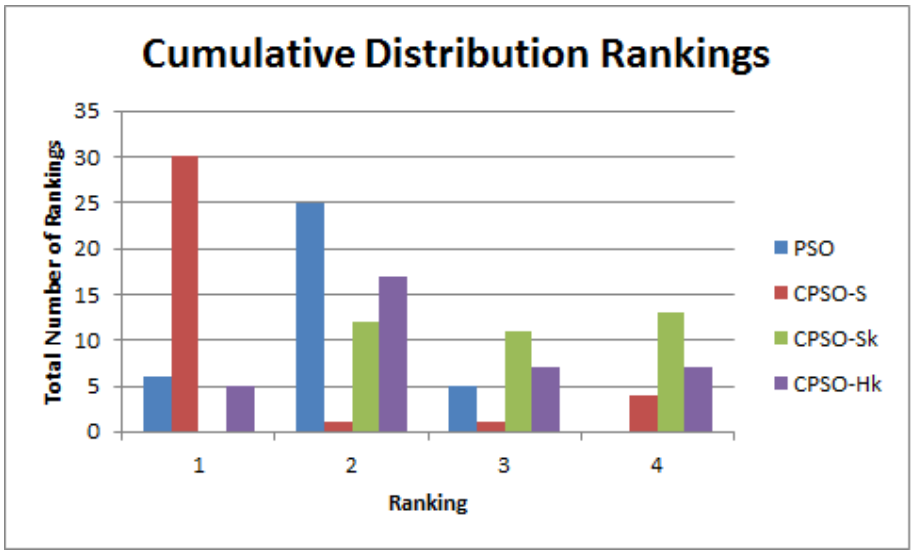

Figure 5.2: Cumulative rankings are shown for each algorithm with respect to the distribution metric.

Table 5.6: Mann-Whitney Wins and Losses For CPSO-S

			WFG Function								
Algorithm	**Metric**	**Result**	1	2	3	4	5	6	7	8	9
Ring	Hypervolume	Wins	0	2	1	1	1	1	1	1	1
		Losses	2	1	0	0	2	1	0	1	1
		Difference	-2	+1	+1	+1	-1	0	+1	0	0
		Rank	4	2	2	1	3	2	1	2	2
	Distribution	Wins	0	1	0	0	0	1	1	0	0
		Losses	2	2	1	2	1	2	1	1	2
		Difference	-2	-1	-1	-2	-1	-1	0	-1	-2
		Rank	4	3	4	4	3	3	2	3	4
Random	Hypervolume	Wins	0	1	0	0	0	1	1	0	2
		Losses	0	2	3	2	3	2	1	1	1
		Difference	0	-1	-3	-2	-3	-1	0	-1	+1
		Rank	2	3	4	4	4	3	3	3	1
	Distribution	Wins	0	1	1	1	1	0	0	0	2
		Losses	1	1	1	1	1	2	1	1	1
		Difference	-1	0	0	0	0	-2	-1	-1	+1
		Rank	3	2	2	3	2	4	3	3	1
PCX GBest	Hypervolume	Wins	2	3	1	1	2	3	2	2	1
		Losses	0	0	1	1	1	0	1	0	2
		Difference	+2	+3	0	0	+1	+3	+1	+2	-1
		Rank	1	1	3	3	2	1	1	1	4
	Distribution	Wins	2	2	2	1	3	3	3	1	1
		Losses	0	0	1	0	0	0	0	0	1
		Difference	+2	+2	+1	+1	+3	+3	+3	+1	0
		Rank	1	1	1	1	1	1	1	1	3
PCX Archive	Hypervolume	Wins	1	0	2	1	3	0	0	0	1
		Losses	1	3	0	0	0	2	2	1	1
		Difference	0	-3	+2	+1	+3	-2	-2	-1	0
		Rank	2	4	1	1	1	4	4	3	2
	Distribution	Wins	1	1	1	1	0	1	0	1	1
		Losses	0	2	1	0	2	1	2	0	0
		Difference	+1	-1	0	+1	-2	0	-2	+1	+1
		Rank	2	3	2	1	4	2	4	1	1

Table 5.7: Mann-Whitney Wins and Losses For CPSO-S_K

			WFG Function								
Algorithm	**Metric**	**Result**	1	2	3	4	5	6	7	8	9
Ring	Hypervolume	Wins	1	2	2	1	2	1	2	2	1
		Losses	0	1	1	0	1	2	1	0	2
		Difference	+1	+1	+1	+1	+1	-1	+1	+2	-1
		Rank	1	2	1	1	2	3	2	1	4
	Distribution	Wins	0	0	0	1	1	0	2	1	0
		Losses	1	2	1	1	2	2	1	0	2
		Difference	-1	-2	-1	0	-1	-2	+1	+1	-2
		Rank	4	3	3	2	3	4	1	2	4
Random	Hypervolume	Wins	0	0	0	0	0	0	1	0	1
		Losses	2	2	3	1	3	1	2	3	0
		Difference	-2	-2	-3	-1	-3	-1	-1	-3	+1
		Rank	4	3	4	3	4	3	3	4	1
	Distribution	Wins	1	0	1	0	2	1	1	0	1
		Losses	0	2	2	2	1	2	1	2	0
		Difference	+1	-2	-1	-2	+1	-1	0	-2	+1
		Rank	1	3	3	4	1	3	3	4	2
PCX GBest	Hypervolume	Wins	1	3	2	2	3	2	3	2	1
		Losses	1	0	1	1	0	0	0	1	1
		Difference	0	+3	+1	+1	+3	+2	+3	+1	0
		Rank	3	1	1	1	1	1	1	2	2
	Distribution	Wins	0	2	2	0	1	2	0	0	2
		Losses	0	0	0	1	2	1	2	1	0
		Difference	0	+2	+2	-1	-1	+1	-2	-1	+2
		Rank	2	1	1	3	3	2	4	3	1
PCX Archive	Hypervolume	Wins	1	0	2	0	1	1	0	1	1
		Losses	0	2	1	1	2	1	3	1	1
		Difference	+1	-2	+1	-1	-1	0	-3	0	0
		Rank	1	3	1	3	3	2	4	3	2
	Distribution	Wins	0	2	1	3	2	2	2	2	1
		Losses	0	0	1	0	1	0	1	0	2
		Difference	0	+2	0	+3	+1	+2	+1	+2	-1
		Rank	2	1	2	1	1	1	1	1	3

Table 5.8: Mann-Whitney Wins and Losses For CPSO-H$_K$

			WFG Function								
Algorithm	**Metric**	**Result**	1	2	3	4	5	6	7	8	9
Ring	Hypervolume	Wins	0	2	1	1	2	0	0	1	1
		Losses	2	0	1	0	1	1	3	1	1
		Difference	-2	+2	0	+1	+1	-1	-3	0	0
		Rank	4	1	3	1	2	3	4	2	2
	Distribution	Wins	1	2	0	0	1	0	0	0	0
		Losses	1	0	2	2	0	2	2	2	1
		Difference	0	+2	-2	-2	+1	-2	-2	-2	-1
		Rank	2	1	4	4	1	3	4	4	3
Random	Hypervolume	Wins	1	0	0	0	0	1	2	0	0
		Losses	1	1	3	2	2	1	1	1	1
		Difference	0	-1	-3	-2	-2	0	+1	-1	-1
		Rank	3	3	4	4	4	2	2	3	3
	Distribution	Wins	0	0	2	1	0	3	1	1	0
		Losses	1	3	1	0	2	0	2	2	2
		Difference	-1	-3	+1	+1	-2	+3	-1	-1	-2
		Rank	4	4	1	1	4	1	3	3	4
PCX GBest	Hypervolume	Wins	1	2	2	1	2	3	3	2	2
		Losses	0	1	1	1	0	0	0	0	0
		Difference	+1	+1	+1	0	+2	+3	+3	+2	+2
		Rank	1	2	2	3	1	1	1	1	1
	Distribution	Wins	1	1	1	1	0	0	1	1	1
		Losses	1	1	1	0	0	2	0	1	0
		Difference	0	0	0	+1	0	-2	+1	0	+1
		Rank	2	3	3	1	3	3	2	2	2
PCX Archive	Hypervolume	Wins	1	0	2	1	1	0	1	1	0
		Losses	0	2	0	0	2	2	2	2	1
		Difference	+1	-2	+2	+1	-1	-2	-1	-1	-1
		Rank	1	4	1	1	3	4	3	3	3
	Distribution	Wins	2	1	2	1	1	2	2	3	2
		Losses	1	0	1	1	0	1	0	0	0
		Difference	+1	+1	+1	0	+1	+1	+2	+3	+2
		Rank	1	2	1	3	1	2	1	1	1

Table 5.9: Highest Ranking KTS For Each Algorithm

Function	**Group**	**Hypervolume Measure**	**Distribution Measure**
WFG1	CPSO-S	PCX GBest	PCX GBest
	CPSO-S_K	Ring	Random
	CPSO-H_K	PCX GBest	PCX Archive
WFG2	CPSO-S	PCX GBest	PCX GBest
	CPSO-S_K	PCX GBest	PCX Archive
	CPSO-H_K	Ring	Ring
WFG3	CPSO-S	PCX Archive	PCX GBest
	CPSO-S_K	PCX GBest	PCX GBest
	CPSO-H_K	PCX Archive	PCX Archive
WFG4	CPSO-S	PCX Archive	PCX Archive
	CPSO-S_K	PCX GBest	PCX Archive
	CPSO-H_K	PCX Archive	Random
WFG5	CPSO-S	PCX Archive	PCX GBest
	CPSO-S_K	PCX GBest	PCX Archive
	CPSO-H_K	PCX GBest	Ring
WFG6	CPSO-S	PCX GBest	PCX GBest
	CPSO-S_K	PCX GBest	PCX Archive
	CPSO-H_K	PCX GBest	Random
WFG7	CPSO-S	PCX GBest	PCX GBest
	CPSO-S_K	PCX GBest	PCX Archive
	CPSO-H_K	PCX GBest	PCX Archive
WFG8	CPSO-S	PCX GBest	PCX Archive
	CPSO-S_K	Ring	PCX Archive
	CPSO-H_K	PCX GBest	PCX Archive
WFG9	CPSO-S	Random	PCX Archive
	CPSO-S_K	Random	PCX GBest
	CPSO-H_K	PCX GBest	PCX Archive

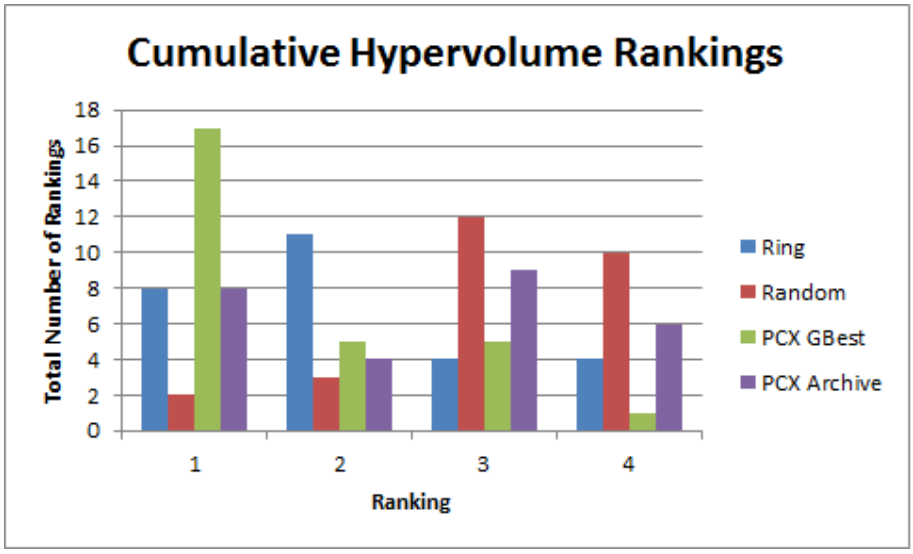

Figure 5.3: Cumulative rankings are shown for each KTS with respect to the hypervolume metric.

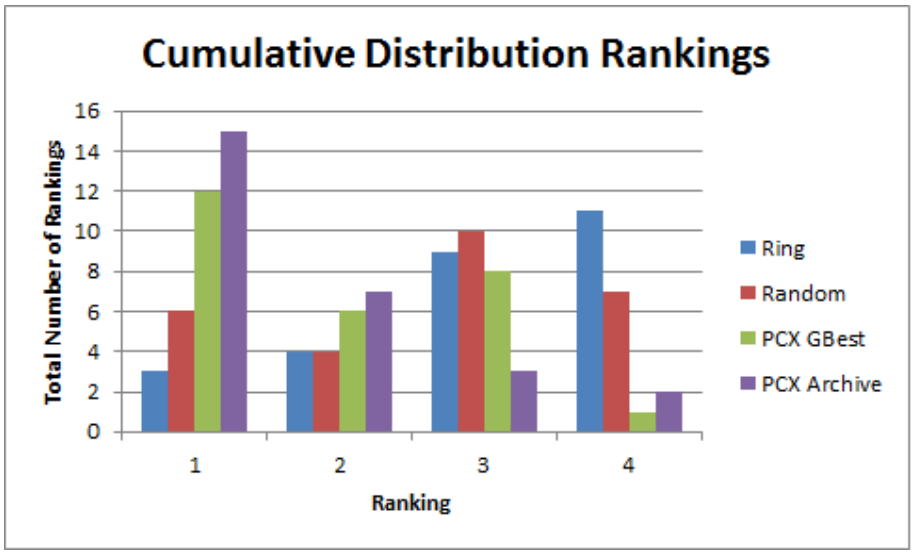

Figure 5.4: Cumulative rankings are shown for each KTS with respect to the distribution metric.

Table 5.10: Mann-Whitney Wins and Losses In 3-D Objective Space

			WFG Function								
Algorithm	**Metric**	**Result**	1	2	3	4	5	6	7	8	9
CPSO-S	Hypervolume	Wins	0	0	1	3	3	2	1	1	1
		Losses	3	3	2	0	0	0	1	2	2
		Difference	-3	-3	-1	+3	+3	+2	0	-1	-1
		Rank	4	4	3	1	1	1	3	3	3
	Distribution	Wins	0	1	1	3	3	2	3	3	1
		Losses	3	2	2	0	0	0	0	0	2
		Difference	-3	-1	-1	+3	+3	+2	+3	+3	-1
		Rank	4	3	3	1	1	1	1	1	3
CPSO-S_K	Hypervolume	Wins	1	1	0	1	0	0	1	0	0
		Losses	2	2	3	1	3	3	0	3	3
		Difference	-1	-1	-3	0	-3	-3	+1	-3	-3
		Rank	3	3	4	2	4	4	2	4	4
	Distribution	Wins	1	0	0	0	0	0	0	0	0
		Losses	2	3	3	3	3	3	3	3	3
		Difference	-1	-3	-3	-3	-3	-3	-3	-3	-3
		Rank	3	4	4	4	4	4	4	4	4
oMOPSO	Hypervolume	Wins	2	3	3	1	1	2	2	2	3
		Losses	0	0	0	1	1	0	0	1	0
		Difference	+2	+3	+3	0	0	+2	+2	+1	+3
		Rank	1	1	1	2	2	1	1	2	1
	Distribution	Wins	3	2	2	2	1	2	1	1	2
		Losses	0	0	0	1	1	0	1	1	0
		Difference	+3	+2	+2	+1	0	+2	0	0	+2
		Rank	1	1	1	2	2	1	2	2	1
SMPSO	Hypervolume	Wins	2	2	2	0	1	1	0	3	2
		Losses	0	1	1	3	1	2	3	0	1
		Difference	+2	+1	+1	-3	0	-1	-3	+3	+1
		Rank	1	2	2	4	2	3	4	1	2
	Distribution	Wins	2	2	2	1	1	1	1	1	2
		Losses	1	0	0	2	1	2	1	1	0
		Difference	+1	+2	+2	-1	0	-1	0	0	+2
		Rank	2	1	1	3	2	3	2	2	1

Table 5.11: Mann-Whitney Wins and Losses In 5-D Objective Space

			WFG Function								
Algorithm	**Metric**	**Result**	1	2	3	4	5	6	7	8	9
CPSO-S	Hypervolume	Wins	0	0	0	2	2	2	2	2	3
		Losses	3	3	3	1	1	0	1	1	0
		Difference	-3	-3	-3	+1	+1	+2	+1	+1	+3
		Rank	4	4	4	2	2	1	2	2	1
	Distribution	Wins	0	1	2	3	3	3	1	3	3
		Losses	3	2	1	0	0	0	0	0	0
		Difference	-3	-1	+1	+3	+3	+3	+1	+3	+3
		Rank	4	3	2	1	1	1	2	1	1
CPSO-S_K	Hypervolume	Wins	1	3	1	3	3	2	3	3	0
		Losses	2	0	2	0	0	0	0	0	2
		Difference	-1	+3	-1	+3	+3	+2	+3	+3	0
		Rank	3	1	3	1	1	1	1	1	2
	Distribution	Wins	1	0	3	0	0	0	0	0	0
		Losses	2	3	0	3	3	3	3	3	3
		Difference	-1	-3	+3	-3	-3	-3	-3	-3	-3
		Rank	3	4	1	4	4	4	4	4	4
oMOPSO	Hypervolume	Wins	2	1	2	0	1	0	0	0	1
		Losses	0	1	0	3	2	3	3	2	2
		Difference	+2	0	+2	-3	-1	-3	-3	-2	-1
		Rank	1	2	1	4	3	4	4	3	3
	Distribution	Wins	3	2	0	1	1	2	2	1	1
		Losses	0	0	2	1	2	1	0	1	1
		Difference	+3	+2	-2	0	-1	+1	+2	0	0
		Rank	1	1	3	2	3	2	1	2	2
SMPSO	Hypervolume	Wins	2	1	2	1	0	1	1	0	1
		Losses	0	1	0	2	3	2	2	2	1
		Difference	+2	0	+2	-1	-3	-1	-1	-2	0
		Rank	1	2	1	3	4	3	3	3	2
	Distribution	Wins	2	2	0	1	2	1	1	1	1
		Losses	1	0	2	1	1	2	1	1	1
		Difference	+1	+2	-2	0	+1	-1	0	0	0
		Rank	2	1	3	2	2	3	3	2	2

Table 5.12: Highest Ranking Algorithm In Each Group

Function	**Group**	**Hypervolume Measure**	**Distribution Measure**
WFG1	3-D	oMOPSO	oMOPSO
	5-D	SMPSO	oMOPSO
WFG2	3-D	oMOPSO	oMOPSO
	5-D	CPSO-S_K	SMPSO
WFG3	3-D	oMOPSO	oMOPSO
	5-D	SMPSO	CPSO-S_K
WFG4	3-D	CPSO-S	CPSO-S
	5-D	CPSO-S_K	CPSO-S
WFG5	3-D	CPSO-S	CPSO-S
	5-D	CPSO-S_K	CPSO-S
WFG6	3-D	CPSO-S	CPSO-S
	5-D	CPSO-S	CPSO-S
WFG7	3-D	oMOPSO	CPSO-S
	5-D	CPSO-S_K	oMOPSO
WFG8	3-D	SMPSO	CPSO-S
	5-D	CPSO-S_K	CPSO-S
WFG9	3-D	oMOPSO	oMOPSO
	5-D	CPSO-S	CPSO-S

Chapter 6

Concluding Remarks and Future Work

This thesis examined the performance effects of using various co-operative PSO variants within VEPSO swarms. Performance of VEPSO while using the CPSO-S, CPSO-S_K and CPSO-H_K algorithms were all measured over multiple knowledge-transfer strategies to ensure a diverse and complete testing environment. KTS performance was also examined for each of the co-operative variants. Well-known performance measures were used to gauge the effectiveness of each algorithm. The best performing algorithm-KTS combinations were then compared to well-known multi-objective optimization algorithms. Statistical procedures were performed on results to compare the different algorithms/strategies and determine which had superior performance, if any.

Results indicated that co-operation is a powerful addition to VEPSO. PSO was outperformed in terms of hypervolume by CPSO-S, CPSO-S_K and CPSO-H_K for all knowledge-transfer strategies. PSO obtained equal or better distribution in comparison to CPSO-S_K and CPSO-H_K. CPSO-S was dominant over PSO in both metrics, as it consistently earned higher placings.

It has been observed that CPSO-S is more sensitive to functions which contain complex shapes. Functions which were strictly convex seemed to present the most

difficuly, however disconnection, linearity and degeneration were also observed to occasionally yield performance degradation. This should be kept in mind when selecting CPSO-S for use in VEPSO swarms, as performance may be slightly worse in these types of environments. This may not be possible to avoid as the function shape would have to be known *a priori* to optimization, a feat which is very rare in practice.

Concerning co-operative variant selection, it is very apparent that CPSO-S and CPSO-S_K are the ideal choices for use within VEPSO. It was concluded that CPSO-H_K did not improve performance over CPSO-S or CPSO-S_K in deceptive environments. Since this is the main reason that CPSO-H_K was created, there does not seem to be any reason to use this algorithm over the others. Performance was repeatedly observed to be worse than CPSO-S and CPSO-S_K.

Results of the KTS experiments revealed that CPSO-S worked extremely well with the PCX GBest strategy, as it had better hypervolume and at least equal distribution in comparison to all other strategies. It was also observed for CPSO-S that the ring strategy only outperformed the random strategy for hypervolume, which was not the case for PSO (in previous literature). In the case of CPSO-S_K, PCXArchive ranked highest in terms of the distribution measure and PCXGBest ranked highest for hypervolume. Contrary to observations in previous work concerning PSO, the ring strategy did not respond more favourably to deceptive functions. However, conclusions drawn were still nearly identical to PSO, indicating that these conclusions were likely independent of the algorithm used.

Conclusions drawn from KTS experiments on CPSO-H_K were predictably a combination of the algorithms which it consists of, CPSO-S_K and PSO. The ring strategy recorded higher hypervolume than the random strategy, however both strategies were relatively equal in terms of solution distribution. However, the ring strategy performed very well in the presence of deception and disconnection, a trait likely inherited from PSO. Concerning the distribution measure, the PCX Archive strategy

was superior to all others. PCX GBest ranked highest overall for the hypervolume measure.

The two best performing algorithm-KTS combinations, CPSO-S PCXGBest and CPSO-S_K PCXGBest, were compared to two well known multi-objective optimization algorithms. CPSO-S_K scaled exceptionally well in terms of hypervolume but poorly in terms of solution distribution. It was concluded that CPSO-S_K is ideal when one desires higher hypervolume with little regard for solution distribution. CPSO-S exhibited a balance between hypervolume and solution distribution, performing equally as well or better than oMOPSO and SMPSO for all functions except the few with complex shapes. Both algorithms were found to be very competitive with the well known oMOPSO and SMPSO algorithms, as they often performed equally or better. Previous work has indicated that traditional VEPSO stagnates and does not scale well with these algorithms. Thus using co-operative strategies within VEPSO improves it considerably, allowing it to finally contend with the top multi-objective algorithms.

There are many intriguing opportunities for future work in this area. More research into the scalability of CPSO-S_K and CPSO-S is encouraged, specifically when the number of objectives becomes very large. Another potential area of interest is the sensitivity to dimensionality of VEPSO. Performance of VEPSO while using co-operative variants over a varying number of decision dimensions can be observed with the intent of determining the corresponding sensitivity to decision-space dimensionality in comparison to traditional VEPSO. This can also be repeated for the number of objectives to investigate the VEPSO performance influence of objective-space size on the swarm algorithm chosen. Finally, the knowledge transfer mechanism of CPSO-H_K is a viable area to improve as no previous work on this subject exists.

Bibliography

[1] L. Bradstreet, University of Western Australia. School of Computer Science, and Software Engineering. *The Hypervolume Indicator for Multi-objective Optimisation: Calculation and Use*. University of Western Australia, 2011.

[2] Karl Bringmann and Tobias Friedrich. Approximating the volume of unions and intersections of high-dimensional geometric objects. In Seok-Hee Hong, Hiroshi Nagamochi, and Takuro Fukunaga, editors, *Algorithms and Computation*, volume 5369 of *Lecture Notes in Computer Science*, pages 436–447. Springer Berlin Heidelberg, 2008.

[3] Anthony Carlisle and Gerry Dozier. An Off-The-Shelf PSO. 2001.

[4] Anthony Jack Carlisle. *Applying the Particle Swarm Optimizer to Non-stationary Environments*. PhD thesis, Auburn, AL, USA, 2002. AAI3070763.

[5] Rachid Chelouah and Claude Baron. Ant colony algorithm hybridized with tabu and greedy searches as applied to multi-objective optimization in project management. *Journal of Heuristics*, 13(6):640–640, 2007.

[6] Scott H. Clearwater, Tad Hogg, and Bernardo A. Huberman. Cooperative problem solving. In *COMPUTATION: THE MICRO AND THE MACRO VIEW*, pages 33–70. World Scientific, 1992.

[7] M. Clerc. The swarm and the queen: towards a deterministic and adaptive particle swarm optimization. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 3, pages –1957 Vol. 3, 1999.

[8] C.A.C. Coello, G.T. Pulido, and M.S. Lechuga. Handling multiple objectives with particle swarm optimization. *Evolutionary Computation, IEEE Transactions on*, 8(3):256–279, June 2004.

[9] C. A. Coello Coello and M. S. Lechuga. Mopso: A proposal for multiple objective particle swarm optimization. In *Proceedings of the Evolutionary Computation on 2002. CEC '02. Proceedings of the 2002 Congress - Volume 02*, CEC '02, pages 1051–1056, Washington, DC, USA, 2002. IEEE Computer Society.

[10] Carlos A Coello Coello and M.S. Lechuga. Mopso: a proposal for multiple objective particle swarm optimization. In *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on*, volume 2, pages 1051–1056, 2002.

[11] Richard K. Crump, V. Joseph Hotz, Guido W. Imbens, and Oscar A. Mitnik. Nonparametric tests for treatment effect heterogeneity. Working Paper 324, National Bureau of Economic Research, June 2006.

[12] Kalyanmoy Deb. Multi-objective genetic algorithms: Problem difficulties and construction of test problems. *Evol. Comput.*, 7(3):205–230, September 1999.

[13] Kalyanmoy Deb, Ashish Anand, and Dhiraj Joshi. A computationally efficient evolutionary algorithm for real-parameter optimization. *Evol. Comput.*, 10(4):371–395, December 2002.

[14] F. Djeffal and N. Lakhdar. An improved analog electrical performance of submicron dual-material gate (dm) gaas-mesfets using multi-objective computation. *Journal of Computational Electronics*, 12(1):29–35, 2013.

[15] Juan J. Durillo and Antonio J. Nebro. jmetal: A java framework for multi-objective optimization. *Advances in Engineering Software*, 42(10):760 – 771, 2011.

[16] Mohammed El-Abd. *Cooperative Models of Particle Swarm Optimizers*. PhD thesis, Waterloo, Ont., Canada, Canada, 2008. AAINR43264.

[17] Mohammed El-Abd, Hassan Hassan, Mohab Anis, Mohamed S. Kamel, and Mohamed Elmasry. Discrete cooperative particle swarm optimization for fpga placement. *Appl. Soft Comput.*, 10(1):284–295, January 2010.

[18] M. Fleischer. The measure of pareto optima applications to multi-objective metaheuristics. In *Proceedings of the 2Nd International Conference on Evolutionary Multi-criterion Optimization*, EMO'03, pages 519–533, Berlin, Heidelberg, 2003. Springer-Verlag.

[19] M. Fleischer. The measure of pareto optima applications to multi-objective metaheuristics. In *Proceedings of the 2Nd International Conference on Evolutionary Multi-criterion Optimization*, EMO'03, pages 519–533, Berlin, Heidelberg, 2003. Springer-Verlag.

[20] C.K. Goh and K.C. Tan. An investigation on noisy environments in evolutionary multiobjective optimization. *Evolutionary Computation, IEEE Transactions on*, 11(3):354–381, June 2007.

[21] Jacomine Grobler, Andries P. Engelbrecht, and V. S. S. Yadavalli. Multi-objective de and pso strategies for production scheduling. In Jun Wang, editor, *2008 IEEE World Congress on Computational Intelligence*, pages –, Hong Kong, 1-6 June 2008. IEEE Computational Intelligence Society, IEEE Press.

[22] K.R. Harrison, AP. Engelbrecht, and B.M. Ombuki-Berman. A scalability study of multi-objective particle swarm optimizers. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 189–197, June 2013.

[23] Kyle Robert Harrison, Beatrice Ombuki-Berman, and Andries P. Engelbrecht. Knowledge transfer strategies for vector evaluated particle swarm optimization. In RobinC. Purshouse, PeterJ. Fleming, CarlosM. Fonseca, Salvatore Greco, and Jane Shaw, editors, *Evolutionary Multi-Criterion Optimization*, volume 7811 of *Lecture Notes in Computer Science*, pages 171–184. Springer Berlin Heidelberg, 2013.

[24] J.S. Heo, K.Y. Lee, and R. Garduno-Ramirez. Multiobjective control of power plants using particle swarm optimization techniques. *Energy Conversion, IEEE Transactions on*, 21(2):552–561, June 2006.

[25] S.L. Ho, Shiyou Yang, Guangzheng Ni, and H. C. Wong. A particle swarm optimization method with enhanced global search ability for design optimizations of electromagnetic devices. *Magnetics, IEEE Transactions on*, 42(4):1107–1110, April 2006.

[26] Xiaohui Hu, R.C. Eberhart, and Yuhui Shi. Swarm intelligence for permutation optimization: a case study of n-queens problem. In *Swarm Intelligence Symposium, 2003. SIS '03. Proceedings of the 2003 IEEE*, pages 243–246, April 2003.

[27] Bufu Huang, Zhancheng Wang, and Yangsheng Xu. Multi-objective genetic algorithm for hybrid electric vehicle parameter optimization. In *Intelligent Robots*

and Systems, 2006 IEEE/RSJ International Conference on, pages 5177–5182, Oct 2006.

[28] S. Huband, P. Hingston, L. Barone, and L. While. A review of multiobjective test problems and a scalable test problem toolkit. *Evolutionary Computation, IEEE Transactions on*, 10(5):477–506, Oct 2006.

[29] Simon Huband, Luigi Barone, Lyndon While, and Phil Hingston. A scalable multi-objective test problem toolkit. In CarlosA. Coello Coello, Arturo Herndez Aguirre, and Eckart Zitzler, editors, *Evolutionary Multi-Criterion Optimization*, volume 3410 of *Lecture Notes in Computer Science*, pages 280–295. Springer Berlin Heidelberg, 2005.

[30] I.N. Kassabalidis, M.A. El-Sharkawi, R.J. Marks, L.S. Moulin, and A.P. Alves da Silva. Dynamic security border identification using enhanced particle swarm optimization. *Power Systems, IEEE Transactions on*, 17(3):723–729, Aug 2002.

[31] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948, Nov 1995.

[32] Salinda Buyamin Anita Ahmad Faradila Naim Kamarul Hawari Ghazali Norrima Mokhtar Kian Sheng Lim, Zuwairie Ibrahim. Improving vector evaluated particle swarm optimisation by incorporating nondominated solutions, 2013.

[33] Satoshi Kitayama, Jirasak Srirat, Masao Arakawa, and Koetsu Yamazaki. Sequential approximate multi-objective optimization using radial basis function network. *Structural and Multidisciplinary Optimization*, 48(3):501–515, 2013.

[34] J. Knowles, L. Thiele, and E. Zitzler. A Tutorial on the Performance Assessment of Stochastic Multiobjective Optimizers. Technical report, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Switzerland, 2006.

[35] Xiaodong Li, Senior Member, and Xin Yao. Cooperatively coevolving particle swarms for large scale optimization. *IEEE Transactions on Evolutionary Computation*, pages 210–224, 2012.

[36] Xiaodong Li and Xin Yao. Tackling high dimensional nonseparable optimization problems by cooperatively coevolving particle swarms. In *Proceedings of the Eleventh Conference on Congress on Evolutionary Computation*, CEC'09, pages 1546–1553, Piscataway, NJ, USA, 2009. IEEE Press.

[37] Wen-Chung Liu. Design of a multiband cpw-fed monopole antenna using a particle swarm optimization approach. *Antennas and Propagation, IEEE Transactions on*, 53(10):3273–3279, Oct 2005.

[38] H. B. Mann and D. R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *The Annals of Mathematical Statistics*, 18(1):50–60, 03 1947.

[39] Makoto Matsumoto and Takuji Nishimura. Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. Model. Comput. Simul.*, 8(1):3–30, January 1998.

[40] W. Matthysen, A.P. Engelbrecht, and K.M. Malan. Analysis of stagnation behavior of vector evaluated particle swarm optimization. In *Swarm Intelligence (SIS), 2013 IEEE Symposium on*, pages 155–163, April 2013.

[41] AJ. Nebro, J.J. Durillo, J. Garcia-Nieto, C.A Coello Coello, F. Luna, and E. Alba. Smpso: A new pso-based metaheuristic for multi-objective optimization. In *Computational intelligence in miulti-criteria decision-making, 2009. mcdm '09. ieee symposium on*, pages 66–73, March 2009.

[42] O. Olorunda and A.P. Engelbrecht. An analysis of heterogeneous cooperative algorithms. In *Evolutionary Computation, 2009. CEC '09. IEEE Congress on*, pages 1562–1569, May 2009.

[43] G. Pampara, A.P. Engelbrecht, and T. Cloete. Cilib: A collaborative framework for computational intelligence algorithms - part i. In *Neural Networks, 2008. IJCNN 2008. (IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, pages 1750–1757, June 2008.

[44] K. E. Parsopoulos, D. K. Tasoulis, M. N. Vrahatis, and Key Words. Multiobjective optimization using parallel vector evaluated particle swarm optimization. In *In Proceedings of the IASTED International Conference on Artificial Intelligence and Applications (AIA 2004*, pages 823–828. ACTA Press, 2004.

[45] K. E. Parsopoulos and M. N. Vrahatis. Particle swarm optimization method in multiobjective problems. In *Proceedings of the 2002 ACM Symposium on Applied Computing*, SAC '02, pages 603–607, New York, NY, USA, 2002. ACM.

[46] Konstantinos E. Parsopoulos. Parallel cooperative micro-particle swarm optimization: A masterslave model. *Applied Soft Computing*, 12(11):3552 – 3579, 2012.

[47] M. Rosendo and A. Pozo. Applying a discrete particle swarm optimization algorithm to combinatorial problems. In *Neural Networks (SBRN), 2010 Eleventh Brazilian Symposium on*, pages 235–240, Oct 2010.

[48] J. David Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pages 93–100, Hillsdale, NJ, USA, 1985. L. Erlbaum Associates Inc.

[49] X.H. Shi, Y. Zhou, L.M. Wang, Q.X. Wang, and Y.C. Liang. A discrete particle swarm optimization algorithm for travelling salesman problem. In G.R. LIU, V.B.C. TAN, and X. HAN, editors, *Computational Methods*, pages 1063–1068. Springer Netherlands, 2006.

[50] Yuhui Shi and RussellC. Eberhart. Parameter selection in particle swarm optimization. In V.W. Porto, N. Saravanan, D. Waagen, and A.E. Eiben, editors, *Evolutionary Programming VII*, volume 1447 of *Lecture Notes in Computer Science*, pages 591–600. Springer Berlin Heidelberg, 1998.

[51] Joseph E. Stiglitz. Self-selection and Pareto efficient taxation. *Journal of Public Economics*, 17(2):213–240, March 1982.

[52] G.S. Tewolde, D.M. Hanna, and R.E. Haskell. Enhancing performance of pso with automatic parameter tuning technique. In *Swarm Intelligence Symposium, 2009. SIS '09. IEEE*, pages 67–73, March 2009.

[53] N.J. Unger, B.M. Ombuki-Berman, and A.P. Engelbrecht. Cooperative particle swarm optimization in dynamic environments. In *Swarm Intelligence (SIS), 2013 IEEE Symposium on*, pages 172–179, April 2013.

[54] F. van den Bergh and A. Engelbrecht. Cooperative Learning in Neural Networks using Particle Swarm Optimizers, 2000.

[55] F. Van den Bergh and A.P. Engelbrecht. A cooperative approach to particle swarm optimization. *Evolutionary Computation, IEEE Transactions on*, 8(3):225–239, June 2004.

[56] Frans Van Den Bergh. *An Analysis of Particle Swarm Optimizers.* PhD thesis, Pretoria, South Africa, South Africa, 2002. AAI0804353.

[57] L. While, P. Hingston, L. Barone, and S. Huband. A faster algorithm for calculating hypervolume. *Evolutionary Computation, IEEE Transactions on*, 10(1):29–38, Feb 2006.

[58] Wan-zhong Zhao, Chun-yan Wang, Lei-yan Yu, and Tao Chen. Performance optimization of electric power steering based on multi-objective genetic algorithm. *Journal of Central South University*, 20(1):98–104, 2013.

[59] E. Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications.* PhD thesis, ETH Zurich, 1999.

[60] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *Evolutionary Computation, IEEE Transactions on*, 3(4):257–271, Nov 1999.

[61] E. Zitzler, L. Thiele, M. Laumanns, C.M. Fonseca, and V.G. da Fonseca. Performance assessment of multiobjective optimizers: an analysis and review. *Evolutionary Computation, IEEE Transactions on*, 7(2):117–132, April 2003.

[62] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evol. Comput.*, 8(2):173–195, June 2000.

[63] Eckart Zitzler and Lothar Thiele. Multiobjective optimization using evolutionary algorithms a comparative case study. In AgostonE. Eiben, Thomas Bck, Marc Schoenauer, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature PPSN V*, volume 1498 of *Lecture Notes in Computer Science*, pages 292–301. Springer Berlin Heidelberg, 1998.

Appendix A

Summary of Experiments Performed

Table A.1 presents a comprehensive summary of all experiments performed. *Pruning* denotes the solution selection method which was invoked if the archive size limit was reached. Each experiment was repeated 35 times to ensure statistical significance.

Table A.1: Experiments Performed

Algorithm	**Objectives**	**KTS**	**Function**	**Archive Size**	**Pruning**
PSO	3	Ring	WFG1	250	Random
PSO	3	Ring	WFG2	250	Random
PSO	3	Ring	WFG3	250	Random
PSO	3	Ring	WFG4	250	Random
PSO	3	Ring	WFG5	250	Random
PSO	3	Ring	WFG6	250	Random
PSO	3	Ring	WFG7	250	Random

Continued on next page

Table A.1: Experiments Performed (continued)

Algorithm	**Objectives**	**KTS**	**Function**	**Archive Size**	**Pruning**
PSO	3	Ring	WFG8	250	Random
PSO	3	Ring	WFG9	250	Random
CPSO-S	3	Ring	WFG1	250	Random
CPSO-S	3	Ring	WFG2	250	Random
CPSO-S	3	Ring	WFG3	250	Random
CPSO-S	3	Ring	WFG4	250	Random
CPSO-S	3	Ring	WFG5	250	Random
CPSO-S	3	Ring	WFG6	250	Random
CPSO-S	3	Ring	WFG7	250	Random
CPSO-S	3	Ring	WFG8	250	Random
CPSO-S	3	Ring	WFG9	250	Random
CPSO-S_K	3	Ring	WFG1	250	Random
CPSO-S_K	3	Ring	WFG2	250	Random
CPSO-S_K	3	Ring	WFG3	250	Random
CPSO-S_K	3	Ring	WFG4	250	Random
CPSO-S_K	3	Ring	WFG5	250	Random
CPSO-S_K	3	Ring	WFG6	250	Random

Continued on next page

Table A.1: Experiments Performed (continued)

Algorithm	**Objectives**	**KTS**	**Function**	**Archive Size**	**Pruning**
CPSO-S_K	3	Ring	WFG7	250	Random
CPSO-S_K	3	Ring	WFG8	250	Random
CPSO-S_K	3	Ring	WFG9	250	Random
CPSO-H_K	3	Ring	WFG1	250	Random
CPSO-H_K	3	Ring	WFG2	250	Random
CPSO-H_K	3	Ring	WFG3	250	Random
CPSO-H_K	3	Ring	WFG4	250	Random
CPSO-H_K	3	Ring	WFG5	250	Random
CPSO-H_K	3	Ring	WFG6	250	Random
CPSO-H_K	3	Ring	WFG7	250	Random
CPSO-H_K	3	Ring	WFG8	250	Random
CPSO-H_K	3	Ring	WFG9	250	Random
PSO	3	Random	WFG1	250	Random
PSO	3	Random	WFG2	250	Random
PSO	3	Random	WFG3	250	Random
PSO	3	Random	WFG4	250	Random
PSO	3	Random	WFG5	250	Random

Continued on next page

Table A.1: Experiments Performed (continued)

Algorithm	**Objectives**	**KTS**	**Function**	**Archive Size**	**Pruning**
PSO	3	Random	WFG6	250	Random
PSO	3	Random	WFG7	250	Random
PSO	3	Random	WFG8	250	Random
PSO	3	Random	WFG9	250	Random
CPSO-S	3	Random	WFG1	250	Random
CPSO-S	3	Random	WFG2	250	Random
CPSO-S	3	Random	WFG3	250	Random
CPSO-S	3	Random	WFG4	250	Random
CPSO-S	3	Random	WFG5	250	Random
CPSO-S	3	Random	WFG6	250	Random
CPSO-S	3	Random	WFG7	250	Random
CPSO-S	3	Random	WFG8	250	Random
CPSO-S	3	Random	WFG9	250	Random
CPSO-S_K	3	Random	WFG1	250	Random
CPSO-S_K	3	Random	WFG2	250	Random
CPSO-S_K	3	Random	WFG3	250	Random
CPSO-S_K	3	Random	WFG4	250	Random

Continued on next page

Table A.1: Experiments Performed (continued)

Algorithm	**Objectives**	**KTS**	**Function**	**Archive Size**	**Pruning**
CPSO-S_K	3	Random	WFG5	250	Random
CPSO-S_K	3	Random	WFG6	250	Random
CPSO-S_K	3	Random	WFG7	250	Random
CPSO-S_K	3	Random	WFG8	250	Random
CPSO-S_K	3	Random	WFG9	250	Random
CPSO-H_K	3	Random	WFG1	250	Random
CPSO-H_K	3	Random	WFG2	250	Random
CPSO-H_K	3	Random	WFG3	250	Random
CPSO-H_K	3	Random	WFG4	250	Random
CPSO-H_K	3	Random	WFG5	250	Random
CPSO-H_K	3	Random	WFG6	250	Random
CPSO-H_K	3	Random	WFG7	250	Random
CPSO-H_K	3	Random	WFG8	250	Random
CPSO-H_K	3	Random	WFG9	250	Random
PSO	3	PCX GBest	WFG1	250	Random
PSO	3	PCX GBest	WFG2	250	Random
PSO	3	PCX GBest	WFG3	250	Random

Continued on next page

Table A.1: Experiments Performed (continued)

Algorithm	**Objectives**	**KTS**	**Function**	**Archive Size**	**Pruning**
PSO	3	PCX GBest	WFG4	250	Random
PSO	3	PCX GBest	WFG5	250	Random
PSO	3	PCX GBest	WFG6	250	Random
PSO	3	PCX GBest	WFG7	250	Random
PSO	3	PCX GBest	WFG8	250	Random
PSO	3	PCX GBest	WFG9	250	Random
CPSO-S	3	PCX GBest	WFG1	250	Random
CPSO-S	3	PCX GBest	WFG2	250	Random
CPSO-S	3	PCX GBest	WFG3	250	Random
CPSO-S	3	PCX GBest	WFG4	250	Random
CPSO-S	3	PCX GBest	WFG5	250	Random
CPSO-S	3	PCX GBest	WFG6	250	Random
CPSO-S	3	PCX GBest	WFG7	250	Random
CPSO-S	3	PCX GBest	WFG8	250	Random
CPSO-S	3	PCX GBest	WFG9	250	Random
CPSO-S_K	3	PCX GBest	WFG1	250	Random
CPSO-S_K	3	PCX GBest	WFG2	250	Random

Continued on next page

Table A.1: Experiments Performed (continued)

Algorithm	**Objectives**	**KTS**	**Function**	**Archive Size**	**Pruning**
CPSO-S_K	3	PCX GBest	WFG3	250	Random
CPSO-S_K	3	PCX GBest	WFG4	250	Random
CPSO-S_K	3	PCX GBest	WFG5	250	Random
CPSO-S_K	3	PCX GBest	WFG6	250	Random
CPSO-S_K	3	PCX GBest	WFG7	250	Random
CPSO-S_K	3	PCX GBest	WFG8	250	Random
CPSO-S_K	3	PCX GBest	WFG9	250	Random
CPSO-H_K	3	PCX GBest	WFG1	250	Random
CPSO-H_K	3	PCX GBest	WFG2	250	Random
CPSO-H_K	3	PCX GBest	WFG3	250	Random
CPSO-H_K	3	PCX GBest	WFG4	250	Random
CPSO-H_K	3	PCX GBest	WFG5	250	Random
CPSO-H_K	3	PCX GBest	WFG6	250	Random
CPSO-H_K	3	PCX GBest	WFG7	250	Random
CPSO-H_K	3	PCX GBest	WFG8	250	Random
CPSO-H_K	3	PCX GBest	WFG9	250	Random
PSO	3	PCX Archive	WFG1	250	Random

Continued on next page

Table A.1: Experiments Performed (continued)

Algorithm	**Objectives**	**KTS**	**Function**	**Archive Size**	**Pruning**
PSO	3	PCX Archive	WFG2	250	Random
PSO	3	PCX Archive	WFG3	250	Random
PSO	3	PCX Archive	WFG4	250	Random
PSO	3	PCX Archive	WFG5	250	Random
PSO	3	PCX Archive	WFG6	250	Random
PSO	3	PCX Archive	WFG7	250	Random
PSO	3	PCX Archive	WFG8	250	Random
PSO	3	PCX Archive	WFG9	250	Random
CPSO-S	3	PCX Archive	WFG1	250	Random
CPSO-S	3	PCX Archive	WFG2	250	Random
CPSO-S	3	PCX Archive	WFG3	250	Random
CPSO-S	3	PCX Archive	WFG4	250	Random
CPSO-S	3	PCX Archive	WFG5	250	Random
CPSO-S	3	PCX Archive	WFG6	250	Random
CPSO-S	3	PCX Archive	WFG7	250	Random
CPSO-S	3	PCX Archive	WFG8	250	Random
CPSO-S	3	PCX Archive	WFG9	250	Random

Continued on next page

Table A.1: Experiments Performed (continued)

Algorithm	**Objectives**	**KTS**	**Function**	**Archive Size**	**Pruning**
CPSO-S_K	3	PCX Archive	WFG1	250	Random
CPSO-S_K	3	PCX Archive	WFG2	250	Random
CPSO-S_K	3	PCX Archive	WFG3	250	Random
CPSO-S_K	3	PCX Archive	WFG4	250	Random
CPSO-S_K	3	PCX Archive	WFG5	250	Random
CPSO-S_K	3	PCX Archive	WFG6	250	Random
CPSO-S_K	3	PCX Archive	WFG7	250	Random
CPSO-S_K	3	PCX Archive	WFG8	250	Random
CPSO-S_K	3	PCX Archive	WFG9	250	Random
CPSO-H_K	3	PCX Archive	WFG1	250	Random
CPSO-H_K	3	PCX Archive	WFG2	250	Random
CPSO-H_K	3	PCX Archive	WFG3	250	Random
CPSO-H_K	3	PCX Archive	WFG4	250	Random
CPSO-H_K	3	PCX Archive	WFG5	250	Random
CPSO-H_K	3	PCX Archive	WFG6	250	Random
CPSO-H_K	3	PCX Archive	WFG7	250	Random
CPSO-H_K	3	PCX Archive	WFG8	250	Random

Continued on next page

Table A.1: Experiments Performed (continued)

Algorithm	**Objectives**	**KTS**	**Function**	**Archive Size**	**Pruning**
CPSO-H_K	3	PCX Archive	WFG9	250	Random
oMOPSO	3	N/A	WFG1	250	N/A
oMOPSO	3	N/A	WFG2	250	N/A
oMOPSO	3	N/A	WFG3	250	N/A
oMOPSO	3	N/A	WFG4	250	N/A
oMOPSO	3	N/A	WFG5	250	N/A
oMOPSO	3	N/A	WFG6	250	N/A
oMOPSO	3	N/A	WFG7	250	N/A
oMOPSO	3	N/A	WFG8	250	N/A
oMOPSO	3	N/A	WFG9	250	N/A
SMPSO	3	N/A	WFG1	250	N/A
SMPSO	3	N/A	WFG2	250	N/A
SMPSO	3	N/A	WFG3	250	N/A
SMPSO	3	N/A	WFG4	250	N/A
SMPSO	3	N/A	WFG5	250	N/A
SMPSO	3	N/A	WFG6	250	N/A
SMPSO	3	N/A	WFG7	250	N/A

Continued on next page

Table A.1: Experiments Performed (continued)

Algorithm	**Objectives**	**KTS**	**Function**	**Archive Size**	**Pruning**
SMPSO	3	N/A	WFG8	250	N/A
SMPSO	3	N/A	WFG9	250	N/A
CPSO-S	5	PCX GBest	WFG1	250	Random
CPSO-S	5	PCX GBest	WFG2	250	Random
CPSO-S	5	PCX GBest	WFG3	250	Random
CPSO-S	5	PCX GBest	WFG4	250	Random
CPSO-S	5	PCX GBest	WFG5	250	Random
CPSO-S	5	PCX GBest	WFG6	250	Random
CPSO-S	5	PCX GBest	WFG7	250	Random
CPSO-S	5	PCX GBest	WFG8	250	Random
CPSO-S	5	PCX GBest	WFG9	250	Random
CPSO-S_K	5	PCX GBest	WFG1	250	Random
CPSO-S_K	5	PCX GBest	WFG2	250	Random
CPSO-S_K	5	PCX GBest	WFG3	250	Random
CPSO-S_K	5	PCX GBest	WFG4	250	Random
CPSO-S_K	5	PCX GBest	WFG5	250	Random
CPSO-S_K	5	PCX GBest	WFG6	250	Random

Continued on next page

Table A.1: Experiments Performed (continued)

Algorithm	**Objectives**	**KTS**	**Function**	**Archive Size**	**Pruning**
CPSO-S_K	5	PCX GBest	WFG7	250	Random
CPSO-S_K	5	PCX GBest	WFG8	250	Random
CPSO-S_K	5	PCX GBest	WFG9	250	Random
oMOPSO	5	N/A	WFG1	250	N/A
oMOPSO	5	N/A	WFG2	250	N/A
oMOPSO	5	N/A	WFG5	250	N/A
oMOPSO	5	N/A	WFG4	250	N/A
oMOPSO	5	N/A	WFG5	250	N/A
oMOPSO	5	N/A	WFG6	250	N/A
oMOPSO	5	N/A	WFG7	250	N/A
oMOPSO	5	N/A	WFG8	250	N/A
oMOPSO	5	N/A	WFG9	250	N/A
SMPSO	5	N/A	WFG1	250	N/A
SMPSO	5	N/A	WFG2	250	N/A
SMPSO	5	N/A	WFG5	250	N/A
SMPSO	5	N/A	WFG4	250	N/A
SMPSO	5	N/A	WFG5	250	N/A

Continued on next page

Table A.1: Experiments Performed (continued)

Algorithm	**Objectives**	**KTS**	**Function**	**Archive Size**	**Pruning**
SMPSO	5	N/A	WFG6	250	N/A
SMPSO	5	N/A	WFG7	250	N/A
SMPSO	5	N/A	WFG8	250	N/A
SMPSO	5	N/A	WFG9	250	N/A