The Heterogeneous Meta-hyper-heuristic: From Low Level Heuristics to Low Level Meta-heuristics

by

Jacomine Grobler

Submitted in partial fulfillment of the requirements for the degree
Doctor of Philosophy in Engineering (Industrial Engineering)
in the Faculty of Engineering, Built Environment and Information Technology
University of Pretoria, Pretoria

February 2015

Publication data:

Jacomine Grobler. The Heterogeneous Meta-hyper-heuristic: From Low Level Heuristics to Low Level Meta-heuristics. PhD dissertation, University of Pretoria, Department of Industrial and Systems Engineering, Pretoria, South Africa, February 2015.

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

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

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

The Heterogeneous Meta-hyper-heuristic: From Low Level Heuristics to Low Level Meta-heuristics

by

Jacomine Grobler
E-mail: jacomine.grobler@gmail.com

Abstract

Meta-heuristics have already been used extensively for the successful solution of a wide range of real world problems. A few industrial engineering examples include inventory optimization, production scheduling, and vehicle routing, all areas which have a significant impact on the economic success of society. Unfortunately, it is not always easy to predict which meta-heuristic from the multitude of algorithms available, will be best to address a specific problem. Furthermore, many algorithm development options exist with regards to operator selection and parameter setting. Within this context, the idea of working towards a higher level of automation in algorithm design was born. Hyper-heuristics promote the design of more generally applicable search methodologies and tend to focus on performing relatively well on a large set of different types of problems.

This thesis develops a heterogeneous meta-hyper-heuristic algorithm (HMHH) for single-objective unconstrained continuous optimization problems. The algorithm development process focused on investigating the use of meta-heuristics as low level heuristics in a hyper-heuristic context. This strategy is in stark contrast to the problem-specific low level heuristics traditionally employed in a hyper-heuristic framework. Alternative low level meta-heuristics, entity-to-algorithm allocation strategies, and strategies for incorporating local search into the HMHH algorithm were evaluated to obtain an algorithm which performs well against both its constituent low level meta-heuristics and four state-of-the-art multi-method algorithms.

Finally, the impact of diversity management on the HMHH algorithm was investigated. Hyper-heuristics lend themselves to two types of diversity management, namely

solution space diversity (SSD) management and heuristic space diversity (HSD) management. The concept of heuristic space diversity was introduced and a quantitative metric was defined to measure heuristic space diversity. An empirical evaluation of various solution space diversity and heuristic space diversity intervention mechanisms showed that the systematic control of heuristic space diversity has a significant impact on hyper-heuristic performance.

Keywords: Hyper-heuristics, Multi-method algorithms, Memetic algorithms.

Supervisors : Prof. V. S. S. Yadavalli
Prof. A. P. Engelbrecht
Prof. G. Kendall
Department : Department of Industrial and Systems Engineering
Degree : Doctor of Philosophy in Engineering (Industrial Engineering)

Acknowledgements

"*Alone we can do so little; yet together we can do so much.*"
Helen Keller

There are a number of people and institutions which I would like to acknowledge for their help and support during the completion of this thesis:

- Prof. Sarma Yadavalli for always believing in me and encouraging me to do better.
- Prof. Andries P. Engelbrecht for his excellent mentorship, and financial support which enabled me to attend a large number of international conferences and workshops.
- Prof. Graham Kendall for his useful feedback, patience, encouragement, and sense of humour.
- The Department of Electrical, Electronic and Computer Engineering at the University of Pretoria and specifically Mr. Hans Grobler for providing the required computing infrastructure for this thesis.
- All my work colleagues at Denel Dynamics Pty (Ltd.) and the Council for Scientific and Industrial Research for their patience, understanding, and financial support.
- Dr. F. Peng and Prof. X. Yao for their generous provision of the population-based algorithm portfolio source code.
- Dr. J. Vrugt for his generous provision of the population-based genetic adaptive method for single objective optimization source code.
- All my friends and colleagues in and outside of the Computational Intelligence Research Group for their patience and valuable comments.
- The staff of the Department of Industrial and Systems Engineering for their encouragement and interest in my work.

- The Automated Scheduling Optimization and Planning research group at the University of Nottingham for hosting me at their university.
- Marita Engelbrecht, Melanie Saayman, Marde Helbig, Carmen Opperman, and all my other friends and family, for their friendship and encouragement during the completion of this thesis.
- Lindt and Chloe for their boundless enthusiasm and unconditional love.
- My parents for their unwavering love and support.
- My Creator for His blessing during the completion of this thesis.

Contents

List of Figures

List of Algorithms

List of Tables

Chapter 1

Introduction

Over the last five decades meta-heuristic algorithms have become established as the solution strategies of choice for a large range of optimization problems. The ability of a meta-heuristic algorithm to avoid local optima more successfully than, for example, local search algorithms, as well as its robustness and ease of implementation have contributed to the large amount of research carried out in recent years. Unfortunately, it is not always easy, or even possible, to predict which one of the many algorithms already in existence will be most suitable for solving a specific problem. This unpredictability is not only limited to different algorithms on different problem classes, but there may even be issues with respect to large variations in algorithm performance over different instances of the same problem. Furthermore, a large variety of problem dependent control parameter values, mapping mechanisms and operators need to be selected during the algorithm design process.

Within this context, the idea of working towards a higher level of automation in heuristic design was born. Hyper-heuristics [17] promote the design of more generally applicable search methodologies and tend to focus on performing relatively well on a large set of different types of problems, in contrast to specialized algorithms which focus on outperforming the state-of-the-art for a single application. Most recent hyper-heuristic algorithms consist of a high level methodology which control the selection or generation of a generic search strategy while using a set of low level heuristics as input. This strategy facilitates the automatic design of several algorithmic aspects, thus the impact

of hyper-heuristic research on recent optimization trends is significant.

Unfortunately, the simple low level heuristics often used in hyper-heuristic algorithms carry a risk of converging to a suboptimal solution. This thesis aims to address this gap by developing a meta-hyper-heuristic algorithm with meta-heuristics as low level heuristics, referred to as low level meta-heuristics in this thesis. The resulting meta-hyper-heuristic algorithm provides an excellent opportunity to investigate various design aspects associated with this type of algorithm. Alternative algorithm-to-candidate solution allocation strategies, strategies for utilizing local search in meta-hyper-heuristics, and solution and heuristic space diversity management strategies can be easily investigated. A number of these, and other aspects, are considered in this thesis.

The first objective of this introductory chapter was to provide a rationale for the development of a meta-hyper-heuristic. The objectives and contributions of this thesis are further highlighted in Sections 1.1 and 1.2 before a brief outline of the rest of this thesis is provided in Section 1.3.

1.1 Objectives

To investigate the use of meta-heuristics as low level algorithms in a hyper-heuristic framework, the following sub-objectives have been defined:

- To develop a meta-hyper-heuristic algorithm capable of addressing the entity-to-algorithm allocation problem effectively
- To contextualize the developed meta-hyper-heuristic algorithm within existing hyper-heuristic literature
- To investigate the impact of alternative evolutionary selection strategies on heterogeneous meta-hyper-heuristic performance
- To investigate the use of local search in the heterogeneous meta-hyper-heuristic by considering various entity selection mechanisms

- To investigate the value of diversity management on meta-hyper-heuristic performance by evaluating various diversity management strategies. Diversity management was considered in both the context of solution space and heuristic space
- To benchmark the developed HMHH algorithms against both its constituent low level meta-heuristics as well as existing state-of-the-art multi-method algorithms

1.2 Contributions

The main contributions of this thesis are summarized as follows:

- The first investigation of alternative evolutionary selection strategies in a meta-hyper-heuristic framework
- The first investigation of local search strategies in conjunction with meta-heuristic-based low level heuristics in a hyper-heuristic framework
- The first investigation of the use of an adaptive local search in conjunction with meta-heuristic-based low level heuristics in a hyper-heuristic framework
- The first explicit definition of heuristic space diversity (HSD) and the first metric defined for measuring HSD
- The development of various successful heuristic space diversity management strategies
- Development of a heterogeneous-meta-hyper-heuristic (HMHH) algorithm which outperforms four state-of-the-art multi-method algorithms
- Generation of new knowledge with regard to the impact of alternative solution-to-algorithm allocation methods as well as diversity management methods on meta-hyper-heuristic performance

1.3 Thesis Outline

Chapter 2 provides an overview and analysis of the scientific literature on existing single-method and multi-method methodologies. Various research fields have already considered the algorithm selection problem which is critical to successful multi-method algorithms. Memetic computing, hyper-heuristics, adaptive operator selection, ensembles, and portfolios are also introduced and compared in this chapter.

The HMHH algorithm framework is described in detail in Chapter 3, since this algorithm forms the basis of the rest of the investigations in this thesis.

Chapter 4 describes the initial investigations performed to determine the best HMHH framework. Various selection strategies are compared, the use of local search for performance improvement is investigated, and selection of the set of low level meta-heuristics is reconsidered.

The importance of diversity management to hyper-heuristic algorithm performance is studied in Chapter 5. Diversity is considered firstly in terms of solution space diversity. Secondly, the concept of heuristic space diversity is defined and various strategies for controlling heuristic space diversity throughout the optimization run is investigated. The performance gains from having *a priori* information available with regards to the performance of constituent low level meta-heuristic algorithms on the benchmark set in question, is also investigated.

The HMHH algorithm is benchmarked in Chapter 6. Four state-of-the-art multi-method algorithms is identified for comparison purposes. The population-based algorithm portfolio (PAP) [122], the evolutionary algorithm based on self-adaptive learning population search techniques (EEA-SLPS) [177], the fitness-based area-under-curve bandit operator selection method (FAUC-Bandit) [55], and the modified population-based genetic adaptive method for single-objective optimization (AMALGAM-SO) [172] are analyzed and compared to the HMHH algorithm. The HMHH algorithm is also compared with each constituent low level meta-heuristic used in a single-method optimization context.

Chapter 7 concludes the thesis with a summary of the major findings and future research opportunities identified during the completion of this study. Finally, the benchmark problem set, tables and graphs of results, the definitions of all symbols and

acronyms used, as well as publications derived from this thesis, are described in the six appendices.

Chapter 2

Single-method and Multi-method Literature

A large number of real world problems can be described by means of an objective function, f, a vector of variables, $\boldsymbol{x}$, and a set of constraints [47]. The objective function typically represents the quantity to be minimized or maximized. The set of variables determines the value of the objective function and $f(\boldsymbol{x})$ denotes the value of the objective function f at $\boldsymbol{x}$. The set of constraints restricts the values that can be assigned to $\boldsymbol{x}$.

Numerous methodologies have been developed over the last couple of decades to solve such optimization problems. This chapter provides a review of existing single-method optimization strategies in Section 2.1. Section 2.2 describes the state-of-the-art in multi-method algorithm literature. Finally, the chapter is concluded in Section 2.3.

2.1 An overview of single-method optimization algorithms

Feoktistov [52] differentiated between optimal solution strategies and approximate methods (refer to Figure 2.1). The suitability of optimal solution strategies is highly dependent on the complexity of the problem. In complexity theory, different classes of problem complexity have been identified based on the number of steps required to complete the algorithm for a given input. In Figure 2.2 the set **P** denotes all problems which can be

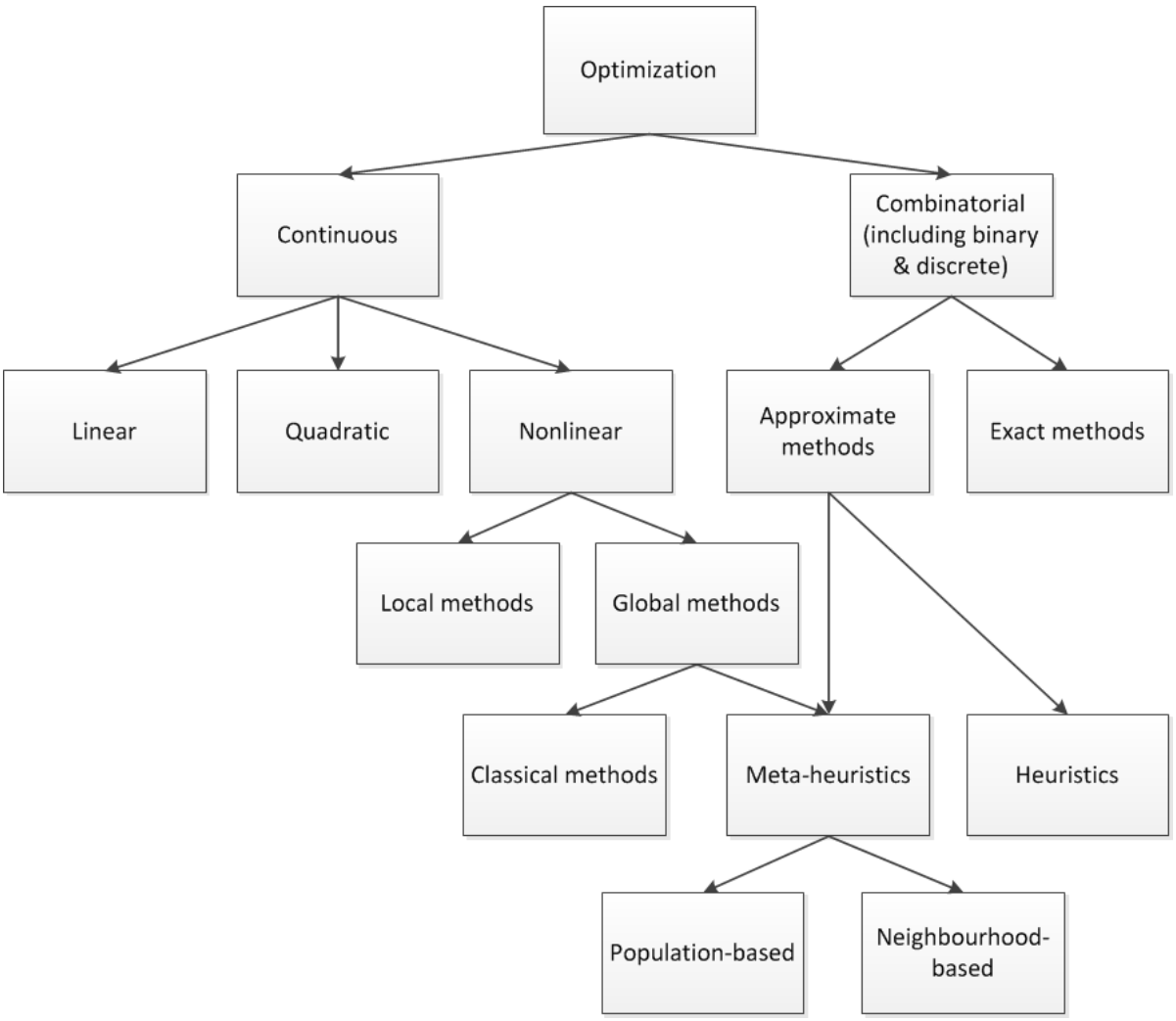

Figure 2.1: A classification of popular search methodologies [52].

solved in polynomial time. **P** is a subset of **NP**, the set of nondeterministic polynomial time problems, namely decision problems for which a solution is verifiable in polynomial time. The set **NP-Hard** contains problems which are at least as hard as the hardest problems from **NP**. Finally, the intersection of **NP** and **NP-Hard** is referred to as the set of **NP-complete** problems [160].

The practical implication of this discussion is, that apart from only very specific instances, most real world problems are not solvable within polynomial time and optimal solution strategies are then of limited use. Approximation methods, on the other hand,

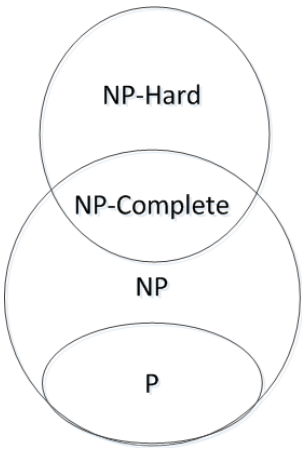

Figure 2.2: An euler diagram describing P, NP, NP-complete and NP-Hard problems [160].

are an attractive alternative. Even though the optimality of the solutions cannot be guaranteed, larger problems can be solved more efficiently.

Approximate methods can in turn be categorized into heuristics and meta-heuristics. In general, heuristic methods simply aim to obtain a "good enough" solution by selecting decision variables to obtain solutions which continuously progress towards a superior solution. The inability of heuristic methods to escape local optima have resulted in the development of meta-heuristics. These "intelligent heuristics" temporarily allow non-improving feasible moves which have a positive impact on the algorithm's ability to explore the search space [126]. A number of the more common meta-heuristics are indicated in Figure 2.3.

Neighbourhood meta-heuristics refer to those search methodologies where a single solution is transformed over time by making use of predefined neighbourhoods. Population-based meta-heuristics, on the other hand, are characterized by a population of candidate solutions which are adapted over time. The candidate solutions in an evolutionary algorithm (EA) compete for survival [19], whereas the agents in a swarm communicate and cooperate with each other by acting on the environment [47]. The rest of this section discusses a number of these popular meta-heuristic algorithms in more detail.

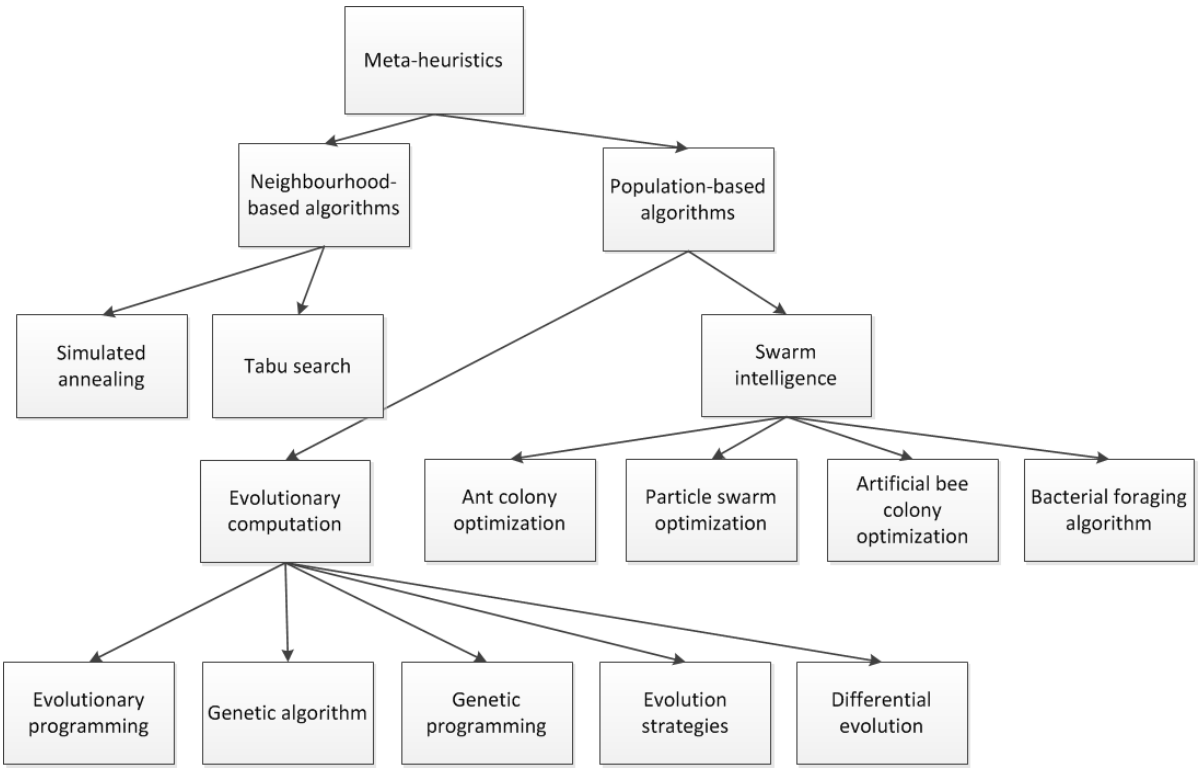

Figure 2.3: A number of common metaheuristics.

2.1.1 Tabu search

Tabu search (TS) [58, 59] is a neighbourhood-based optimization algorithm developed in the late 1980s. TS temporarily forbids moves that would return to a solution recently visited, preventing the algorithm from continuously cycling through the same solutions. This prevention is accomplished by means of a tabu list which records the most recent solutions and prevents the search from continuing with these now non-feasible moves. This list can act as both a recency-based memory (where the list classifies solutions according to the length of time they have spent in the list) and frequency-based memory (where the number of times a solution occurs has an influence). Additionally, an incumbent solution [184] is used to keep track of the best solution found thus far and certain aspiration criteria can also be defined to override the tabu list if this should become necessary. Pseudocode of the basic TS algorithm is provided in Algorithm 2.1.

$\boldsymbol{c}(t)$ denotes the candidate list of solutions at time t
$MaxTabu$ is the maximum tabu list size
Initialize an n_x-dimensional solution $\boldsymbol{x}(0)$
$t = 0$
$\boldsymbol{x}^*(0) = \boldsymbol{x}(0)$
$\boldsymbol{\tau} = \emptyset$
while *no stopping condition is satisfied* **do**
  $\boldsymbol{c}(t) = \emptyset$
  for $\boldsymbol{n} \in$ *neighbourhood*, $\boldsymbol{N}$ **do**
    if $\boldsymbol{n} \notin \boldsymbol{\tau}$ **then**
      $\boldsymbol{c}(t) = \boldsymbol{c}(t) \cup \boldsymbol{n}$
    end
  end
  Find the best admissible solution, $\boldsymbol{x}(t)$, from $\boldsymbol{c}(t)$
  if $f(\boldsymbol{x}(t)) < f(\boldsymbol{x}^*(t))$ **then**
    $\boldsymbol{x}^*(t) = \boldsymbol{x}(t)$
    $\boldsymbol{\tau} = \boldsymbol{\tau} \cup \boldsymbol{x}(t)$
    while $|\boldsymbol{\tau}| > MaxTabu$ **do**
      Remove oldest solution from tabu list
    end
  end
  $t = t + 1$
end

Algorithm 2.1: The basic TS algorithm [14].

2.1.2 Simulated annealing

Simulated annealing (SA) is an optimization algorithm based on the cooling process of liquids and solids. As a substance cools, the molecules tend to align themselves in a crystalline structure associated with the minimum energy state of the system. This is analogous to the algorithm converging to the optimal solution of an optimization problem. As the temperature of the metals decreases, the alignment of the atoms in the structure continually changes. This alignment is analogous to the fitness of the solution:

an alignment which results in a lower energy state also results in an improved solution.

The acceptance of worsening solutions over the SA run is determined by the decreasing temperature parameter. At the start of the optimization run when the temperature of the system is relatively high, there is a larger probability of accepting worsening solutions. As the optimization process nears completion, the lower temperature parameter results in a lower probability of accepting worsening solutions and subsequent increased exploitation of the good solutions found thus far. More specifically, any new solution is probabilistically accepted based on the Boltzmann–Gibbs distribution as follows:

$$P_{i_1 i_2}(t) \triangleq \begin{cases} 1 & \text{if } f(\boldsymbol{x}_{i_2}) < f(\boldsymbol{x}_{i_1}) \\ e^{\frac{f(\boldsymbol{x}_{i_2}) - f(\boldsymbol{x}_{i_1})}{a\Upsilon}} & \text{otherwise,} \end{cases} \tag{2.1}$$

where $P_{i_1 i_2}(t)$ is the probability of moving from point $\boldsymbol{x}_{i_1}$ to $\boldsymbol{x}_{i_2}$, a is a positive constant and Υ is the temperature of the system [47]. The pseudocode of the SA algorithm is presented in Algorithm 2.2.

Jain and Meeran [76] described SA as a generic technique requiring excessive computational effort due to its inability to achieve good solutions quickly. However, the hybridization of SA with other solution strategies, including genetic algorithms, has greatly improved its competitiveness.

2.1.3 Great deluge

The great deluge (GD) algorithm was first proposed by Duek in 1990 [44]. Assuming a maximization problem, the algorithm is based on the idea of finding a maximum point on a surface in the midst of a rainstorm. As the water level rises, the algorithm continues to move in such a way as not to get its feet wet, in other words, away from the rising water level. Eventually the water level covers the entire surface, but not before the algorithm has found one of the highest points or maximums of the surface.

The pseudocode of the GD algorithm is presented in Algorithm 2.3. The main advantages of GD are considered to be its simplicity and that it has only one parameter which needs to be tuned.

$\boldsymbol{c}(t)$ denotes the candidate list of solutions at time t
Initialize an n_x-dimensional solution $\boldsymbol{x}(0)$
Initialize the temperature of the system, Υ
$t = 0$
$\boldsymbol{x}^*(0) = \boldsymbol{x}(0)$
while *no stopping condition is satisfied* **do**
  Calculate the temperature of the system
  Select a neighbour n from the neighbourhood N surrounding $\boldsymbol{x}(t)$
  Calculate the acceptance probability, $P_{\boldsymbol{x}(t)n}(t)$, of moving to solution n from solution $\boldsymbol{x}(t)$ according to Equation (2.1)
  if $P_{\boldsymbol{x}(t)n}(t) > r \sim U(0,1)$ **then**
    $\boldsymbol{x}(t) = n$
  end
  if $f(\boldsymbol{x}(t)) < f(\boldsymbol{x}^*(t))$ **then**
    $\boldsymbol{x}^*(t) = \boldsymbol{x}(t)$
  end
  $t = t + 1$
end

Algorithm 2.2: The SA algorithm.

2.1.4 Genetic algorithms

The genetic algorithm (GA) was popularized by Holland [73] in the 1960s. Based on the process of genetic evolution, selection and recombination operators are applied to a population of candidate solutions with the aim of evolving better solutions over time. The rest of this subsection first describes the basic algorithm before the specific GA variant which is used in this thesis is discussed.

The basic algorithm

The original canonical GA [57] consisted of a bit string representation, proportional selection, and a cross-over mechanism as the primary method to produce new individuals. As can be seen in Algorithm 2.4, the fitness $f(\boldsymbol{x}_i(t))$, of all individuals are first calculated

Assume a minimization problem
Initialize an n_x-dimensional initial solution $\boldsymbol{x}(0)$
Select the speed of the rising water, Δ, where $\Delta > 0$
Select the initial water level, Ψ
$t = 0$
$\boldsymbol{x}^*(0) = \boldsymbol{x}(0)$
while *no stopping condition is satisfied* **do**
  Generate a new candidate solution, $\boldsymbol{c}(t)$, which is a stochastic small perturbation of $\boldsymbol{x}(t)$
  if $f(\boldsymbol{c}(t)) > \Psi$ **then**
    $\boldsymbol{x}(t) = \boldsymbol{c}(t)$
    $\Psi = \Psi + \Delta$
  end
  if $f(\boldsymbol{x}(t)) < f(\boldsymbol{x}^*(t))$ **then**
    $\boldsymbol{x}^*(t) = \boldsymbol{x}(t)$
  end
  $t = t + 1$
end

Algorithm 2.3: The GD algorithm.

before a reproduction operator such as cross-over is applied to create a new offspring population. The population of the next generation can then be selected from the previous parent and new offspring populations. This process is repeated until one of the stopping criteria is met.

A large number of variations of the original GA algorithm has since been developed. Different representation schemes, selection, cross-over, mutation, and elitism operators have been developed and tested [151]. For the purposes of this thesis, the GA used in Olorunda and Engelbrecht [116] was considered to be the most appropriate due to its previous success as constituent algorithm in a multi-method framework. This algorithm is discussed in the next section.

Let $t = 0$ be the generation counter
Initialize an n_x-dimensional population, $\boldsymbol{X}(0)$, of n_s individuals
while *no stopping condition is satisfied* **do**
  Evaluate the fitness, $f(\boldsymbol{x}_i(t))$, of each individual, $\boldsymbol{x}_i(t)$
  Perform reproduction to create offspring
  Select the new population, $\boldsymbol{X}(t+1)$
  Advance to the new generation, i.e. $t = t + 1$
end

Algorithm 2.4: The basic GA algorithm [46].

GA with floating-point representation, tournament selection, blend crossover and Gaussian mutation

Olorunda and Engelbrecht's GA used a floating-point representation, tournament selection, blend crossover [51] and Gaussian mutation in their heterogeneous cooperative algorithm implementation. Olorunda and Engelbrecht thought that the component-wise nature of the blend crossover and Gaussian mutation was well suited for fine-tuning candidate solutions.

For each individual, i, in a population two parent vectors are selected from the current population by means of tournament selection, namely $\boldsymbol{x}_{p_1}(t)$ and $\boldsymbol{x}_{p_2}(t)$, where $x_{p_kj}(t)$ denotes the j^{th} dimension (or component) of the k^{th} vector of parent individual i of generation t where $i \neq p_1 \neq p_2$. Then, for all dimensions, j, if $r \sim U(0,1) \leq p_c$, where p_c denotes the crossover probability,

$$c_{ij}(t) = (1 - \delta_j)x_{p_1 j}(t) + \delta_j x_{p_2 j}(t), \tag{2.2}$$

where $c_{ij}(t)$ denotes the j^{th} dimensions of the i^{th} offspring solution after crossover, $\delta_j = 2U(0,1) - 0.5$ and $x_{p_1 j}(t) < x_{p_2 j}(t)$. Similarly, for all dimensions, j, if $r \sim U(0,1) \leq p_m$, where p_m denotes the probability of mutation, then the j^{th} dimension of the i^{th} offspring of individual i at time t, $c'_{ij}(t)$, can be calculated as follows:

$$c'_{ij}(t) = c_{ij}(t) + \varsigma_{ij}(t)N_{ij}(0,1), \tag{2.3}$$

with $\varsigma_{ij}(t+1) = \varsigma_{ij}(t+1)e^{\tau_1 N(0,1) + \tau_2 N(0,1)}$, $\tau_1 = \frac{1}{\sqrt{2\sqrt{n_x}}}$ and $\tau_2 = \frac{1}{\sqrt{2n_x}}$, where n_x denotes

the number of dimensions. If the fitness of $\boldsymbol{c}_i'(t)$ is better than the fitness of the i^{th} individual of the original population, $\boldsymbol{x}_i(t)$, this individual is replaced by $\boldsymbol{c}_i'(t)$ [47].

2.1.5 Evolution strategies

Evolution strategies [129, 130] are based on the concept of *evolution of evolution*. The objective is to optimize the evolution process itself by defining a set of strategy parameters which influence the evolution process. By adapting the strategy parameters in parallel with the population, the optimization process is also optimized [47].

The covariance matrix adapting evolutionary strategy algorithm (CMAES) [3] is one of the most recent successful evolutionary strategy algorithms. The rest of this section describes CMAES in more detail since it is one of the algorithms that will be used throughout the rest of this thesis.

The covariance matrix adapting evolutionary strategy algorithm

The CMAES algorithm consists of four main phases, namely solution generation, selection and recombination, covariance matrix update, and step size update. During the first generation phase, a population of solutions is generated at each iteration according to a multivariate normal distribution such that

$$\boldsymbol{x}_i(t+1) \sim N(\boldsymbol{m}(t), \sigma_{CMA}^2(t))\boldsymbol{C}(t) \tag{2.4}$$

where $N(\boldsymbol{m}(t), \sigma_{CMA}^2(t))$ denotes a normal distribution with mean $\boldsymbol{m}(t)$ and standard deviation $\sigma_{CMA}(t)$. The mean of the CMAES population at time t is denoted by $\boldsymbol{m}(t)$, σ_{CMA} denotes the step size of the algorithm at time t, and $\boldsymbol{C}(t)$ is the covariance matrix at time t. After the solutions are evaluated and sorted, selection and recombination takes place by adjusting the mean of the population as follows:

$$\boldsymbol{m}(t+1) = \sum_{k=1}^{n_s} w_k \boldsymbol{x}_k \tag{2.5}$$

where w_k is the k^{th} recombination weight in the CMAES algorithm.

The covariance matrix, $\boldsymbol{C}(t)$, is then updated as:

$$\begin{aligned}\boldsymbol{C}(t+1) =&(1-c_{cov})\boldsymbol{C}(t) + \frac{c_{cov}}{\mu_{cov}} p_{c_{CMA}} p_{c_{CMA}}^T + c_{cov}\left(1-\frac{1}{\mu_{cov}}\right) \\ &\times \sum_{k=1}^{n_s} w_k \left(\frac{x_k(t+1)-\boldsymbol{m}(t)}{\sigma_{CMA}(t)}\right)\left(\frac{x_k(t+1)-\boldsymbol{m}(t)}{\sigma_{CMA}(t)}\right)^T \end{aligned} \tag{2.6}$$

where

$$\mu_{cov} \geq 1, \tag{2.7}$$

$$\mu_{cov} = \mu_{eff}, \text{ and} \tag{2.8}$$

$$c_{cov} \approx \min(\mu_{cov}, \mu_{eff}, n_x^2)/n_x^2 \tag{2.9}$$

The symbol c_{cov} denotes the learning rate for the covariance matrix update, μ_{eff} denotes the variance effective selection mass and μ_{cov} denotes the parameter which weighs between the rank-one update and rank-μ update. The rank-one update uses only the previous iteration to estimate the covariance matrix where the rank-μ update uses all previous iterations.

The CMAES algorithm makes use of cumulative step-size adaptation. A cumulative path is used which is a combination of all the steps an algorithm has made with the importance of a step decreasing exponentially with time [31]. Two evolution paths are used in the CMAES algorithm, the anisotropic evolution path, $p_{c_{CMA}}$, associated with the covariance matrix and the isotropic evolution path, p_σ, associated with the step size. $p_{c_{CMA}}$ is calculated as follows:

$$p_{c_{CMA}} = (1-c_{c_{CMA}})p_{c_{CMA}} + \sqrt{c_{c_{CMA}}(2-c_{c_{CMA}})\mu_{eff}}\left(\frac{\boldsymbol{m}(t+1)-\boldsymbol{m}(t)}{\sigma_{CMA}(t)}\right) \tag{2.10}$$

where μ_{eff} is given by

$$\mu_{eff} = \left(\sum_{k=1}^{n_s} w_k^2\right)^{-1} \tag{2.11}$$

and $c_{c_{CMA}}$ is the backward time horizon of the anisotropic evolution path.

Finally, the step size, $\sigma_{CMA}(t+1)$, is updated as follows:

$$\sigma_{CMA}(t+1) = \sigma_{CMA}(t)\exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|p_\sigma(t+1)\|}{E\|N(\boldsymbol{0},\boldsymbol{I})\|}-1\right)\right) \tag{2.12}$$

where d_σ is the damping parameter in the CMAES algorithm, $\frac{1}{c_\sigma}$ is the backward time horizon of the isotropic evolution path, p_σ:

$$p_\sigma = (1 - c_\sigma)p_\sigma + \sqrt{c_\sigma(2 - c_\sigma)\mu_{eff}}\boldsymbol{C}(t)^{-0.5}\left(\frac{\boldsymbol{m}(t+1) - \boldsymbol{m}(t)}{\sigma_{CMA}(t)}\right). \tag{2.13}$$

The pseudocode of the CMAES algorithm is provided in Algorithm 2.5.

Set the population size to n_s
Initialize all algorithm control parameters, namely, $\boldsymbol{m}(1)$, $\sigma_{CMA}(1)$, $\boldsymbol{C}(1) = \boldsymbol{I}$,$p_\sigma = 0$, $p_{c_{CMA}} = 0$
while *while no stopping conditions is satisfied* **do**
    Sample n_s new solutions using Equation (2.4)
    Calculate the fitness of the individuals in the population
    Sort the solutions from smallest to largest fitness
    Update the mean, $\boldsymbol{m}(t+1)$, using Equation (2.5)
    Update the isotropic evolution path, p_σ, using Equation (2.13)
    Update the anisotropic evolution path, $p_{c_{CMA}}$, using Equation (2.10)
    Update the covariance matrix, $\boldsymbol{C}(t+1)$, using Equation (2.6)
    Update step size, $\sigma_{CMA}(t+1)$, using isotropic path length, by means of Equation (2.12)
end

Algorithm 2.5: The CMAES algorithm [3].

CMAES significantly outperformed a number of EAs in the 2005 IEEE Congress on Evolutionary Computation Special Session on Real-parameter Optimization [2]. The algorithm has a further advantage with regards to its execution time due to a relatively small population size being required. CMAES can obtain good results with as few as 14 individuals in comparison to, for example, differential evolution where fewer than 50 individuals could lead to significant deterioration in performance [122].

2.1.6 Particle swarm optimization

Particle swarm optimization (PSO) [82] is classified as a stochastic population-based optimization technique, which was developed from a model of the flocking behaviour

of birds. Since its development, the algorithm has established itself as a competitive solution strategy for a wide range of real-world problems.

Kennedy and Eberhart [82] traced the origins of the PSO algorithm back to Reynold's "boid" simulations [133] and Heppner and Grenander's rooster effect [72]. The initial objectives of these studies and the other collective behaviour studies of the late 80s was to simulate the graceful, unpredictable choreography of collision-proof birds in a flock [45]. However, the optimization potential, of what was at that stage only a conceptual model, soon became apparent. Simplification and parameter derivation resulted in the first simplistic implementation by Kennedy and Eberhart [82] in 1995.

Since its humble beginnings, PSO has established itself as a simple and computationally efficient optimization method in both the fields of artificial intelligence and mathematical optimization. Applications range from more traditional implementations such as function optimization [82], training artificial neural networks [50, 165] and task allocation [142], to more specific applications, such as the design of aircraft wings [170] and the generation of interactive, improvised music [11], amongst many others. The rest of this section introduces the basic concepts of PSO before the actual algorithm and associated algorithm parameters and variations are discussed in more detail.

The basic algorithm

The PSO algorithm represents each candidate solution by the position of a particle in multi-dimensional hyperspace. Throughout the optimization process velocity and displacement updates are applied to each particle to move it to a different position and thus a different solution in the search space.

The velocity update is often thought to be the most critical component of the PSO algorithm since it incorporates the concepts of emergence and social intelligence. Figure 2.4 illustrates that the magnitude and direction of a particle's velocity at time t is considered to be the result of three vectors: the particle velocity vector at time $t-1$, the *pbest* position, which is a vector representation of the best solution found to date by the specific particle, and the *gbest* position, which is a vector representation of the best solution found to date by all the particles in the swarm. The *gbest* PSO [82] calculates

the velocity of particle i in dimension j at time $t+1$ using

$$v_{ij}(t+1) = wv_{ij}(t) + c_1 r_{1j}(t)[\hat{x}_{ij}(t) - x_{ij}(t)] + c_2 r_{2j}(t)[x^*_j(t) - x_{ij}(t)] \tag{2.14}$$

where $v_{ij}(t)$ represents the velocity of particle i in dimension j at time t, c_1 and c_2 are the cognitive and social acceleration constants, $\hat{x}_{ij}(t)$ and $x_{ij}(t)$ respectively denotes the personal best (*pbest*) position and the position of particle i in dimension j at time t. $x^*_j(t)$ denotes the global best (*gbest*) position in dimension j, w refers to the inertia weight, and $r_{1j}(t)$ and $r_{2j}(t)$ are sampled from a uniform random distribution, $U(0,1)$.

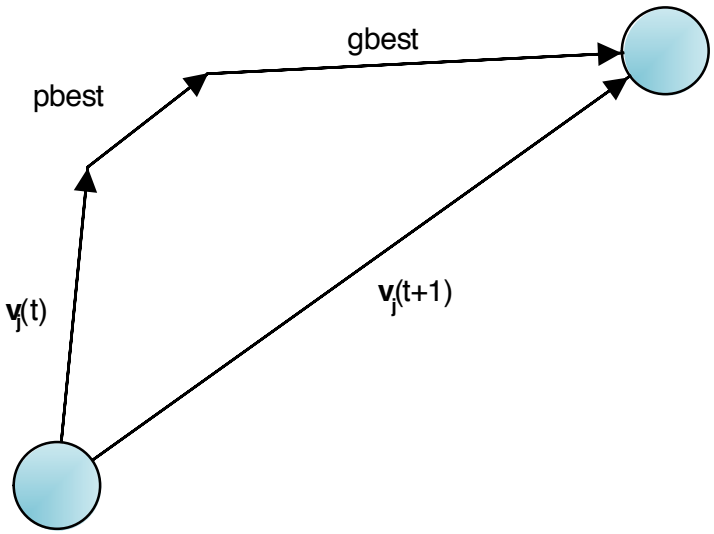

Figure 2.4: Particle velocity as resultant of three components.

The displacement of particle i at time t is simply derived from the calculation of $v_{ij}(t+1)$ in Equation (2.14) and is given as

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \tag{2.15}$$

This simultaneous movement of particles towards their own previous best solutions (*pbest*) and the best solution found by the entire swarm (*gbest*) results in the particles converging to one good solution in the search space. For the sake of completeness, pseudocode of the basic PSO algorithm is provided in Algorithm 2.6.

```
Initialize an n_x-dimensional swarm of n_s particles
t = 1
while no stopping condition is satisfied do
    for all particles i do
        if f(x_i(t)) < f(x̂_i) then
            x̂_i = x_i(t)
        end
        if f(x̂_i) < f(x*) then
            x* = x̂_i
        end
    end
    for all particles i do
        Update the particle velocity using Equation (2.14)
        Update the particle position using Equation (2.15)
    end
    t = t + 1
end
```

Algorithm 2.6: The basic *gbest* PSO algorithm [47].

Variations on the *gbest* PSO algorithm

To address the inherent limitations and requirements of the PSO algorithm, a number of variations on the *gbest* PSO algorithm have been developed over the years. However, of all the PSO variations developed, the degree of social interaction between particles has probably received the most attention. A number of alternative social network structures have been developed to explore different information exchange mechanisms between the particles within a swarm. Kennedy and Mendes [84] empirically evaluated a number of these social network structures, including the *gbest*, *lbest*, *pyramid*, *star* and Von Neumann structures. The *gbest* and Von Neumann topologies (refer to Figure 2.5) are the most important variations for the purposes of this thesis.

It is well known in PSO literature that the *gbest* PSO algorithm converges fairly quickly [83], since all particles are partially attracted to the best position found by the swarm. Depending on the problem, this relatively fast loss of diversity can result in a

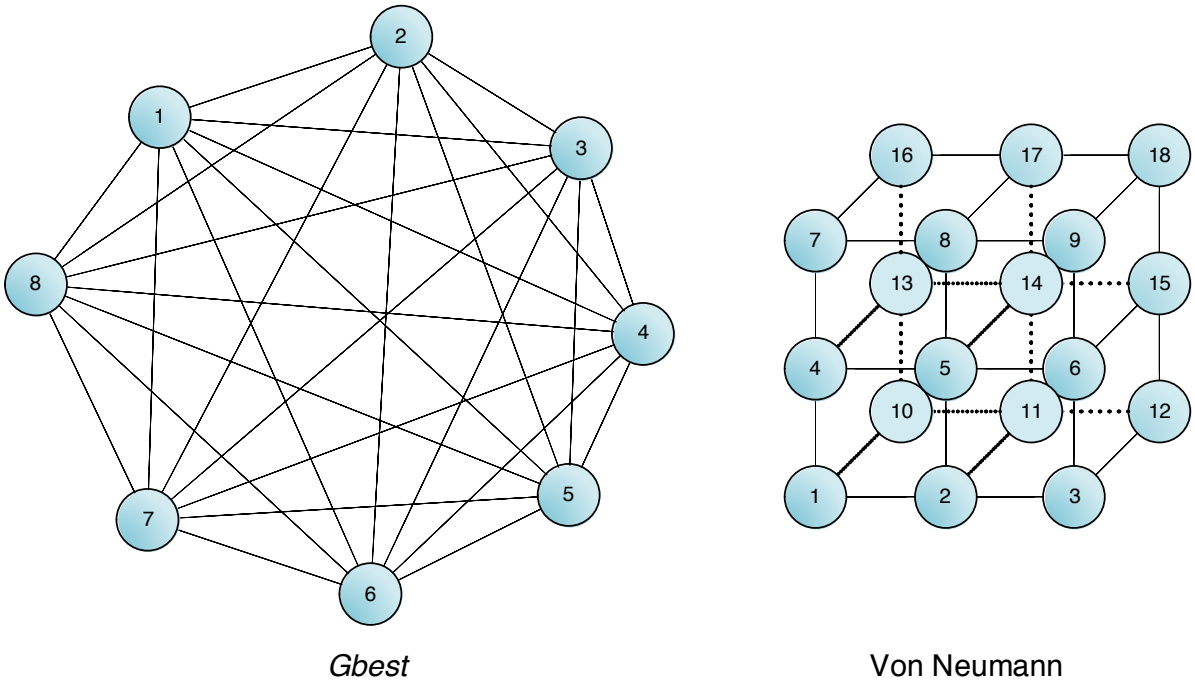

Figure 2.5: The *gbest* and Von Neumann topologies [47]. The lines between particles indicate the propagation of information through the swarm.

suboptimal solution within relatively few iterations.

The Von Neumann PSO organizes the particles into a lattice according to the particle indices. Each particle belongs to a neighbourhood consisting of its nearest neighbours in the cubic structure. Instead of being partially attracted to *gbest*, the velocity of a particle is influenced by the best solution found by the other particles in the same neighbourhood. Since these neighbourhoods overlap, information about good solutions is eventually propagated throughout the swarm, but at a much slower rate. In so doing more diversity and subsequent slower convergence is obtained, leading to significantly improved chances of finding a good solution.

Of all the PSO variations considered, the guaranteed convergence PSO (GCPSO) algorithm [166] and the barebones PSO (BBPSO) algorithm was selected for use as candidate LLMs in this thesis and are discussed throughout the rest of this section.

The guaranteed convergence PSO algorithm

Unfortunately, the basic PSO algorithm has a potentially dangerous property. The algorithm is "driven" by the fact that as a particle moves through the decision space, it is always attracted towards its *pbest* position and the swarm's *gbest* position. If any of the particles reach a position in the search space where

$$\hat{\boldsymbol{x}} = \boldsymbol{x}(t) = \boldsymbol{x}^* \tag{2.16}$$

only the momentum term ($wv_{ij}(t)$ in Equation (2.14)) remains to act as a driving force for the specific particle to continue exploring the rest of the search space. However, if the condition described in Equation (2.16) persists, it can result in the swarm stagnating on a solution which is not necessarily a local optimum [166]. The GCPSO algorithm [166] has been shown to address this problem effectively. This algorithm (Algorithm 2.7) requires that different velocity and displacement updates, defined as

$$v_{\tau j}(t+1) = -x_{\tau j}(t) + x_j^*(t) + wv_{\tau j}(t) + \rho(t)(1 - 2r_j(t)) \tag{2.17}$$

and

$$x_{\tau j}(t+1) = x_j^*(t) + wv_{\tau j}(t) + \rho(t)(1 - 2r_j(t)), \tag{2.18}$$

are applied to the global best particle, where $\rho(t)$ is a time-dependent scaling factor, $r_j(t)$ is sampled from a uniform random distribution, $U(0,1)$, and all other particles are updated by means of Equations (2.14) and (2.15). This algorithm forces the *gbest* particle into a random search around the global best position. The size of the search space is adjusted based on the number of consecutive successes or failures of the particle, where success is defined as an improvement in the objective function value. In Algorithm 2.7, the number of consecutive successes is denoted by ζ and the number of consecutive failures are denoted by η.

The barebones PSO algorithm

Kennedy [81] proposed the BBPSO algorithm by replacing the PSO velocity by random numbers sampled from a Gaussian distribution as follows:

Let $\rho(t)$ be the time-dependent scaling factor at time t
Let ζ be the number of consecutive successes experienced by the algorithm
Let η be the number of consecutive failures experienced by the algorithm
Let τ be the index of the gbest particle
Initialize an n_x-dimensional swarm of n_s particles
$t = 1$
$\rho(t) = 1$
$\zeta = 0$
$\eta = 0$
while *no stopping condition is satisfied* **do**
  for *all particles* i **do**
    if $f(\boldsymbol{x}_i(t)) < f(\hat{\boldsymbol{x}}_i)$ **then**
      $\hat{\boldsymbol{x}}_i = \boldsymbol{x}_i(t)$
    end
    if $f(\hat{\boldsymbol{x}}_i) < f(\boldsymbol{x^*})$ **then**
      $\zeta = \zeta + 1$
      $\eta = 0$
      $\boldsymbol{x}^* = \hat{\boldsymbol{x}}_i$
    else
      $\eta = \eta + 1$
      $\zeta = 0$
    end
  end
  for *all particles* $i | i \neq \tau$ **do**
    Update the particle velocity using Equation (2.14)
    Update the particle position using Equation (2.15)
  end
  Update the *gbest* particle velocity using Equation (2.17)
  Update the *gbest* particle position using Equation (2.18)
  $t = t + 1$
end

Algorithm 2.7: The GCPSO algorithm [166].

$$v_{ij}(t+1) = N\left(\frac{\hat{x}_{ij}(t) + x^*_{ij}(t)}{2}, \sigma_{BBPSO}\right) \tag{2.19}$$

with

$$\sigma_{BBPSO} = |\hat{x}_{ij}(t) - x^*_{ij}(t)| \tag{2.20}$$

$$x_{ij}(t+1) = v_{ij}(t+1) \tag{2.21}$$

where σ_{BBPSO} is the standard deviation of the population from which $v_{ij}(t+1)$ is sampled.

It has since been shown that each particle in the PSO algorithm eventually converges to a point which is the weighted average between its personal and neighbourhood best solutions [161, 164, 167]. The BBPSO thus effectively exploits this property and was considered as one of the initial candidate LLMs in this thesis.

2.1.7 Differential evolution

Originally developed from work done on Chebyshev's polynomial fitting problems, differential evolution (DE) found its roots in the genetic annealing algorithm of Storn and Price [153]. Classified as a parallel direct search method [152], DE achieved third place on benchmark problems at the first international contest on evolutionary optimization in 1996. Since then, the number of DE research papers has increased significantly every year and DE is now well-known in the evolutionary computation community as an alternative to traditional EAs. The algorithm is considered to be easy to understand, simple to implement, reliable, and fast [123]. Application areas are just as diverse, as is the case for the PSO algorithm, and range from function optimization [153] to the determination of earthquake hypocenters [136].

The basic algorithm

The success of DE can be mainly attributed to the use of difference vectors which determine the step size applied to the algorithm. Information regarding the difference between two existing solutions is, in other words, used to guide the algorithm towards better solutions [153].

More specifically, for each individual, i, in the population, a base vector, $\boldsymbol{x}_{i_1}(t)$, as well as two other vectors, $\boldsymbol{x}_{i_2}(t)$ and $\boldsymbol{x}_{i_3}(t)$, are randomly selected from the current population, where $x_{ij}(t)$ denotes the j^{th} dimension of individual i of generation t and $i \neq i_1 \neq i_2 \neq i_3$. The trial vector, $\boldsymbol{T}_i$, is then obtained through the application of a differential mutation operator, as follows:

$$T_{ij}(t) = x_{i_1 j}(t) + F(x_{i_2 j}(t) - x_{i_3 j}(t)) \tag{2.22}$$

where F is the scaling factor. Then for all dimensions, j:

$$\boldsymbol{c}_{ij}(t) = \begin{cases} T_{ij}(t), & \text{if } r \sim U(0,1) \leq p_r \text{ or } j = \nu \sim U(1,...,n_x), \\ x_{ij}(t) & \text{otherwise,} \end{cases} \tag{2.23}$$

where p_r is the probability of reproduction and $\boldsymbol{c}_i$ is the offspring solution.

An individual may only be replaced by an individual with a better fitness function value. In other words, if the fitness of $\boldsymbol{c}_i(t)$ is better than the fitness of the i^{th} individual of the original population, this individual is replaced by $\boldsymbol{c}_i(t)$ [46]. For the sake of completeness, pseudocode of the basic DE algorithm is provided in Algorithm 2.8.

The differential mutation operator in Equation (2.22) has the desirable property that it allows the step sizes of the algorithm to automatically adapt to the objective function landscape [123]. For example, before the population has converged around a specific optimum, the randomly sampled individuals could still be far apart in different areas of the search space. This allows for larger step sizes during the initial iterations of the optimization algorithm when greater exploration of the search space and the ability to escape from local optima is desirable. Later on, when all of the individuals are converging around a single optimum, smaller step sizes are automatically taken since all individuals are close to each other in the search space. This strategy allows the algorithm to more effectively exploit the area around the optimum in search of a better solution.

The number of DE research papers increases each year and as the algorithm is refined, results continue to improve and additional research opportunities become evident. As a result, several variants of DE have been defined over the years. This section describes a number of these variants according to Storn and Price's DE/x/y/z notation [153], where x refers to the method used to select the target vector, y refers to the number of difference vectors used and z denotes the crossover mechanism used.

Initialize an n_x-dimensional population of n_s individuals
$t = 1$
while *no stopping condition is satisfied* **do**
  for *all individuals* i **do**
    Randomly select an individual, i_1, from the population
    Randomly select an individual, i_2, from the population
    while $i_1 = i_2$ **do**
      Randomly select an individual, i_2, from the population
    end
    Randomly select an individual, i_3, from the population
    while $i_2 = i_3$ *or* $i_1 = i_3$ **do**
      Randomly select an individual, i_3, from the population
    end
    Calculate $T_{ij}(t)$ using Equation (2.22)
    Calculate $c_{ij}(t)$ using Equation (2.23)
  end
  for *all individuals* i **do**
    if $f(\boldsymbol{c}_i(t)) \leq f(\boldsymbol{x}_i(t))$ **then**
      $\boldsymbol{x}_i(t+1) = \boldsymbol{c}_i(t)$
    end
  end
  $t = t + 1$
end

Algorithm 2.8: The DE/rand/1/bin algorithm [123].

Alternative target vector selection mechanisms

Storn [152] identifies three different target vector selection mechanisms, i.e. DE/rand/y/z, DE/best/y/z and DE/rand-to-best/y/z (DE/R2B/y/z). These mechanisms are described as follows:

- A randomly selected population member serves as the base vector in DE/rand/y/z.
- The base vector is selected as the population member with the best fitness function,

i.e. the best individual, in DE/best/y/z. Incorporating the best individual into the equation enables faster exploitation, with a subsequent decrease in population diversity. This decrease in population diversity may, unfortunately, lead to fast convergence to a suboptimal solution.

- DE/R2B/y/z aims to address the limitations of DE/best/y/z in terms of potential premature convergence. A linearly or exponentially decreasing value ($\gamma \in (0, 1)$) is incorporated into the equation used to calculate the target vector. This ensures that more emphasis is placed on random target vector selection at the start of the optimization run when population diversity is important to explore a large area of the search space. Towards the end of the optimization run, the DE/best/y/z strategy is emphasized when convergence to the best solution is desirable. The adjusted equation to compute the target vector becomes:

$$T_{ij}(t) = \gamma x_{\tau j}(t) + (1 - \gamma) x_{i_1 j}(t) + F(x_{i_2 j}(t) - x_{i_3 j}(t)), \tag{2.24}$$

where $x_{\tau j}(t)$ denotes the j^{th} component of the best individual in the population at time t.

Self-adaptive differential evolution algorithms

Recently a number of self-adaptive DE algorithms have been developed [12, 94, 96, 124, 173, 178, 179, 180, 185]. Three of these algorithms are described throughout the rest of this section due to their good performance versus the basic DE algorithms discussed above and their potential for use as candidate LLMs in this thesis.

The differential evolution algorithm with neighbourhood search (NSDE) [178] uses a self-adaptive scale factor, F, which allows the algorithm to automatically adjust between sampling values of the scale parameter from either Gaussian or Cauchy distributions. The Gaussian distribution promotes small step sizes, while the Cauchy distribution promotes larger step sizes. The scale factor of individual i, F_i, is calculated as:

$$F_i = \begin{cases} N_i(0.5, 0.5), & \text{if } U_i(0, 1) < p_f, \\ \vartheta & \text{otherwise.} \end{cases} \tag{2.25}$$

where $N_i(0.5, 0.5)$ denotes a Gaussian random number with mean 0.5 and standard deviation 0.5, $U_i(0, 1)$ denotes a uniform random number between 0 and 1, and ϑ denotes a Cauchy random variable with a scale parameter of 1. The probability, p_f, was set to 0.5. The rest of the NSDE algorithm is the same as the basic DE algorithm described above.

The self-adaptive differential evolution (SaDE) algorithm [124] was the first DE algorithm which incorporated two different mutation strategies into the same algorithm. The proposed self-adaptive mutation strategy attempts to solve the dilemma that DE mutation strategies are often highly dependent on the problem under consideration [179]. The target vector, $\boldsymbol{T}_i(t)$, is determined as:

$$\boldsymbol{T}_i(t) = \begin{cases} Eq.(2.22), & \text{if } U_i(0,1) < p_T, \\ Eq.(2.24) & \text{otherwise.} \end{cases} \tag{2.26}$$

where the probability, p_T, is calculated as follows:

$$p_T = \frac{ns_1(ns_2 + nf_2)}{ns_2(ns_1 + nf_1) + ns_1(ns_2 + nf_2)} \tag{2.27}$$

and ns_1 denotes the number of offspring generated by Equation (2.22) which outperformed their associated parent solutions and successfully entered the next generation. Similarly, ns_2 denotes the number of successful offspring generated by Equation (2.24). Finally, nf_1 and nf_2 denote the number of offspring generated by Equations (2.22) and (2.24) which were discarded before reaching the next generation.

The SaDE algorithm samples F_i from a Gaussian distribution as follows:

$$F_i = N_i(0.5, 0.3) \tag{2.28}$$

where $N_i(0.5, 0.3)$ denotes a Gaussian random number with mean 0.5 and standard deviation 0.3. A probability of reproduction, p_{ri}, is generated for each individual i according to:

$$p_{ri} = N_i(p_{r_\mu}, 0.1) \tag{2.29}$$

where p_{r_μ}, the mean of the Normal distribution from which p_{ri} is sampled, is calculated

as follows:

$$p_{r_\mu} = \frac{1}{|\boldsymbol{p}_{r_{succ}}|} \sum_{l=1}^{|\boldsymbol{p}_{r_{succ}}|} q_l \tag{2.30}$$

The set $\boldsymbol{p}_{r_{succ}}$ store the p_{ri} values associated with all successful offspring which outperformed their parent solutions, denoted by q_l.

The self-adaptive DE algorithm with neighbourhood search (SaNSDE) [179] combines the best features of NSDE [178] and SaDE [124]. SaNSDE attempts to improve algorithm performance by means of self-adaptation of two candidate mutation strategies, namely the strategies used in the DE/rand/y/z algorithm and the DE/R2B/y/z algorithm. A self-adaptive scale factor, F, and self-adaptive probability of reproduction, p_r, is also utilized. The self-adaptive mutation strategy is used "as-is" from the SaDE algorithm and is described in Equation (2.26). The self-adaptation mechanism applied to the scale factor, F, is similar to the NSDE mechanism described in Equation (2.25), apart from the probability, p_f, which is self-adapted in the same way as p_T in Equation (2.27). The only difference from Equation (2.27) is that the number of successful offspring is considered per scaling factor update.

The probability of reproduction, p_{ri}, of the SaNSDE algorithm is also self-adapted in a similar fashion to the SaDE algorithm. The only difference is that the list of successful p_{ri} values is weighted according to the corresponding improvements in fitness value. The mean of the Gaussian distribution in Equation (2.29) is calculated as follows:

$$p_{r_\mu} = \sum_{l=1}^{|\boldsymbol{p}_{r_{succ}}|} w_l q_l \tag{2.31}$$

The set $\boldsymbol{p}_{r_{succ}}$ store the p_{ri} values associated with all successful offspring which outperformed their parent solutions, denoted by q_l, as well as their resulting improvement in fitness value, $f_{\delta l}$, where

$$f_{\delta l} = f(\boldsymbol{x}_l)(t) - f(\boldsymbol{c}_l)(t+1) \tag{2.32}$$

The l^{th} weight, w_l, is calculated as

$$w_l = \frac{f_{\delta l}}{\sum_{l=1}^{|\boldsymbol{f}_\delta|} f_{\delta l}} \tag{2.33}$$

SaNSDE was shown to outperform both SaDE and NSDE [179] and is considered a highly successful DE algorithm due to the trade-off between small and large step sizes and control parameters that are self-adapted based on statistical learning experience during evolution instead of just heuristic updating rules. The pseudocode for the SaNSDE algorithm is provided in Algorithm 2.9.

2.2 An overview of multi-method optimization algorithms

As can be seen from the previous section, a number of single-method optimization algorithms have already been developed to address numerous real world optimization problems. Unfortunately, it is not always easy, or even possible, to predict which one of the many algorithms already in existence will be the most suitable for solving a specific problem. Furthermore, this uncertainty with regards to which algorithm to use for a given problem, is not only limited to different algorithms on different problem classes, but there may even be issues with respect to large variations in algorithm performance over different instances of the same problem [149].

To address the issue of selecting an appropriate algorithm for a problem to be solved, Rice [134] formulated the algorithm selection problem in 1976. The problem is formulated as follows:

For a given problem instance $p \in \boldsymbol{P}$*, with features* $\boldsymbol{f}(p)$*, find the selection mapping* $\boldsymbol{S}(\boldsymbol{f}(p))$ *into algorithm space* $\boldsymbol{H}$*, such that the selected algorithm* $m \in \boldsymbol{H}$ *maximizes the performance mapping of* $y(m(p)) \in \boldsymbol{Y}$*.*

According to the No Free Lunch Theorem [176], "*... for any algorithm, any elevated performance over one class of problems is exactly paid for in performance over another class.*" Therefore, no universally best algorithm exists which can outperform all other algorithms on a broad problem domain. The ideal approach is to use specific problem characteristics in determining which algorithm is most suited to the application under consideration.

Within this context, the simultaneous use of more than one algorithm for solving optimization problems became an attractive alternative. A multi-method algorithm can

Initialize an n_x-dimensional population of n_s individuals
$t = 1$
while *no stopping condition is satisfied* **do**
  for *all individuals* i **do**
    Generate a scale factor, F_i, using Equation (2.28)
    Generate a probability of reproduction, p_{ri}, using Equations (2.29) and (2.31)
    Randomly select an individual, i_1, from the population
    Randomly select an individual, i_2, from the population
    while $i_1 = i_2$ **do**
      Randomly select an individual, i_2, from the population
    end
    Randomly select an individual, i_3, from the population
    while $i_2 = i_3$ *or* $i_1 = i_3$ **do**
      Randomly select an individual, i_3, from the population
    end
    Calculate $T_{ij}(t)$ using Equation (2.26)
    Calculate $c_{ij}(t)$ using Equation (2.23)
  end
  for *all individuals* i **do**
    if $f(\boldsymbol{c}_i(t)) \leq f(\boldsymbol{x}_i(t))$ **then**
      $\boldsymbol{x}_i(t+1) = \boldsymbol{c}_i(t)$
    end
  end
  $t = t + 1$
  Update the probabilities p_T, p_f, and p_{r_μ}.
end

Algorithm 2.9: The SaNSDE algorithm [179].

be described as consisting of one or more entities, where an entity represents a candidate solution which is adapted over time, a set of available algorithms or operators, referred to as constituent algorithms, and a high level strategy responsible for allocating the entities to the most suitable algorithms for optimization. The main idea of a multi-method

algorithm is that the simultaneous use of more than one search algorithm during the optimization process allows the algorithms to exploit each other's strengths while also compensating for inherent weaknesses.

EAs, in general, focus on evolving a population of candidate solutions or entities over time in order to converge to a higher quality solution. In most cases, a single algorithm, with its own unique operators, is assigned to all entities and this algorithm is used in isolation to evolve the population. As soon as multiple algorithms are used simultaneously to optimize a common population of entities, the allocation of entities to algorithms needs to be considered. Two main approaches of entity-to-algorithm allocation can be identified from the literature. Static entity-algorithm allocation assigns entities to algorithms at the start of the optimization run and this allocation remains static throughout the rest of the run. Dynamic entity-algorithm allocation continuously updates the allocation of entities to algorithms throughout the optimization run.

Over the years, multi-method algorithms have started appearing in various different domains. Examples include memetic computation [30], algorithm portfolios [62], algorithm ensembles [42], hyper-heuristics [20], and adaptive operator selection methods [40]. The rest of this chapter discusses each of these fields in more detail.

2.2.1 Memetic computation

Meta-heuristics are well known for their robustness and ability to avoid local optima [60]. However, room for improvement exists with regard to a meta-heuristic's ability to successfully exploit good solutions [88]. The hybridization of a meta-heuristic algorithm with a refinement method can be very useful to balance the trade-off between exploration and exploitation. The advantages of a meta-heuristic algorithm, namely generality, robustness, and global search efficiency, can be combined effectively with the ability of a local search algorithm to explore application-specific problem structures and converge rapidly towards a local minimum [112].

Memetic algorithms (MAs), the algorithmic pairing of a population-based search method with one or more refinement methods [30], can be considered the first multi-method techniques applied in the field of computational intelligence [71]. The ability of global optimization algorithms to quickly identify promising areas of the search space

is combined with local search algorithms which are able to refine good solutions more efficiently. The wider area of memetic computing is concerned with algorithms composed of heterogeneous operators (memes) for solving optimization problems [26]. Within the context of EAs, Chen *et al.* [30] proposed a classification of simple hybrids and adaptive hybrids. Ong *et al.* [117] further investigated the mechanisms of coordination among algorithmic components in MC approaches and proposed the following classification of adaptive coordinating mechanisms:

- Hyper-heuristics [17], which Ong *et al.* [117] defined as a heuristic scheme where a set of prearranged rules determines the activation of each component.
- Meta-Lamarckian learning [86, 90, 111] which allows an adaptive algorithm choice from a set of available local search algorithms at different decision points.
- Self-adaptive and co-evolutionary MAs [88, 147, 148, 182], where the rules coordinating the memes are evolved in parallel with the candidate solutions of the optimization algorithm.
- Fitness diversity adaptive algorithms [24, 25, 110, 156, 159], where the population diversity is estimated through the diversity of the fitness values and this estimate is used to balance the exploration-exploitation trade-off of the search.

A number of key issues that need to be considered during MA design are defined by [88] and [30]. Firstly, the number of individuals which should undergo refinement should be determined. The first MAs recommended that all individuals in the population should be refined at each iteration of the MA [104]. Due to various constraints such as computational budget and the need to maintain a suitable level of diversity in the population, this strategy is not always desirable [71].

Various other MA design issues such as the intensity of refinement [154], type of local search algorithm employed, integration of local search with existing evolutionary operators [89], and Lamarckian versus Baldwinian learning [175], could also be considered. These strategies, however, fall outside of the scope of this thesis and are not explicitly investigated.

2.2.2 Ensemble and portfolio algorithms

The idea of combining different algorithms in an ensemble originated from the field of neural networks. Hansen *et al.* [68] introduced an ensemble neural network consisting of various neural network models which are averaged. The idea quickly migrated to the evolutionary computation field where different operators were combined to improve algorithm performance and robustness. One of the first of such examples is Yao *et al's* improved fast EP [181] which mixed both Cauchy and Gaussian mutation operators. In the DE domain, the intelligent selection of mutation operators and control parameters during optimization is considered in [96], [186], and [163]. The self-adaptive DE algorithm of Qin and Suganthan [124] which makes use of different DE learning strategies which are weighted based on previous algorithm success, is another example. Various heterogeneous PSO algorithms have also been developed [48, 49, 107, 108, 150]. For the purposes of this thesis, however, the ensembles and portfolios consisting of operators from different algorithms such as PSO, DE, GA, etc. in a multi-method framework, are of significantly more interest. The rest of this section discusses various such heterogeneous portfolio and ensemble algorithms.

Vrugt *et al.*'s highly successful population-based genetic adaptive method for single objective optimization (AMALGAM-SO) [172] was developed after the success of the multi-objective AMALGAM algorithm. This algorithm is one of the few examples of an algorithm which continually updates the allocation of algorithms to entities during the optimization run. AMALGAM-SO employs a self-adaptive learning strategy to determine the percentage of candidate solutions in a common population to be allocated to each of three EAs. A restart strategy is used to update the percentages based on algorithm performance. Refer to Section 6.1.3 for more details. This technique performed well when compared to a number of single method EAs on the 2005 IEEE Congress of Evolutionary Computation benchmark problem set [155]. Closer inspection of the algorithm uncovers a large bias towards CMAES. Between 80% and 90% of the entities in the initial population is allocated to CMAES and during the optimization run a minimum of 25% of entities are allocated to CMAES at all times. Since it is well-known that CMAES is the best choice among the available algorithms for solving the CEC2005 problems, the algorithm is in effect being unfairly assisted. The question of how well the algorithm

will perform without this assistance, remains to be answered.

Another algorithm which inspired the development of the heterogeneous-meta-hyper-heuristic (HMHH) algorithm is the heterogenous cooperative algorithm of Olorunda and Engelbrecht [116]. Traditionally, cooperative algorithms are multi-population techniques where problem variables are distributed over a number of subpopulations to be optimized separately. A solution is constructed by combining the best solution obtained by each subpopulation. The heterogeneous cooperative algorithm makes use of different EAs to update each of the subpopulations, thereby combining the strengths and weaknesses of various optimization strategies within the same algorithm. Promising results in terms of robustness and consistent performance were obtained when compared to other single-method EAs.

Peng *et al.* [122] developed the population-based algorithm portfolio (PAP). This algorithm is based on the principle of multiple subpopulations each assigned to one algorithm from a portfolio of available algorithms. At pre-specified time intervals, entities are migrated between subpopulations to ensure effective information sharing between the different optimization algorithms. A pairwise metric was also proposed which can be used to determine the risk associated with an algorithm failing to solve the problem in question. It should be noted, however, that PAP makes use of a static entity-to-algorithm allocation strategy. Thus, the algorithm is stuck with the initial allocation throughout the rest of the optimization run. This implies that when a different entity-to-algorithm allocation is required at a different stage of the optimization run, no update to the entity-to-algorithm allocation can be made. The influence this has on algorithm performance, remains to be determined.

Tang *et al.* [156] recently developed an extended version of PAP, namely PAP based on an estimated performance matrix (EPM-PAP) which contains a novel constituent algorithm selection module. EPM-PAP was shown to outperform a number of single EAs. It should be noted that the motivation of PAP is to select constituent algorithms so as to achieve good overall performance on a set of problem instances in contrast to, for example, AMALGAM-SO which attempts to obtain the best possible solution on a specific problem instance.

The EA based on self-adaptive learning population search techniques (EEA-SLPS) [177]

was inspired by PAP. Similar to PAP [122], EEA-SLPS consists of entities divided into subpopulations. These subpopulations are adapted in parallel by an assigned constituent algorithm, where one constituent algorithm is used per subpopulation. Each entity only has access to other entities within the same subpopulation in order to prevent the same genetic material from being adapted repeatedly. However, an information exchange mechanism is used to ensure that each constituent algorithm benefits from the learning of the other algorithms. A strong focus of Xue *et al*'s work was the investigation of alternative information exchange mechanisms and their impact on portfolio performance. Eighteen mechanisms were evaluated and the best strategy was identified as replacing the worst individual of each subpopulation by the current best individual of the entire ensemble. This replacement was found to work best at each iteration. This algorithm also makes use of a static entity-to-algorithm allocation.

Recently, Yuen *et al.* claimed superior performance of their multiple EA (Multi-EA) [183] when compared to PAP and AMALGAM-SO. The algorithm makes use of a linear regression model and a bootstrapping mechanism to predict at each iteration which algorithm would perform the best at a common future point. Their improved performance claim has, however, been questioned [66]. The referenced study pointed out the lack of information exchange mechanism of the Multi-EA which prevented the constituent algorithms from learning from each other. A comparison between Multi-EA and EEA-SLPS further highlighted the poor performance of Multi-EA. Three unchallenged algorithms thus remain, namely EEA-SLPS, PAP, and AMALGAM-SO.

2.2.3 Adaptive operator selection strategies

Adaptive operator selection (AOS) strategies autonomously select between different operators such as, for example, various mutation and crossover operators, in an online manner based on the recent performance of the strategies within an optimization process [91]. AOS techniques consist of two main components, namely a credit assignment mechanism and an operator selection mechanism. As can be seen in Figure 2.6 an operator is applied in the context of an EA. The impact of the applied operator with regard to the effect on the objective function value is then passed to the credit assignment mechanism. This mechanism determines the award that should be assigned to the oper-

ator and updates the credit registry which stores the current credit associated with each operator. The operator selection method then uses the credit registry as input to select an operator, which is then used by the EA to update the relevant candidate solutions.

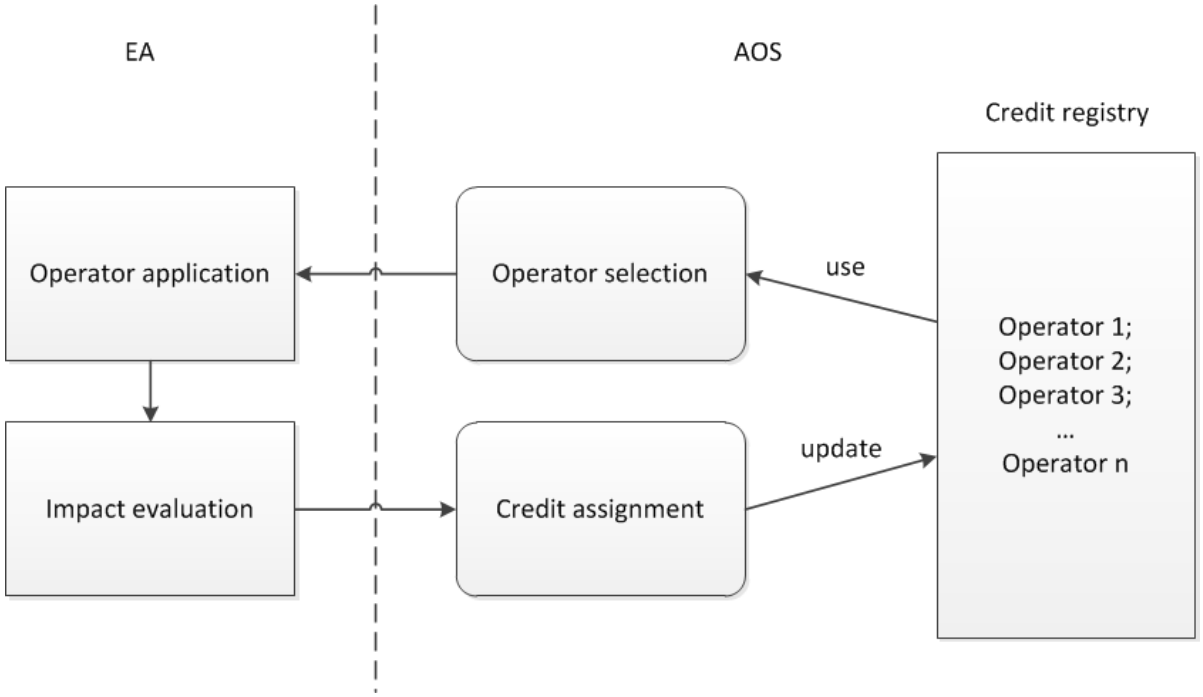

Figure 2.6: A generic adaptive operator selection strategy [97].

Various credit assignment mechanisms have been developed over the last couple of years. The reward amount for each operator has already been determined by considering for each operator:

- The best solution of the current population generated by the operator under consideration [40]
- The best solution of the parent population [63]
- The best solution of the ancestors of the population [92, 162]
- The median solution of the population [77]
- The average of the recent fitness improvements [174]

- The maximal fitness improvement achieved recently [54], which emphasizes the reward of rare, large improvements in fitness
- Both diversity and fitness improvement [97], which aims to reduce the chances of getting stuck in a local optimum
- A rank-based credit assignment scheme such as the area-under-curve and sum-of-ranks schemes of Fialho *et al.* [56] and the fitness-rate-rank-based scheme of Li *et al.* [91]

A number of operator selection mechanisms have also been developed, namely:

- Probability matching [61], where the credit assignment mechanism is used as input to a roulette-wheel-like process
- Adaptive pursuit [158], which incorporates a winner takes all strategy into the roulette-wheel process to increase the chances of selecting the best operator
- Various bandit-based methods [53, 91], inspired from the multi-armed bandit game theory problem [8]

Further analysis of the literature indicates that the bandit-based methods are the most successful operator selection strategies. With regard to credit assignment, the rank-based strategies provide the most robust solutions. The fitness-based area-under-curve bandit (FAUC-Bandit) algorithm [54] is thus identified as one of the most promising AOS algorithms for further investigation.

2.2.4 Hyper-heuristics

Burke *et al.* [18] defined a hyper-heuristic as "*a search method or learning mechanism for selecting or generating heuristics to solve computational search problems*". Hyper-heuristics promote the design of more generally applicable search methodologies and tend to focus on performing relatively well on a large set of different problems, in contrast to specialized algorithms which focus on outperforming the state-of-the-art for a single application. The basic idea is thus not only to obtain an appropriate solution for a

specific problem, but rather to focus on automating the development of the method used to obtain an appropriate solution. This increased generality of hyper-heuristic algorithms is a valuable attribute considering the specialist resources required for the development of advanced artificial intelligence-based algorithms as well as the problem-dependent nature of most meta-heuristic algorithm implementations.

A framework used in early hyper-heuristic research [18] to describe the elements of a hyper-heuristic algorithm is provided in Figure 2.7. The hyper-heuristic acts as a high level methodology which receives performance information through a domain barrier. This performance information drives the selection or generation of a set of domain specific low level heuristics. The purpose of the domain barrier between the low level heuristics and the hyper-heuristic is to allow for improved generality. Since the hyper-heuristic's functioning is only dependent on the domain independent fitness information obtained from the solutions generated by the low level heuristics, it is theoretically possible to re-use the hyper-heuristic as-is in another domain.

Burke *et al.* [18] provided a unified classification of recent hyper-heuristic research (Figure 2.8). Hyper-heuristics were classified according to the nature of the heuristic search space as well as the source of feedback during learning. With respect to the nature of the heuristic search space, two popular approaches, namely heuristic selection and heuristic generation, was identified. Heuristic selection focuses on combining pre-existing heuristics in one higher-level search strategy, whereas heuristic generation is concerned with generating completely new heuristics consisting of basic components or building blocks of existing heuristics. On a more detailed level, construction heuristics attempt to construct a single good candidate solution through the intelligent application of different low level heuristics. Perturbation heuristics use one low level heuristic to construct a complete solution, but repetitively apply low level heuristics in a local search approach to obtain better and better candidate solutions.

Many hyper-heuristics use information regarding the performance of previously selected low level heuristics to "learn" good combinations of low level heuristics to be applied during the optimization process. Burke *et al.*'s classification differentiates between no-learning, online learning, where the algorithm learns to adaptively solve a single instance of the optimization problem, and offline learning, where a training set of

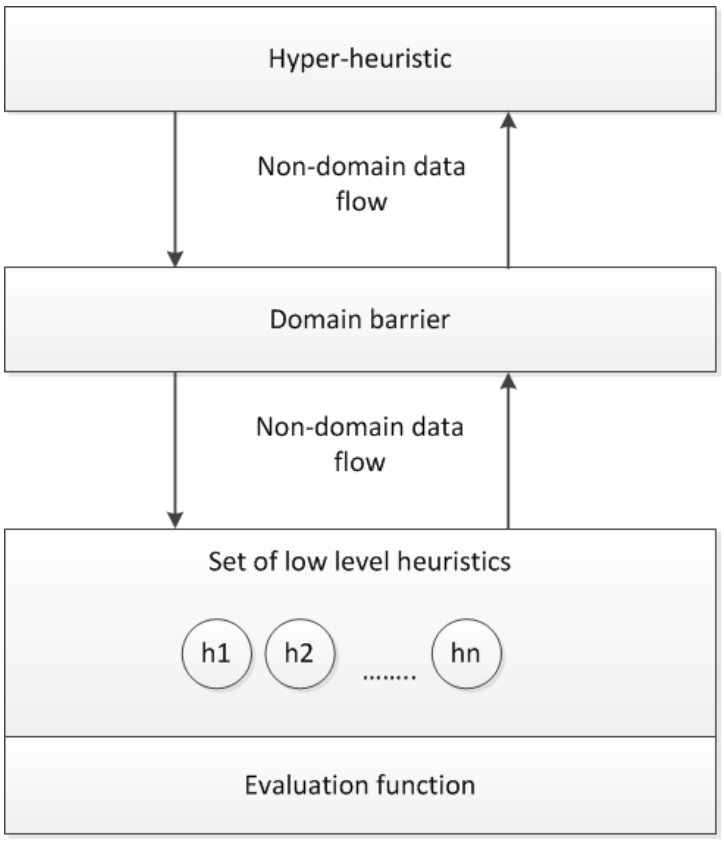

Figure 2.7: Framework of hyper-heuristic algorithms [18].

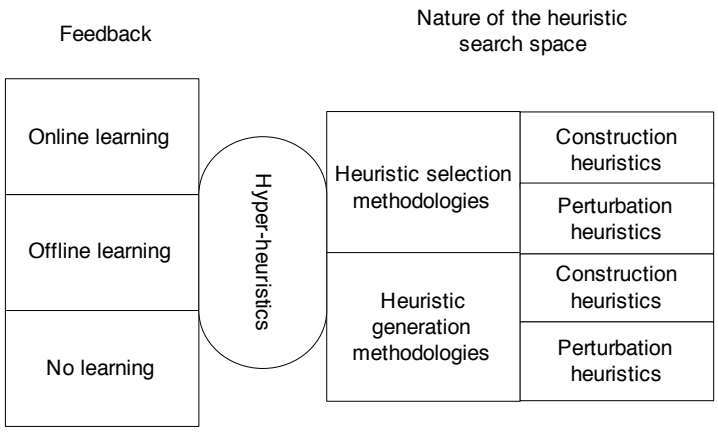

Figure 2.8: Classification of hyper-heuristic algorithms [18].

problems is used to develop a method that may generalize to unseen problem instances.

A comprehensive review of hyper-heuristic research conducted over the last two decades can be found in [17]. This thesis, however, specifically focuses on investigating the use of meta-heuristic algorithms in a hyper-heuristic framework. The high level methodology should select between a set of available perturbative meta-heuristics in contrast to generating new algorithms. The rest of this section thus focuses on only reviewing heuristic selection methodologies based on perturbative low level heuristics. A distinction is made between single-point search-based hyper-heuristics and multi-point search-based (population-based) hyper-heuristics, as well as between heuristic selection methods and move acceptance methods.

Various common heuristic selection methods have been employed in single-point-based hyper-heuristics:

- The simple random strategy [35] selects a low level heuristic randomly at each step.
- The random gradient strategy [35] applies a randomly selected heuristic repeatedly until no further improvement is obtained.
- The random permutation strategy [35] generates a random ordering of heuristics which are applied in the provided sequence at each step.
- The random permutation gradient strategy [35] only updates the random ordering of heuristics when no further improvement is obtained.
- The greedy strategy [35] applies all heuristics exhaustively before the heuristic generating the most improved solution is selected.
- The choice function strategy [35, 36, 37, 43, 95, 128] is a score-based learning approach. Each of the low level heuristics are ranked according to individual performance, performance compared to the preceding heuristic, and the elapsed time since it was last used.
- The reinforcement learning strategy [105] attempts to learn which heuristic is the best at a given decision point. If a heuristic improves a solution it is rewarded and a worsening move results in punishment of the heuristic by reducing its score.

- The reinforcement learning with TS strategy [20, 21] incorporates a dynamic tabu list of low level heuristics into the reinforcement learning heuristic selection strategy.
- The TSRoulWheel strategy [22] is a roulette-wheel selection-based mechanism which uses a tabu list.
- The peckish strategy [28, 33, 34] first reduces the number of heuristics before the greedy heuristic selection strategy is applied.
- The markov chain-based strategy [99] maintains a set of weighted edges representing probabilities of transitioning from one heuristic to another. After each selection the weights are updated based on the performance of the heuristics in a reinforcement learning scheme.

Similarly, a number of move acceptance strategies which determines if a solution generated by the algorithm selected by the heuristic selection method should be accepted into the population. These move acceptance strategies include:

- The all moves strategy [35] accepts all moves regardless of their influence on solution performance.
- The only improvements strategy [35] accepts only improving moves.
- The monte carlo-based strategy [4, 137] accepts all improving moves while non-improving moves are accepted with a certain probability. Linear and exponential probability functions as well as a formulation based on computation time and a counter of consecutive non-improvement iterations have been utilized.
- The great deluge strategy [44, 79] accepts any heuristic which is not significantly worse than a predefined level at each iteration. This level changes at a linear rate every step from an initial value to a target objective value.
- The record-to-record travel strategy [44, 80] accepts any move which is not much worse than a current solution with a fixed limit.

- The simulated annealing strategy [1, 6] always accepts improving moves while non-improving moves are accepted according to the Metropolis criteria [85].
- The late acceptance strategy [15] compares the move to the objective function L iterations earlier, where L is a user defined control parameter.
- The steepest descent late acceptance strategy [41] compares only improving moves to the objective function L iterations earlier.
- The adaptive iteration limited list-based threshold acceptance strategy [101] uses an adaptive threshold to determine whether a worsening solution should be accepted. This threshold is relaxed every time no better solutions are discovered after a fixed number of predefined iterations.

A number of population-based hyper-heuristics have also been developed based on different optimization algorithms. These include hyper-heuristics based on:

- GAs [93], DEs [9] and other EAs [32]
- Genetic programming such as Nasser *et al's.* [140] dynamic multi-armed bandit-extreme value-based selection strategy with a gene expression programming framework for automatic generation of acceptance strategies. The algorithm performed well when compared to a large number of state-of-the-art hyper-heuristics on examination timetable problems, dynamic vehicle routing problems and the hyper-heuristic competition test suite (CHeSC). The gene expression programming framework has also been successfully used to evolve both the heuristic selection and acceptance strategy [139].
- Artificial immune systems [70, 69, 146, 144, 145]
- Ant colony optimization [29, 131]
- Cooperative search methods [38]
- Agent-based approaches [100, 119]

A comprehensive review of the specific application areas and results obtained by the various combinations of selection and acceptance strategies that have been used in hyper-heuristic algorithms can be obtained in [17].

From the brief review given in this section, two observations are made: Firstly, significantly fewer population-based hyper-heuristic implementations exist when compared to the single-point search-based hyper-heuristics. Secondly, even though meta-heuristic algorithms have been used many times as a higher-level search strategy in a hyper-heuristic framework, far fewer implementations focus on the use of meta-heuristics as low level search strategies in a hyper-heuristic framework.

A single clear winning combination of heuristic selection and move acceptance strategy is difficult to identify and the selection of these appropriate hyper-heuristic strategies are very problem specific. It thus makes sense to evaluate a number of options with regards to these hyper-heuristic strategies once a new domain is explored. Simulated annealing [1, 6], late acceptance [15], and variants of threshold acceptance [101] are, however, the most promising acceptance strategies. Finally, the acceptance strategy seems to have a bigger influence in performance than the selection strategy.

2.2.5 Summary of findings from the multi-method literature review

To conclude this chapter, the differences and similarities between the various multi-method areas identified are discussed in this section. Some of the early ideas leading to hyper-heuristics came from the scheduling domain where a number of domain-specific scheduling rules were selected and sequenced based on a high-level strategy. A large portion of hyper-heuristic research is, in fact, still focused on generating and selecting a combination of low level domain-specific heuristics for solving practical problems such as bin-packing [70, 93, 135], examination timetabling [23, 78, 125], and production scheduling [114, 115, 168]. Research into AOS strategies typically focuses on the selection of alternative algorithm operators such as mutation strategies in a DE algorithm [53, 54, 63]. Algorithm ensembles [62] and portfolios [42] are concerned with dividing a function evaluation budget effectively between various meta-heuristics which make up the ensemble or portfolio. Finally, the first memetic algorithms attempted to improve the exploitation

ability of meta-heuristic algorithms [104], but today multiple memetic algorithm components or memes can be selected for use in a memetic algorithm [30]. All the areas discussed in this thesis have, however, a set of low level operators, memes, heuristics, or algorithms whose use is governed by a high level strategy which performs some sort of learning over time to decide which low level component should be used at what time. Unfortunately, an analysis of current best practice quickly makes it clear that research within these different multi-method areas is often performed in isolation with little interaction between the different research areas.

Another factor for consideration is that algorithms are not always compared on the same benchmark problem set and the selection of constituent algorithms and algorithm control parameters such as population size can also have a significant impact on the fairness with which algorithms are compared. It thus makes sense to be able to compare multi-method algorithms in a meaningful way, the same constituent algorithms and control parameters should be used for each investigated multi-method framework. Unfortunately, this is not common practice in literature making it exceedingly difficult to gauge the success of different multi-method algorithms from different research areas. The value that can be obtained from a comparison of different multi-method algorithms under strictly similar conditions, should thus not be underestimated.

Based on the available literature, four "unchallenged" algorithms were identified from the various multi-method research areas. These algorithms are PAP [122], the FAUC-Bandit method [54], EEA-SLPS [177], and AMALGAM-SO [172].

2.3 Chapter summary

This chapter acted as both an introduction to optimization and commonly used single-method optimization methods, as well as providing a literature review of multi-method literature. The first part of the chapter described the basics of, as well as notable variations of, a number of single-method optimization algorithms used throughout this thesis, namely PSO, DE, GA, ES, and TS. The multi-method literature looked at all fields addressing the algorithm selection problem, namely ensemble algorithms, portfolio algorithms, hyper-heuristics and adaptive operator selection. A number of algorithms

differentiated themselves as state-of-the-art algorithms in each field, namely the FAUC-Bandit method, PAP, EEA-SLPS, and AMALGAM-SO.

Chapter 3

The Heterogeneous Meta-hyper-heuristic

The use of meta-heuristics as low level heuristics is a novel concept in hyper-heuristic research. A framework thus needs to be defined which should firstly enable the effective simultaneous use of meta-heuristics in a hyper-heuristic context. Secondly, it should be of use throughout the rest of the thesis as basis for the investigation of various algorithmic aspects in a meta-hyper-heuristic context. The framework is described in detail throughout the rest of this chapter.

3.1 Algorithm description

The proposed framework consist of a number of algorithmic elements. As indicated in Figure 3.1, these elements consists of a common population of entities, where each entity represents a candidate solution which is adapted over time, a set of low level meta-heuristic (LLM) algorithms, and an acceptance strategy.

At the start of the optimization run, each entity is randomly allocated to a LLM from the set of available LLMs. The allocated LLM is then used to adapt the entity for the following k iterations. This entity-to-LLM allocation is then updated on a dynamic basis throughout the optimization run at each subsequent k iterations while the

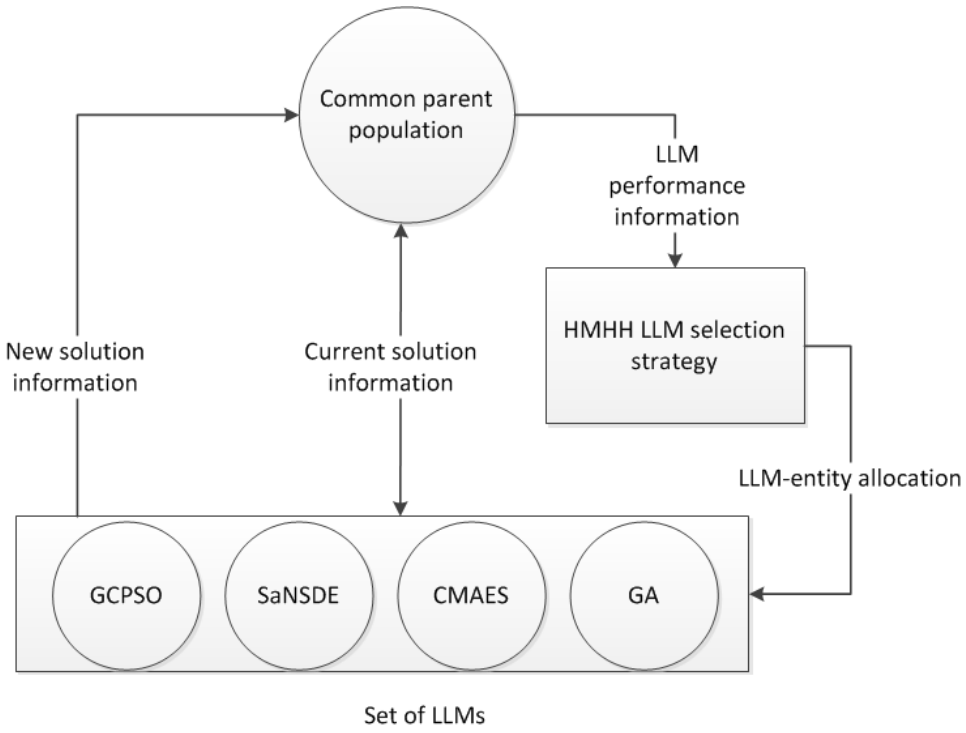

Figure 3.1: The heterogeneous meta-hyper-heuristic.

common parent population is continuously updated with better solutions. Throughout this process, the performance of the various LLMs is stored as defined by $Q_{\delta m}(t)$, the total improvement in fitness function value of all entities assigned to the m^{th} LLM from iteration $t-k$ to iteration t. More specifically,

$$Q_{\delta m}(t) = \sum_{i=1}^{|\boldsymbol{I}_m(t)|} (f(\boldsymbol{x}_i(t-k)) - f(\boldsymbol{x}_i(t))) \qquad \forall i \in \boldsymbol{I}_m(t) \tag{3.1}$$

where $f(\boldsymbol{x}_i(t))$ denotes the objective function value of entity i at time t and $\boldsymbol{I}_m(t)$ is the set of entities allocated to the m^{th} LLM at time t. $Q_{\delta m}(t)$ is used throughout the optimization process as input to the HMHH selection process responsible for allocating entities to LLMs. Various entity-to-LLM allocation strategies can be used in the context of the HMHH algorithm. A number of these strategies are investigated in the next chapter.

The main idea of the HMHH algorithm is that an intelligent algorithm can be evolved which selects the appropriate LLM at each k^{th} iteration to be applied to each entity within

the context of the common parent population, to ensure that the population of entities converge to a high quality solution. Each entity can then use unique meta-heuristics that are helpful for dealing with the specific search space characteristics it is encountering at that specific stage of the optimization process. The HMHH algorithm thus functions as a single method with different search operators. As an example, if entity i at time t is allocated to a GA, entity i "sees" all other entities as individuals in a GA population and utilizes these other "individuals" in the crossover and mutation operators used to generate entity i at time $t + 1$. The fact that different operators can be applied to each entity also implies that addition information must be stored per entity throughout the optimization process. An examples of this type of information is the best previous position found by the entity since the start of the optimization process. Various entity-specific algorithm control parameters such as for example DE scaling factors are stored while an entity is allocated to the LLM in question (DE) and reset as soon as the entity is allocated to another LLM.

Furthermore, unlike with for example PAP, all entities have access to the genetic material of all other entities, as if part of a common population of entities. Finally, with reference to Burke *et al.*'s classification [18], the heterogeneous meta-hyper-heuristic algorithm can be considered a heuristic selection methodology and an online learning perturbation hyper-heuristic. The HMHH pseudocode is provided in Algorithm 3.1.

The initial implementation of the meta-hyper-heuristic algorithm made use of seven LLMs, selected for their reputation as successful single-method optimization algorithms:

- A GA with a floating-point representation, tournament selection, blend crossover [51, 116], and self-adaptive Gaussian mutation [65]
- The guaranteed convergence PSO (GCPSO) [166]
- The barebones PSO algorithm (BBPSO) [84]
- A DE algorithm variation (*DE/best/bin*) [153]
- A second DE algorithm variation (*DE/rand/bin*) [153]
- A third DE algorithm variation (*DE/rand-to-best/bin*) [153]

Initialize the parent population $\boldsymbol{X}$
$A_i(t)$ denotes the algorithm applied to entity i at iteration t
k denotes the number of iterations between entity-to-algorithm allocation
$t = 0$
for *All entities* $i \in \boldsymbol{X}(0)$ **do**
    Randomly select an initial algorithm $A_i(0)$ from the set of LLMs to apply to entity i
end
while *no stopping condition is satisfied* **do**
    for *All entities* $i \in \boldsymbol{X}(t)$ **do**
        Apply constituent algorithm $A_i(t)$ to entity i for k iterations
    end
    $t = t + k$
    Calculate $Q_{\delta m}(t)$, the total improvement in fitness function value of all entities assigned to algorithm m from iteration $t - k$ to iteration t using Equation (3.1).
    for *All entities* $i \in \boldsymbol{X}(t)$ **do**
        Use $Q_{\delta m}(t)$ as input to select LLM $A_i(t)$ according to the specified selection mechanism. (The alternative selection mechanisms are considered in Section 4.1).
    end
end

Algorithm 3.1: The heterogeneous meta-hyper-heuristic.

- The covariance matrix adapting ES algorithm (CMAES) [3]

A slight modification had to be made to the CMAES algorithm to ensure that it functions effectively in the HMHH framework: When the population has reached certain conditions CMAES terminates even though the other LLMs are still available for selection. The complete list of CMAES stopping conditions is described in Section 6.1.3. One example of a stopping condition is when the condition number of the covariance matrix exceeds 10^{14}. In such a case, the entities previously assigned to CMAES are redistributed among the other LLMs as defined by the selection strategy of the algorithm.

3.2 Summary

The purpose of this chapter was to present the HMHH algorithm framework which is used throughout the rest of this thesis to investigate various algorithmic aspects critical to meta-hyper-heuristic performance. The next chapter elaborates on the initial investigations conducted into alternative HMHH design configurations.

Chapter 4

Initial Analysis of the Meta-hyper-heuristic Framework

Various design alternatives exist with regard to the implementation of the HMHH framework. The aim of this chapter is to investigate the various design alternatives such as the determination of the selection strategy and the selection of LLMs. Section 4.1 describes the investigation conducted into the use of alternative evolutionary selection operators as entity-to-algorithm selection strategies in the HMHH framework. The impact of the set of LLMs is investigated in section 4.2. Section 4.3 considers the use of local search in the HMHH framework. Various local search design issues, such as the selection of the entity to which the local search operators is to be applied, and the application of the local search to heuristic versus solution space, is considered. Finally, the chapter is concluded in Section 4.4.

4.1 Entity-to-algorithm selection strategies

When simultaneously using more than one algorithm to solve optimization problems, two questions arise:"How should the entities be allocated to LLMs for optimization?" and "When should the entity-to-LLM allocation be updated?" The answer to the first question may lie in the traditional EA literature. Selection, one of the main EA operators, determines how entities should be allocated to LLMs to allow good solutions

to propagate to subsequent generations. Attempting to utilize the traditional selection operators used in EAs to allocate entities to LLMs in multi-method optimization strategies, is a promising research area which is explored throughout the rest of this section. Section 4.1.1 provides a brief overview of a number of evolutionary selection strategies before Section 4.1.2 describes the empirical comparison between the selection strategies.

4.1.1 Evolutionary selection strategies

A number of traditional evolutionary selection strategies have been selected for investigation in this section based on their successful application in single-method EAs. These strategies are described below:

- **Random selection**

  The random selection strategy (RAND) randomly assigns entities to LLMs every k^{th} iteration. The probability of selecting an LLM to be applied to an entity is thus equal to $\frac{1}{n_a}$, where n_a is the number of LLMs available for selection. No memory of previous good performance is retained and no learning is attempted. This strategy has a very low selective pressure.

- **Roulette-wheel selection**

  Roulette-wheel selection (ROUL) [74] is a proportional selection strategy which biases the selection towards LLMs which performed well during the previous k iterations. This selection strategy has the highest selection pressure of all the investigated strategies, which could lead to the hyper-heuristic getting stuck in a local optimum. The probability of selecting the m^{th} LLM, p_m, is given as

  $$p_m = \frac{Q_{\delta m}(t)}{\sum_{i=1}^{n_a} Q_{\delta i}(t)}, \tag{4.1}$$

  where $Q_{\delta m}(t)$ is defined as the total improvement in fitness function value of all entities assigned to the m^{th} LLM from iteration $t-k$ to iteration t. All algorithms in this thesis considers only the last k iterations, where k was tuned by means of F-race [10]. This decision was based on the findings of Bai *et al.* [5] that hyper-heuristics with short-term memory produce better results than both algorithms with no memory and infinite memory.

- **Tournament selection**

  Tournament selection (TOUR) [13], with a tournament size, n_t, of three and a population size, n_s of 100, has a slightly higher selective pressure than random selection. For each entity in the population, n_t LLMs are randomly selected from the set of available LLMs. These LLMs are then compared according to their performance over the past k iterations based on $Q_{\delta m}(t)$. The entity under consideration is then assigned to the LLM which showed the best improvement over the last k iterations.

- **Rank-based selection**

  Rank-based selection (RANK) [7] works on the basis that all LLMs are ranked according to their performance during the previous k iterations, where the best performing LLM has the lowest rank and the worst performing LLM has the largest rank. This information is then used to determine the probability of selection. If linear ranking is used, the probability of selecting the m^{th} LLM, p_m, is given as

  $$p_m = \frac{\hat{\lambda} + (r_m(t)/(n_a - 1))(\hat{\lambda} - \tilde{\lambda})}{n_a} \tag{4.2}$$

  where $r_m(t)$ is the rank of LLM m at iteration t, $1 \leq \hat{\lambda} \geq 2$, and $\tilde{\lambda} = 2 - \hat{\lambda}$. Rank-based selection is thought to have a lower selection pressure when compared to Roulette-wheel selection since the number of entities which can be allocated to a single LLM is limited, preventing fast convergence to an initial good performing LLM.

- **Boltzman selection**

  Boltzman selection (BOLT) [98] is directly derived from simulated annealing [85]. The selection probability of the m^{th} LLM is given as

  $$p_m = \frac{1}{1 + e^{-Q_{\delta m}(t)/T_{Bolt}(t)}} \tag{4.3}$$

  where $T_{Bolt}(t)$ is defined as the temperature parameter which decreases linearly over time. The idea is that all LLMs are provided an almost equal opportunity to be selected at the start of the optimization run. However, as the run progresses, more

emphasis is placed on the selection of better performing LLMs. This strategy also has a relatively low selection pressure similar to the tournament selection strategy.

- **TS-based selection**

  The TS-based heuristic selection strategy (TSHH) makes use of the heuristic selection mechanism developed by Burke *et al.* for timetabling problems [20]. Here LLMs compete with each other based on the principles of reinforcement learning and a tabu list is maintained for each entity to ensure that poor performing LLMs are not repeatedly applied to the same entity. The selection strategy ranks the LLMs based on their performance with regard to a specific entity. The rank of the m^{th} LLM with respect to entity i, r_{im}, is increased by one unit for every improving move and decreased by one unit for every non-improving or equal move. A high rank indicates good performance and a low rank indicates poor performance. The highest ranking non-tabu LLM is always selected for use and the LLM that has spent the longest time in the tabu list is made non-tabu as soon as the maximum size of the tabu list is exceeded. TSHH has a relatively high selection pressure, similar to the rank-based strategy.

4.1.2 Comparative analysis of alternative selection strategies

The various strategies were inserted in the framework described in Chapter 3 and were evaluated on the first 17 problems of the 2005 IEEE Congress of Evolutionary Computation benchmark problem set [155]. This benchmark problem set allows algorithm performance evaluation on both uni-modal and multi-modal functions and includes various expanded and hybridized problems, some with noisy fitness functions.

The algorithm control parameters values listed in Table 4.1 were found to work well for the algorithms under study in previous research by the author of this thesis [64]. The number of iterations between re-allocation, k, was, however, tuned by means of F-Race. F-race is a racing algorithm which makes use of a statistical approach to select the best control parameter configuration out of a set of candidate configurations under stochastic evaluations [10]. The notation $a \longrightarrow b$ is used to indicate that the associated parameter is decreased linearly from a to b over 95% of the maximum number of iterations, I_{max}.

Table 4.1: HMHH algorithm parameters.

Parameter	**Value used**
Number of entities in common population (n_s)	100
Number of iterations between re-allocation (k)	5
Maximum number of iterations (I_{max})	$n_x n_s$
PSO parameters	
Acceleration constant (c_1)	$2.0 \longrightarrow 0.7$
Acceleration constant (c_2)	$0.7 \longrightarrow 2.0$
Inertia weight (w)	$0.9 \longrightarrow 0.4$
DE parameters	
Probability of reproduction (p_r)	$0.75 \longrightarrow 0.25$
Scaling factor (F)	$0.75 \longrightarrow 0.125$
GA parameters	
Probability of crossover (p_c)	$0.6 \longrightarrow 0.4$
Probability of mutation (p_{mut})	0.1
Blend crossover parameter (α)	0.5
GA tournament size (N_t)	3
CMAES parameters	As specified in [3].
Tournament selection parameters	
Algorithm selection tournament size (n_t)	3
Rank-based selection parameters	
Number of offspring of best entity ($\hat{\lambda}$)	3
Boltzman selection parameters	
Temperature parameter (T)	$1 \times 10^5 \longrightarrow 0$
Rank-based Tabu selection parameters	
Size of tabu list	3

The results for each strategy on each of the 17 CEC 2005 problems in dimensions 10, 30, and 50, were recorded over 30 independent simulation runs. If the global optimum was reached within the specified accuracy (10^{-6} for problems 1 to 5 and 10^{-2} for the rest

of the problems), the run was stopped and the number of function evaluations required, $\#FEs$, to reach the global optimum, was recorded. Where the global optimum could not be found within the maximum number of iterations, I_{max}, the difference, FFV, between the final solution at I_{max} and the global optimum, was recorded. This same experimental setup is used throughout the rest of the thesis to ensure that algorithms are evaluated under the same conditions. The results of the selection strategy comparison is presented in Tables B.1 to B.3, where μ and σ denote the mean and standard deviation associated with the corresponding performance measure and $\#FEs$ denotes the number of function evaluations which were needed to reach the global optimum within a specified accuracy.

Mann-Whitney U tests were used to evaluate the various strategies according to the number of iterations required to obtain the final fitness function value, as well as the quality of the actual fitness function value. The results in Table 4.2 were obtained by comparing each dimension-problem combination of the strategy under evaluation, to all of the dimension-problem combinations of the other strategies. For every comparison, a Mann-Whitney U test at 95% significance was performed (using the two sets of 30 data samples of the two strategies under comparison) and if the first strategy statistically significantly outperformed the second strategy, a win was recorded. If no statistical difference could be observed a draw was recorded. If the second strategy outperformed the first strategy, a loss was recorded for the first strategy. The total number of wins, draws and losses were then recorded for all combinations of the strategy under evaluation. To illustrate, (2-35-14) in row 1 column 3 of Table 4.2, indicates that random selection outperformed boltzman selection twice over the benchmark problem set, while 35 draws and 14 losses were recorded.

To further analyze the exploration-exploitation behaviour of the various selection strategies, one of the benchmark problems was randomly selected and the population diversity of each of the six selection strategies was plotted over time in Figure 4.1. In this thesis, population diversity or solution space diversity (SSD) is defined as

$$SSD = \frac{1}{n_s} \sum_{i=1}^{n_s} \sqrt{\sum_{j=1}^{n_x} (x_{ij}(t) - \overline{x}_j(t))^2} \tag{4.4}$$

where n_s is the number of entities in the common parent population and n_x is the number of dimensions, $x_{ij}(t)$ is the position of the j^{th} dimension of the i^{th} entity at time t, and

Table 4.2: Hypotheses analysis of alternative selection strategies.

	RAND	**ROUL**	**BOLT**	**TOUR**	**RANK**	**TSHH**	**TOTAL**
RAND	NA	26-16-9	2-35-14	11-22-18	2-31-18	12-23-16	53-127-75
ROUL	9-16-26	NA	7-15-29	5-20-26	5 - 16-30	2 -22-27	28-89-138
BOLT	14-35-2	29-15-7	NA	14-20-17	3-45-3	15-21-15	75-136-44
TOUR	18-22-11	26-20-5	17-20-14	NA	16-23-12	11-27-13	88-112-55
RANK	18-31-2	30-16-5	3-45-3	12-23-16	NA	16-19-16	79-134-42
TSHH	16-23-12	27-22-2	15-21-15	13-27-11	16-19-16	NA	87-112-56

$\overline{x}_j(t)$ is the mean of the j^{th} dimension of all particles in the swarm at time t [171].

From the results it is clear that both the random selection and roulette-wheel selection strategies performed very poorly. This confirms our hypothesis that simply using a number of different algorithms interchangeably throughout the optimization run, is insufficient to obtain good multi-method algorithm performance and that a more intelligent selection strategy promotes better performance. The poor performance of the roulette-wheel selection strategy can be attributed to the high selective pressure and associated quick convergence of the fitness-based algorithms as can be seen in Figure 4.1.

The better performing strategies were those with less selective pressure which allowed the LLMs more time to explore and converge slowly to better performing strategies. Rank-based selection performed the best of all selected strategies over the entire problem set, with Boltzman selection second and tournament selection third. A closer inspection of the results in Tables B.1 to B.3 showed that the good performance of tournament selection could be traced to its performance on uni-modal problems, while the TS-based selection strategy was the best performing strategy when solving more multi-modal problems. It is suspected that the advantages of a multi-method algorithm becomes more evident as the complexity of the problem increases and since the TSHH algorithm showed the best performance on the more difficult multi-modal problems, it was selected for further investigation.

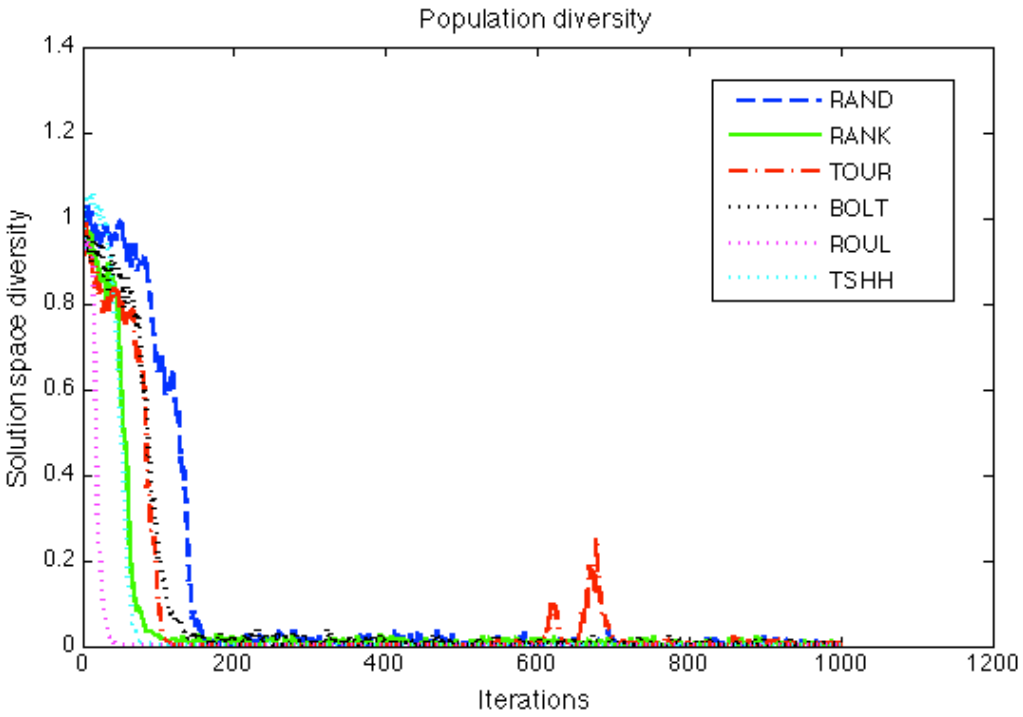

Figure 4.1: Solution space diversity of different selection strategies on the 11^{th} CEC 2005 problem in 10 dimensions.

4.2 Investigating the selection of low level meta-heuristics in the meta-hyper-heuristic framework

At this stage of the thesis, after the main structure of the proposed HMHH was defined with regard to the selection strategy, it became necessary to return to the selection of the set of available LLMs. It was suspected that the properties of the set of LLMs has a significant influence on algorithm performance.

One of the main ideas of using multiple algorithms in the same optimization run is that the algorithms should complement each other. An example of this complementarity would be when the set of LLMs compensates for the strength and weaknesses of each

individual LLM. The aim of this section is to redefine a set of complementary LLMs and compare this new set to the previously used set to identify any performance gains.

There are various methods to define the extent to which a set of algorithms complement each other. Hadka and Reed [67] based the selection of mutation strategies available to their algorithm on the distribution of offspring associated with various operators. Montazeri *et al.* [103] ensured that their set of low level heuristics contains both exploiter heuristics, designed for intensification, and explorer heuristics, aimed at diversification. Peng *et al.* [122] proposed a pairwise metric which can be used to determine the risk associated with an algorithm failing to solve the problem in question. Engelbrecht [49] selected complementary swarm behaviours in a heterogeneous PSO by analyzing the exploration-exploitation finger prints of the different PSO updates.

Two aspects were considered in the selection of the new proposed set of LLMs. Firstly, the single-method performance of a LLM on the benchmark set used in this thesis was considered. Secondly, the diversity profiles of the LLMs were considered and the selection was made to ensure a diverse set of LLMs based on their diversity profiles. The proposed improved set of LLMs consists of the following algorithms:

- A GA with a floating-point representation, tournament selection, blend crossover [51] [116], and self-adaptive Gaussian mutation [65]
- The guaranteed convergence PSO algorithm (GCPSO) [166]
- The self-adaptive DE algorithm with neighborhood search (SaNSDE) [179]
- The covariance matrix adapting ES algorithm (CMAES) [3]

As illustrated in Figure 4.2 for the 11^{th} 2005 IEEE Congress of Evolutionary Computation benchmark problem set in 50 dimensions [155], the population diversity of the GCPSO algorithm decreases at a much slower rate than the population diversity of, for example, the CMAES and GA algorithms. The idea is that, during different stages of the optimization process, different exploration and exploitation rates are required. By making sure that algorithms that address the exploration-exploitation trade-off differently are available, the chances of countering an improper population diversity management strategy by one algorithm from the set, is greater. For example, if three out of the four

LLMs are in exploration mode at the start of the optimization process, only a percentage of the function evaluations are wasted on the single algorithm which is busy exploiting a local minimum.

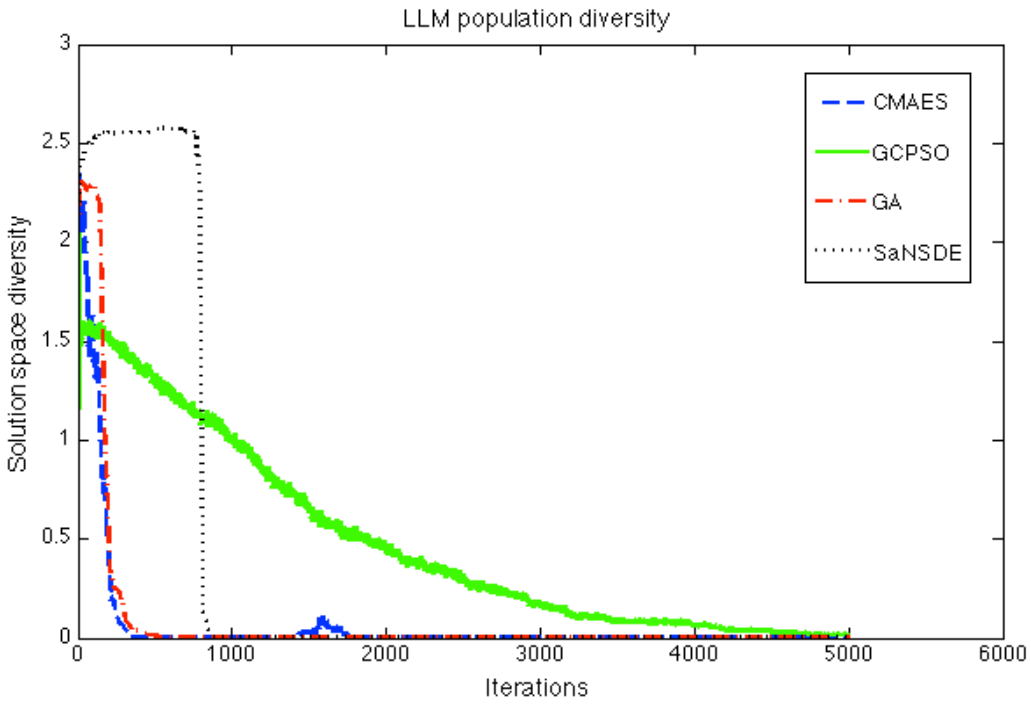

Figure 4.2: Diversity profiles of different LLMs on the 11^{th} CEC 2005 problem in 50 dimensions.

The results used for the comparison between the two sets of LLMs can be found in Tables B.3 and B.6. The results in Table B.3 were obtained by a TSHH algorithm utilizing a set of LLMs consisting of the GA, GCPSO, BBPSO, *DE/rand/bin*, *DE/best/bin*, *DE/rand-to-best/bin*, and CMAES. The results in Table B.6 were obtained by a TSHH algorithm utilizing the diverse set of LLMs described above. The frequency of use of each of the LLMs in the old and new set is indicated in Figures 4.3 and 4.4. It is encouraging to note that in both figures the worse performing algorithms, such as the BBPSO algorithm in Figure 4.3 and the GA and GCPSO algorithm in Figure 4.4, has a lower frequency of selection than the better performing algorithms such as CMAES.

The number of Mann-Whitney wins, draws and losses were calculated as 19 wins, 20

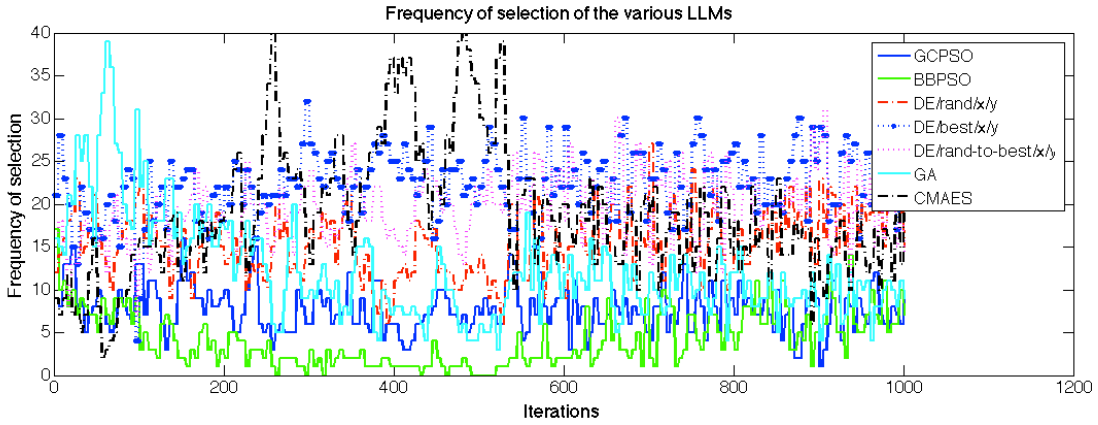

Figure 4.3: Frequency of use of each of the LLMs in the TSHH algorithm of Section 4.1 on the 11^{th} CEC 2005 problem in 50 dimensions. Frequency of use is determined by the number of entities allocated to the LLM under consideration per iteration.

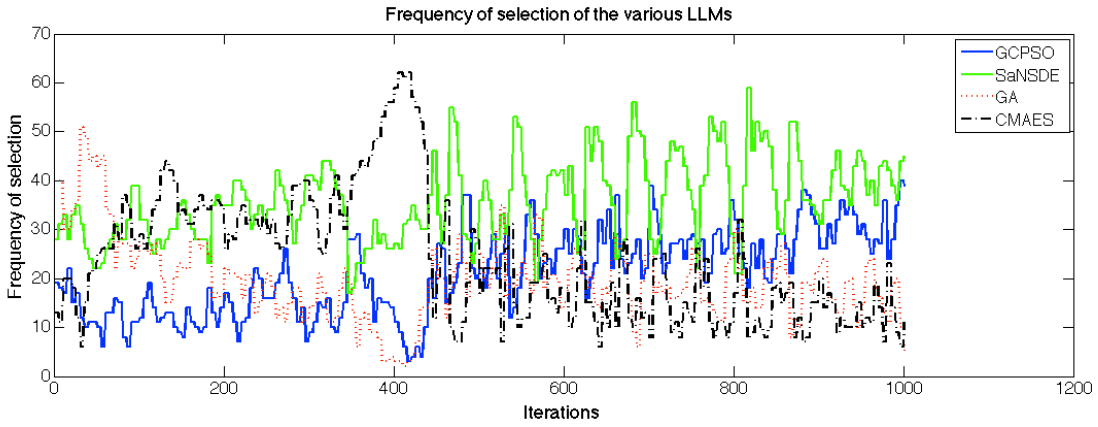

Figure 4.4: Frequency of use of each of the LLMs in the TSHH algorithm with a new set of diverse LLMs on the 11^{th} CEC 2005 problem in 50 dimensions. Frequency of use is determined by the number of entities allocated to the LLM under consideration per iteration.

draws, and 12 losses obtained by the new diverse set of LLMs when compared to the previous set of LLMs. These results indicate that a statistically significant performance benefit can be obtained by selecting LLMs that complement each other with regard to their diversity profiles. This improved set of LLMs are therefore used throughout the rest of the thesis when investigating additional algorithmic aspects.

4.3 Investigating the use of local search in the meta-hyper-heuristic framework

As mentioned in Chapter 2, one of the earliest examples of the hybridization of optimization algorithms can be found in the field of memetic computation. Similar to the use of local search (LS) strategies to improve meta-heuristic performance [104], this section investigates the use of local search strategies for improving hyper-heuristic performance. An overview of relevant existing work is provided in Section 4.3.1. Then the TSHH algorithm from Section 4.1 is used as a basis for investigating the selection of entities for refinement by the local search algorithm in Section 4.3.2.

4.3.1 An overview of local search and hyper-heuristics

A number of different strategies have been used to exploit the benefits of local search algorithms to improve hyper-heuristic algorithm performance. Firstly, local search strategies have been used as high-level hyper-heuristic strategies [117, 120]. These hyper-heuristics consist of a local search algorithm which manipulates a number of low level algorithms. A perturbative hyper-heuristic using the "Only improvements" acceptance strategy [35] is an example of this type of local search application. The aim of perturbative hyper-heuristics is to improve a candidate solution through a process of automatically selecting and applying one of a set of available heuristics to an existing candidate solution. A detailed review of a large number of perturbative hyper-heuristics is provided in [17].

Secondly, local search algorithms can also be incorporated into the set of available low level heuristics [121]. This option can have a significant effect on the diversity of the set of LLMs available for selection, especially where meta-heuristics and local search algorithms are utilized in combination as low level heuristics.

Finally, local search can be applied directly to the solution space. A good example of this is Qu and Burke's graph-based hyper-heuristic framework [125] where a local search algorithm is applied directly to the solution space in conjunction with a hyper-heuristic strategy which is applied to the heuristic space.

4.3.2 Investigating entity selection in a local search-based hyper-heuristic

Four selection mechanisms for selecting entities to which a local search algorithm is to be applied are investigated in this section:

- LS1HH - local search is applied to only the best entity of each iteration at each iteration.
- LS2HH - local search is applied to a single randomly selected entity at each iteration.
- LS3HH - roulette-wheel selection is applied to the entire population to select an entity at each iteration.
- LS4HH - the TS-based selection mechanism is used to select one or more entities at each iteration for further exploitation.

Application of the local search algorithm to the best performing entity is thought to be productive since the assumption is that this entity has a greater probability of being positioned in the basin of attraction of a global optimum. On the other hand, LS1HH also has increased risk of being stuck in a local minimum due to its high selective pressure. Because LS3HH is based on roulette-wheel selection it has a lower selective pressure than LS1HH, but a higher selective pressure than LS2HH.

It should be noted that the number of entities to which local search should be applied was defined *a priori* for the first three algorithms. At each iteration a selection mechanism independent of the hyper-heuristic is applied to the solution space to select the entities to be exploited. LS4HH makes the local search algorithm available for selection and application to the algorithm entities by defining the local search as one of the low level heuristics. The high level hyper-heuristic strategy is thus responsible for selecting the number of entities per iteration, as well as the specific entities of the population to which the local search algorithm should be applied. Schematic depictions of the algorithm structures is provided in Figures 4.5 and 4.6.

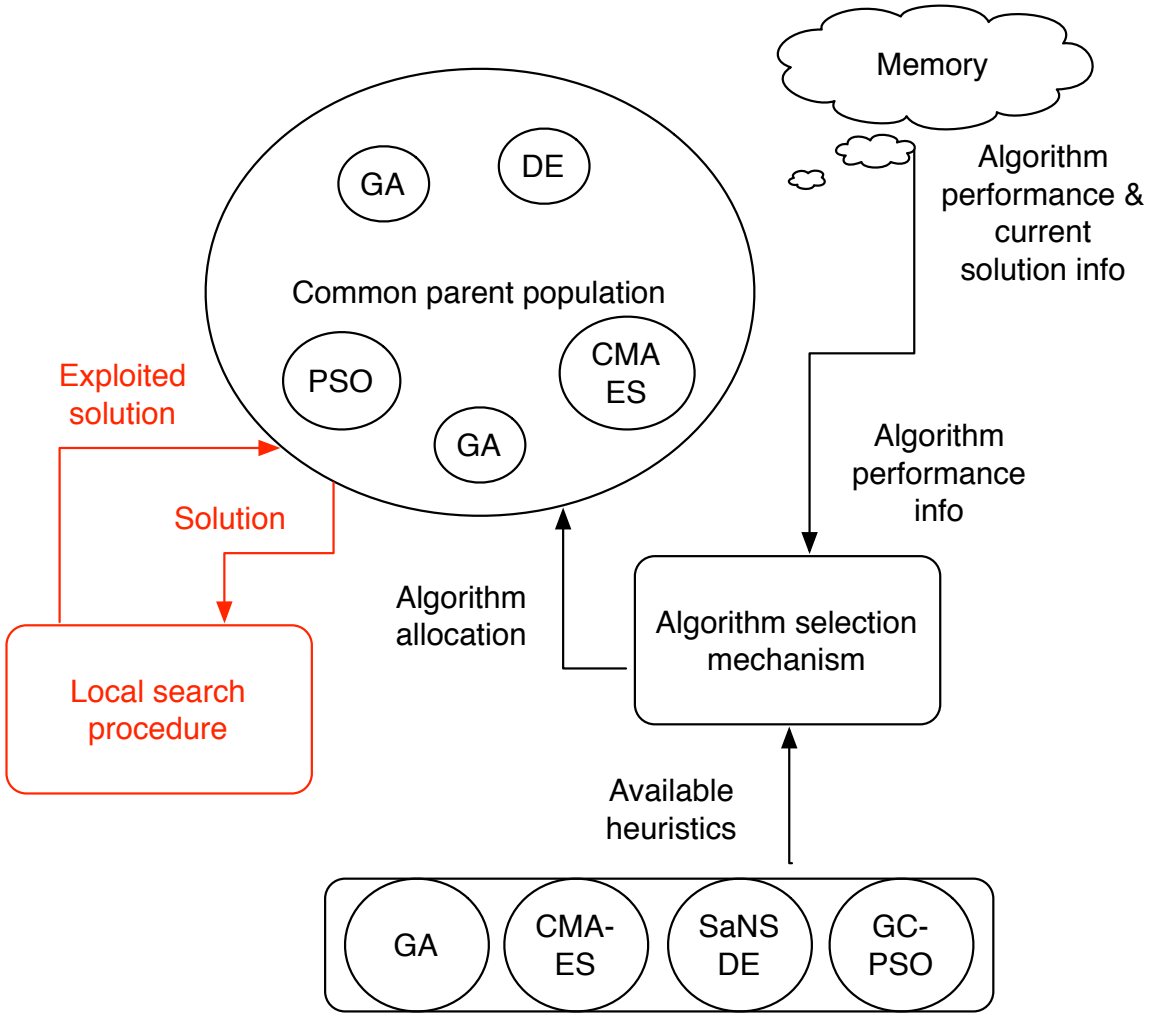

Figure 4.5: A schematic depiction of the structure of LS1HH, LS2HH, and LS3HH.

Throughout the rest of the thesis, a Broyden-Fletcher-Goldfarb-Shanno (BFGS) Quasi-Newton method with a cubic line search procedure, as implemented in Matlabs optimization toolbox, was used as local search algorithm. The BFGS algorithm was selected since it outperformed simpler local search algorithms. The same experimental setup of the previous section was used to evaluate the four algorithm variations and the results are recorded in Tables B.4 and B.5. The Mann-Whitney U test wins-draws-losses are provided in Table 4.3.

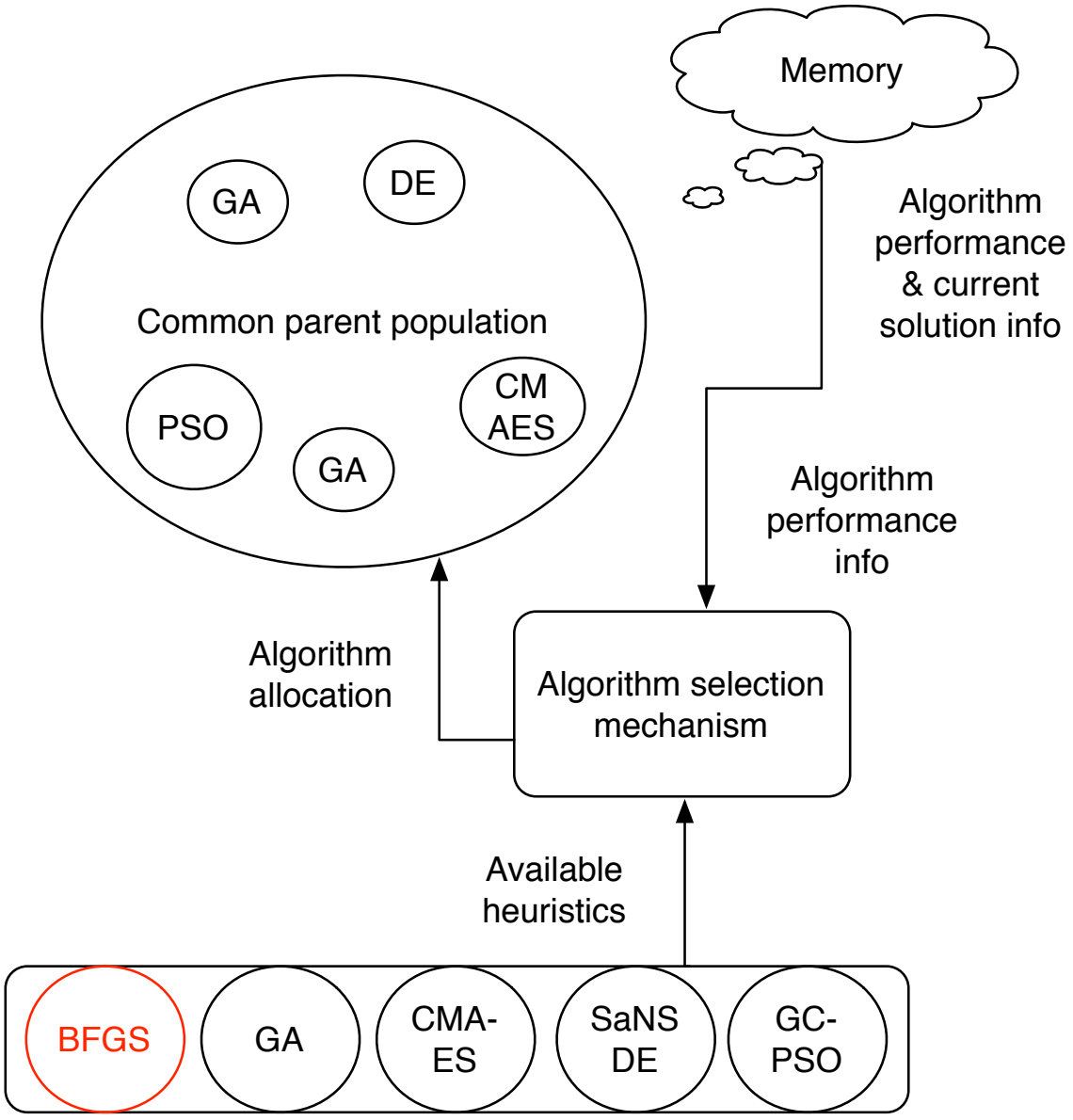

Figure 4.6: A schematic depiction of LS4HH.

It is clear from the results that application of the local search algorithm directly to the search space independently of the hyper-heuristic is a better strategy than defining the algorithm as a low level heuristic. Due to the line search which uses 20 function evaluations per application and is included in a single iteration of the local search al-

Table 4.3: Hypotheses analysis of alternative local search selection strategies.

	LS1HH	**LS2HH**	**LS3HH**	**LS4HH**	**TOTAL**
LS1HH	NA	1-49-1	2-48-1	33-18-0	36-115-2
LS2HH	1-49-1	NA	2-48-1	33-18-0	36-115-2
LS3HH	1-48-2	1-48-2	NA	34-17-0	36-113-4
LS4HH	0-18-33	0-18-33	0-17-34	NA	0-53-100

gorithm, the algorithm is computationally very expensive and consumes a large number of function evaluations per iteration. Whereas LS1HH to LS3HH limits the number of entities which can be exploited by means of local search to one, no such restrictions are placed on LS4HH. It is thus suspected that a larger computational budget is used earlier during the optimization run in the LS4HH algorithm when compared to the other three algorithms. LS1HH, LSHH2, and LSHH3 performed equally well with almost no statistically significant difference between their performance.

4.4 Summary

This chapter described the initial investigations into the meta-hyper-heuristic framework. Firstly, various traditional selection methods borrowed from EAs were investigated, of which the best performance was obtained by the TS-based strategy. Secondly, the impact of alternative LLMs was investigated and finally, the use of local search strategies to improve the performance of a meta-hyper-heuristic algorithm was considered. Various issues such as the application of local search directly to the search space versus the heuristic space and the mechanisms used to select entities for further exploitation were investigated. Experimental results indicated that application of the local search directly to the solution space to a single randomly selected individual per iteration is the best local search strategy.

Chapter 5

Diversity Management in the Meta-hyper-heuristic Framework

Although diversity management is not a new concept and is actually relatively common in single-method literature, its use in the multi-method algorithm world is relatively limited. The aim of this chapter is to investigate whether actively controlling diversity influences algorithm performance in a hyper-heuristic framework. Section 5.1 provides a brief overview of existing diversity management research. Hyper-heuristics lend themselves to two types of diversity management, namely solution space diversity (SSD) management, which is addressed in Section 5.2, and heuristic space diversity (HSD) management, which is addressed in Sections 5.3 and 5.4. Finally, the main findings of this chapter are summarized in Section 5.5.

5.1 An overview of existing diversity management strategies

Diversity management is an important concept that has received increasing attention recently. Traditionally, the ability of an optimization algorithm to balance exploration and exploitation has been shown to have a significant impact on its performance [39]. If the algorithm converges too quickly, it is more likely to become stuck in a local optimum. If the algorithm focuses too much on exploring new areas of the search space near the

end of the optimization run, time is wasted on exploring the search space which could have been used to further refine promising solutions.

Various examples of algorithms which attempt to either further exploit the solution space around good performing solutions [125], or improve the overall exploration ability of the hyper-heuristic by applying diversity management mechanisms directly to the solution space can be found in the hyper-heuristic literature. The AMALGAM-SO algorithm [172] makes use of a species selection mechanism to maintain solution space diversity. Sabar *et al.* [138] and Veerapen *et al.* [169] took both solution space diversity and solution quality into account when allocating entities to algorithms. Segredo *et al.* [143] converted a single objective problem into a multi-objective optimization problem through the addition of a second diversity-based objective. A hyper-mutation operator is triggered in the evolutionary-based hyper-heuristic of Salcedo-Sanz *et al.* [141] when the solution space diversity drops to zero.

Ren *et al.* [132] identified the issue of low level heuristic parameters that could influence performance. They addressed this issue through the development of a hyper-heuristic with low level parameter optimization consisting of a low level heuristic management module and a low level parameter management model. The additional parameter optimization variables did, however, have a significant influence on the size of the search space. This issue was addressed by means of a heuristic space reduction mechanism. The low level heuristics were subdivided into explorers and exploiters and the algorithm continually alternated between the two types of heuristics in an attempt to manage the exploration-exploitation trade-off.

Solution space diversity management is also an important consideration in a field closely related to hyper-heuristics, namely memetic computing [30]. A detailed review of diversity management in memetic computing and other fields is provided in [39]. Notable examples of using solution space diversity to control the exploration-exploitation trade-off of MAs are the fitness-diversity adaptive local search algorithms of Caponio *et al.* [24]. Fitness diversity-adaptive algorithms are based on the idea of using population diversity to guide the exploration versus exploitation balance of the algorithm. Multiple refinement methods are usually involved, each with a different impact on solution space diversity. A fitness-diversity measure is calculated at each iteration and a self-adaptive

criterion determines which refinement method is applied. Caponio *et al's* algorithm made use of a Hooke-Jeeves [75] and Nelder-Mead simplex algorithm [106], but a large number of other fitness-diversity-based algorithms have also been proposed utilizing different types of diversity measures and different algorithms to increase or decrease population diversity [25, 87, 159].

There are, however, a number of issues that have been identified with regard to the use of fitness-diversity adaptive algorithms. A comparative study of different algorithms [109] has shown that algorithm performance is significantly impacted by the choice of diversity measure used and that the best diversity measure is dependent on both the problem features and characteristics of the algorithm framework. Furthermore, a fitness-based diversity measure is normally used instead of calculating the diversity of the actual solutions in order to reduce the computational complexity of the algorithms. Depending on the nature of the fitness landscape this could lead to an incorrect indication of population diversity. For example, consider the case where there are two similar sized local minimums a significant distance apart in the fitness landscape. A fitness-based diversity measure would indicate that all solutions congregating near both these minimums are close to each other in the search space even though two solutions from different minimums are clearly far apart.

Based on the importance of effective management of solution space diversity in traditional optimization algorithms, it is not farfetched to think that the diversity of the heuristic space and how it is managed throughout the optimization run, could have an important impact on hyper-heuristic performance.

Recently, researchers have started to dynamically update the set of low level heuristics during the optimization run. Sim *et al.* [146] made use of a self-organizing network of low level heuristics to ensure that different heuristics were available to cover different areas of the search space. Random heuristics were added at fixed time intervals and an affinity measure related to the difference in performance between the different low level heuristics was used to determine when an under-performing heuristic should be removed.

The evolutionary selection hyper-heuristic of Mısır *et al.* [102] makes use of an adaptive dynamic heuristic set strategy, a move acceptance strategy, and a re-initialisation mechanism to manage the exploration-exploitation trade-off. A number of decision mech-

anisms for activating or de-activating these sub-mechanisms were also employed. The algorithm won the first international domain heuristic search challenge where problems from six different domains were considered [16].

Caraffini *et al.* [26] investigated the advantages of employing diverse local search components during the development of the Parallel Memetic Structure (PMS). When compared to its individual components, the difference in performance was not that obvious for low dimensional problems (30 dimensions), but PMS outperformed its component algorithms for problems of 1000 dimensions. PMS has also been successfully used in a highly successful computational prototype for automatic design of optimization algorithms: the Separability Prototype for Automatic Memes [27].

From the above discussion it is clear that a number of researchers have already considered techniques to improve the exploration-exploitation trade-off in a hyper-heuristics context. The selection of low level heuristics with regards to diversity management and the effective management of the set of low level heuristics over time have also been studied. However, to the best of the author's knowledge, this chapter documents the first attempt to define and measure the concept of heuristic space diversity and to manage the diversity of entity-to-LLM allocation to improve hyper-heuristic performance.

5.2 Investigating alternative solution space diversity management strategies

The first part of this chapter focuses on the effect of solution space diversity on meta-hyper-heuristic performance. Here, the effective balance of exploration and exploitation by applying different refinement methods depending on the population diversity, is explored. The strategies in this section is inspired by a simple concept. At the start of an optimization run a higher population diversity allows for greater exploration of the search space. Towards the end of the optimization run a lower population diversity encourages the algorithm to further exploit good solutions. This behaviour can be encouraged through the definition of a linearly decreasing upper bound, $UB_{div}(t)$, and a lower bound, $LB_{div}(t)$, which can be used to guide the solution space diversity to decrease

over time as shown in Figure 5.1. The bounds can be defined as follows:

$$UB_{div}(1) = SSD(1) + \gamma SSD(1) \tag{5.1}$$

$$UB_{div}(I_{max}) = \gamma SSD(1) \tag{5.2}$$

$$LB_{div}(1) = SSD(1) - \gamma SSD(1) \tag{5.3}$$

$$LB_{div}(I_{max}) = 0 \tag{5.4}$$

where γ is a positive constant between 0 and 1.

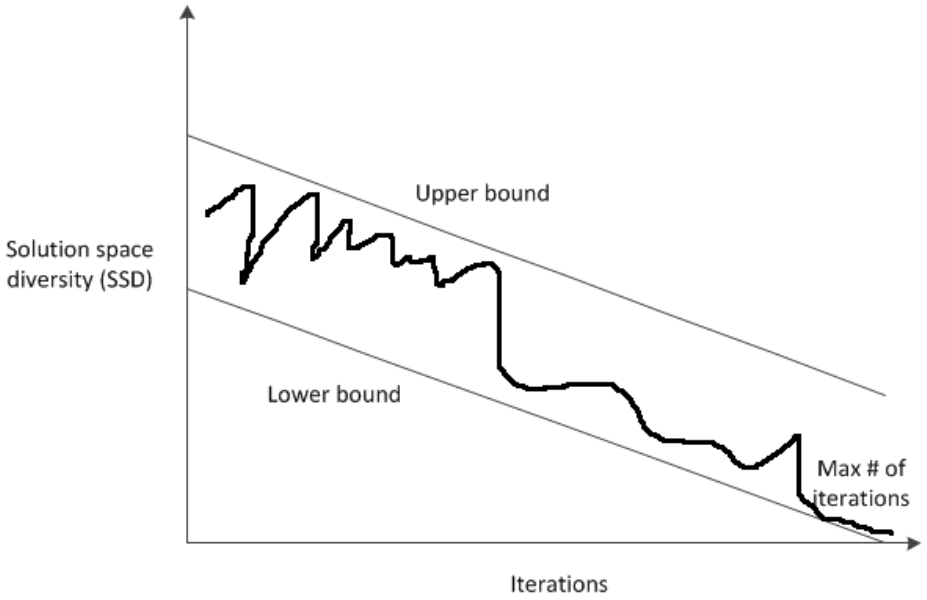

Figure 5.1: Upper and lower SSD bounds.

Throughout the rest of this section, different variations on the idea of managing the solution space diversity to fall within linearly decreasing bounds throughout the optimization run, is investigated. The first investigation focuses on exploitative SSD control strategies, in other words, strategies that aim to reduce the solution space diversity. Three algorithm variations were investigated:

- **TSHH** - This algorithm is the standard HMHH algorithm implemented as described in Section 4.2. No effort is made to manipulate the SSD in this algorithm, thus it will act as a baseline for comparing the other two SSD control strategies.

Table 5.1: Hypotheses analysis of alternative solution space diversity reduction control mechanisms.

	TSHH	**LSHH**	**ALSHH**	**TOTAL**
TSHH	NA	$32-17-2$	$4-46-1$	$36-63-3$
LSHH	$2-17-32$	NA	$0-19-32$	$2-36-64$
ALSHH	$1-46-4$	$32-19-0$	NA	$33-65-4$

- **LSHH** - This algorithm, LS2HH from Section 4.3.2, makes use of a local search algorithm applied consistently throughout the optimization run. The local search is applied to a single randomly selected entity from the population at each iteration.
- **ALSHH** - This algorithm can be considered an adaptive local search algorithm. Every time $SSD(t)$ exceeds the upper diversity bound, $UB_{div}(t)$, the local search algorithm is applied to a single randomly selected entity from the set of candidate solutions.

The results of the comparison between the first three SSD control strategies are presented in Tables B.6 and B.7. The Mann-Whitney U wins, draws and losses are provided in Table 5.1.

The results show that the LSHH algorithm does not perform well with the new set of LLMs selected in Section 4.2. Furthermore, the difference in performance between TSHH and ALSHH is relatively insignificant. TSHH is the preferred algorithm between the two due to the reduced computational complexity in comparison to ALSHH which incorporates a local search operator.

The second part of the investigation into alternative SSD control strategies focused on the impact that a diversity enhancing mechanism aimed at increasing $SSD(t)$ can have on hyper-heuristic performance. For this investigation Vrugt *et al.'s* [172] species selection mechanism was used to increase the solution space diversity of the population.

Assuming a minimization problem, the species selection mechanism takes as input the union of the candidate offspring population $\boldsymbol{c}(t+1)$ and the previous population $\boldsymbol{X}(t)$ sorted in order of decreasing fitness. This combined population is denoted by $\boldsymbol{R}(t)$. As shown in Algorithm 5.1, the best solution of $\boldsymbol{R}(t)$ is initially copied into $\boldsymbol{X}(t+1)$.

Then, in an iterative procedure, the next individual r in $\boldsymbol{R}(t)$ is compared to the species currently present in $\boldsymbol{X}(t+1)$. If the Euclidean distance of this individual to all points in $\boldsymbol{X}(t+1)$ is larger than a user-defined distance, ϵ, then r is added to $\boldsymbol{X}(t+1)$. This process is repeated n_s times until the resulting population is of size n_s. The distance, ϵ, is increased from $10^{-10}\delta$ to $10^{-1}\delta$, where $\delta = x_{max} - x_{min}$, as described in [172].

Let $\boldsymbol{X}(t+1)$ be the population of entities at time $t+1$
Let $\boldsymbol{R}(t)$ be the union of population $\boldsymbol{c}(t+1)$ and $\boldsymbol{X}(t)$ sorted in order of decreasing fitness
Let $D(p,r)$ be the Euclidian distance between entity p and r
Update the distance ϵ
Set $\boldsymbol{X}(t+1) = \emptyset$
while $|\boldsymbol{X}(t+1)| < n_s$ **do**
  Get best unprocessed member $\boldsymbol{r}$ of $\boldsymbol{R}(t)$
  for $\boldsymbol{p} \in \boldsymbol{X}(t+1)$ **do**
    $D(\boldsymbol{p},\boldsymbol{r}) = \sqrt{\sum_{j=1}^{n_x}(x_{pj} - x_{rj})^2}$
  end
  if *no* $\boldsymbol{p}$ *exists for which* $D(\boldsymbol{p},\boldsymbol{r}) \leq \epsilon$ **then**
    $\boldsymbol{X}(t+1) = \boldsymbol{X}(t+1) \cup \boldsymbol{r}$
  end
end

Algorithm 5.1: The species selection mechanism of [172].

Three algorithm variations making use of the species selection mechanism were investigated:

- **TSHH** - This algorithm is again the standard HMHH algorithm implemented as baseline algorithm.
- **DIVHH** - This algorithm employs the species selection approach at each n iterations to increase solution space diversity.
- **ADIVHH** - This algorithm employs the species selection approach adaptively whenever the lower diversity bound is breached.

Table 5.2: Hypotheses analysis of alternative solution space diversity increasing mechanisms.

	TSHH	**DIVHH**	**ADIVHH**	**TOTAL**
TSHH	NA	$32 - 17 - 2$	$4 - 46 - 1$	$36 - 63 - 3$
DIVHH	$2 - 17 - 32$	NA	$0 - 19 - 32$	$2 - 36 - 64$
ADIVHH	$1 - 46 - 4$	$32 - 19 - 0$	NA	$33 - 65 - 4$

The results of the comparison between TSHH, DIVHH, and ADIVHH is presented in Tables B.6 and B.8. The Mann-Whitney U wins, draws and losses are provided in Table 5.2.

The results indicate that, similar to the previous investigation, no significant performance improvement was obtained by the addition of the speciation selection mechanism. It should be noted that Section 4.2 selected the set of LLMs to ensure different diversity profiles. The HMHH without any SSD control mechanisms is thus already successful in managing its exploration-exploitation profile through the appropriate selection of LLMs with the required diversity profiles at different stages throughout the optimization run. Attempting to influence SSD through additional mechanisms does not seem to be valuable in this case. In line with the movement towards higher abstraction, which hyper-heuristics promote, it would be interesting to see whether HSD has any influence on algorithm performance. If this is found to be the case, then manipulation of HSD could be a way of influencing multi-method algorithm performance, similar to how SSD management is a means of influencing single-method performance.

5.3 Heuristic space diversity defined

The concept of heuristic space diversity is best illustrated by means of an example. In Figure 5.2 the entities in the population to the left were divided relatively equally between all of the available LLMs during entity-to-algorithm allocation. This population can be described as having a high HSD. On the other hand, most of the entities in the population on the right were allocated to the GA with only one entity each allocated to PSO and ES. This population can be described as having a low HSD.

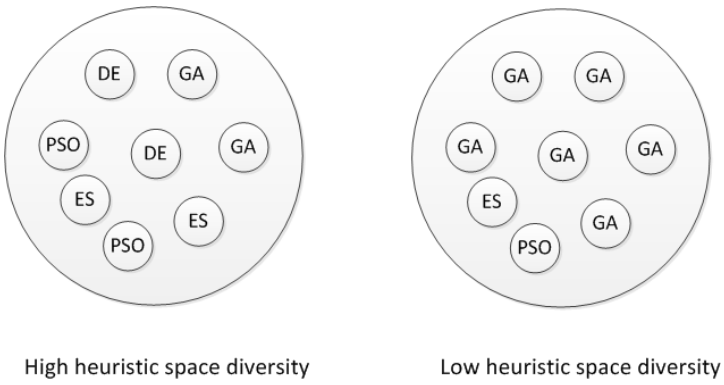

Figure 5.2: An example of a population with a high HSD and a population with a low HSD.

This thesis proposes that heuristic space diversity at time t, $D_h(t)$, be defined as follows:

$$D_h(t) = UB_{D_h(t)} \left(1 - \frac{\sum_{m=1}^{n_a} |\,\mathsf{T} - n_m(t)|}{1.5 n_s}\right) \tag{5.5}$$

with

$$\mathsf{T} = \frac{n_s}{n_a}, \tag{5.6}$$

where n_a is the number of algorithms available for selection by the hyper-heuristics, n_s is the number of entities in the population, $n_m(t)$ is the number of entities allocated to algorithm m at time t, and $UB_{D_h(t)}$ is the upper bound of the HSD measure. For the purposes of this thesis, $UB_{D_h(t)}$ was set to 100 so that $D_h(t) \in [0, 100]$.

The idea of the measure is to calculate a target number of entities, T, per algorithm. The absolute deviation between this target and the actual allocation of entities to algorithms is then used to determine the HSD associated with the entity-to-algorithm allocation. Maximum diversity would be achieved when all entities are assigned equally between algorithms and this translates into a $D_h(t)$ of 100.

5.4 Investigating alternative heuristic space diversity management strategies

Five strategies for controlling HSD throughout the optimization process, inspired by Ratnaweera *et al.'s* PSO parameter control strategies [127], are investigated in this chapter. The aim of the investigation is to evaluate the potential influence of the various strategies on HMHH performance. If it can be shown that effective HSD management can lead to a better performing algorithm, it opens the way for further research into HSD management which in turn will lead to greater performance gains. The rest of this section describes the investigated HSD control strategies in more detail.

- **The baseline HMHH algorithm** - This algorithm is the standard HMHH algorithm implemented as described in Section 4.2. No effort is made to manipulate the HSD in this algorithm.

- **Linearly decreasing HSD hyper-heuristic (LDHH)** - This algorithm is characterized by a linearly decreasing HSD. At the start of the optimization run all four LLMs are available for selection. During the optimization run, the worst performing LLM is removed from the set of available algorithms at predefined constant time intervals. Removing the worst performing algorithm allows more function evaluations for the better performing algorithms, leading to better algorithm performance. As an example, if the maximum allowable function evaluations are 100 000, the worst performing LLM at that time will be removed respectively at 25 000, 50 000 and 75 000 function evaluations. The idea is to force the hyper-heuristic to explore the heuristic space at the start of the optimization run and exploit the best performing LLM towards the end of the optimization run.

- **Exponentially decreasing HSD hyper-heuristic (EDHH)** - This algorithm is characterized by an exponentially decreasing HSD. All LLMs are available for allocation to entities at the start of the optimization run and LLMs are removed according to their performance at predetermined time intervals. This time, however, the LLMs are removed at exponentially increasing time intervals at 10 400,

23 800, 42 700 and 75 000 function evaluations. The result is a faster changeover from exploration to exploitation of the heuristic space.

- **Linearly increasing HSD hyper-heuristic (LIHH)** - This algorithm forces the hyper-heuristic to move from exploitation to exploration. At the start of the optimization run only one LLM is made available to the hyper-heuristic. As the optimization process progresses, additional LLMs are made available at predetermined linear constant time intervals. The idea is to obtain maximum performance gains from the first LLM. As the performance gains decrease, the rest of the LLMs become available to diversify the heuristic space and improve the overall algorithm performance. Two versions of the LIHH algorithm were investigated. LIHH1 assumes *a priori* knowledge of the LLM performance on the benchmark problem set being solved. The LLMs are ranked from best performing to worst performing. Only the best performing algorithm is made available to the hyper-heuristic at the start, with additional algorithms being made available according to their ranking. This has the effect of exploiting the best performing LLM at the start, before other algorithms are considered. LIHH2 requires no *a priori* knowledge. LLMs are sequenced randomly and made available to the hyper-heuristic one by one at linear time intervals without any consideration of previous algorithm performance.

- **Exponentially increasing HSD hyper-heuristic (EIHH)** - This algorithm is similar to the LIHH algorithm, the only difference being that exponential time intervals are used to add algorithms to the set of available LLMs. Exponential time intervals increase the rate of change of HSD, leading to a faster changeover from exploitation to exploration. Similar to the LIHH algorithm, two versions, namely EIHH1 (with *a priori* knowledge) and EIHH2 (without *a priori* knowledge), were investigated.

Since the number of algorithms, n_a, is no longer constant, the size of the tabu list had to be modified. When four algorithms were available for use by the hyper-heuristic the maximum tabu list size was set to two algorithms, to always allow at least two algorithms to be available for use. As soon as the number of available algorithms was reduced to three, the maximum tabu list size was set to one. When two or less algorithms were

in use, no tabu list was used. To ensure that a fair comparison was made between the different strategies, the baseline HMHH algorithm was re-implemented with a maximum tabu size of two. Apart from these changes in tabu list sizes, the same experimental conditions used throughout the thesis were again applied.

The results of the heuristic space diversity comparison are recorded in Tables B.9 to B.11. The statistical comparison between strategies is recorded in Table 5.3. Each strategy was compared to each other strategy and the same Mann-Whitney U wins - draws - losses format of the previous section was used. To illustrate, (5-31-15) in row 1 column 2, indicates that the HMHH strategy outperformed LDHH 5 times over the benchmark problem set. Thirty one draws and fifteen losses were recorded.

Table 5.3: Hypotheses analysis of alternative heuristic space diversity control mechanisms.

	HMHH	**LDHH**	**EDHH**	**LIHH2**	**EIHH2**	**TOTAL**
HMHH	NA	5-31-15	8-29-14	4-31-16	3-26-22	20-117-67
LDHH	15-31-5	NA	8-38-5	9-28-14	5-27-19	37-124-43
EDHH	14-29-8	5-38-8	NA	9-24-18	7-22-22	35-113-56
LIHH	16-31-4	14-28-9	18-24-9	NA	1-42-8	49-125-30
EIHH	22-26-3	19-27-5	2-22-7	8-42-1	NA	71-117-16

From the results it is clear that attempting to manage HSD does lead to statistically significantly different hyper-heuristic performance when compared to strategies where no HSD manipulation is used. Table 5.3 shows that the strategies where the HSD was controlled performed statistically similar or better than the baseline HMHH algorithm for 184 cases out of 204 tested. In contrast, only 20 cases of worse performance could be identified out of the 204 cases which were tested. LDHH was the best performing algorithm for the first five uni-modal problems. However, comparing the increasing HSD strategies (LIHH2 and EIHH2) to the decreasing HSD strategies (LSHH and EDHH) over the entire benchmark problem set resulted in 53 wins, 101 draws, and 30 losses. These results are mainly due to the good performance of the increasing strategies on the more complex multi-modal problems and indicate that the increasing HSD strategies performed better for the selected benchmark problem set. When the rate of change in

diversity is considered, the difference in performance is more subtle. For the decreasing strategies, the linearly decreasing HSD algorithm outperformed the exponentially decreasing algorithm. For the increasing strategies the best performing algorithm was the exponentially increasing EIHH2.

Better insight into these results can be obtained by studying HSD over the total number of iterations of the algorithm. Figure 5.3 plots the HSD of the median run of each of the HSD control strategies for problems 13 to 16 from the CEC 2005 problem set in 50 dimensions. Only the graphs of these four problems are provided here due to space constraints. The graphs of the other problems are very similar and are given in Appendix C.

As expected, the EDHH converged the quickest to a lower HSD where the allocation of entities-to-algorithms have stabilized. LDHH converged the second quickest, followed by HMHH. Similar to entities converging in a solution space, it is also evident that the slower converging LDHH was much more capable of adjusting and recovering towards a higher HSD when this adjustment was required. The alternative would be converging too quickly to a suboptimal entity-to-algorithm allocation which could adversely affect solution quality if insufficient diversity remained in the set of available constituent algorithms.

The success of the increasing HSDs can be largely attributed to the hyper-heuristic being able to exploit the performance benefits of a single algorithm before expanding the set of LLMs. The resulting increased HSD allowed the hyper-heuristic to incorporate new types of operators to continue the optimization process from where the first algorithm may have already stagnated. With regards to the rate of change of HSD, EIHH2 explored the heuristic space sooner than LIHH2 and maintained a higher HSD for longer when compared to LIHH2, leading to better exploration of the heuristic space and better overall solution quality.

The next experiment focused on investigating the impact of the availability of *a priori* knowledge. The two increasing strategies which make use of *a priori* knowledge, namely LIHH1 and EIHH1, were compared to the increasing strategies which do not make use of any *a priori* knowledge, namely LIHH2 and EIHH2. The results of this comparison are provided in Tables B.12 and 5.4. From the results it is clear that the availability of *a*

Table 5.4: Hypothesis analysis of the benefit of *a priori* information.

	LIHH2	**EIHH2**	**TOTAL**
LIHH1	$30 - 16 - 5$	$33 - 12 - 6$	$63 - 28 - 11$
EIHH1	$30 - 19 - 2$	$31 - 16 - 4$	$62 - 35 - 6$

priori knowledge of LLM performance on the benchmark problem set under consideration did have significant advantages. A dramatic performance improvement was obtained for most problems by LIHH1 and EIHH1 when compared to LIHH2 and EIHH2. The exception is problems nine and 12, where the no *a priori* strategies performed better. Unfortunately, this knowledge is not always readily available or can be time consuming to obtain.

5.5 Summary

This chapter investigated the performance benefits of diversity management in a hyper-heuristic framework. Constant and adaptive solution space diversity management strategies were proposed which utilized either a local search for reducing SSD or a species selection mechanism for increasing SSD. The results, however, indicated that influencing SSD had a relatively insignificant impact on hyper-heuristic performance.

The results from the investigation into HSD management were, however, significantly more promising. Six HSD control strategies were proposed and compared. The results indicated that a significant performance improvement can be obtained by controlling the HSD of the HMHH algorithm. The exponentially increasing HSD strategy was shown to outperform the decreasing, linearly increasing, and uncontrolled HSD strategies. An analysis into the performance benefits of *a priori* knowledge was also performed.

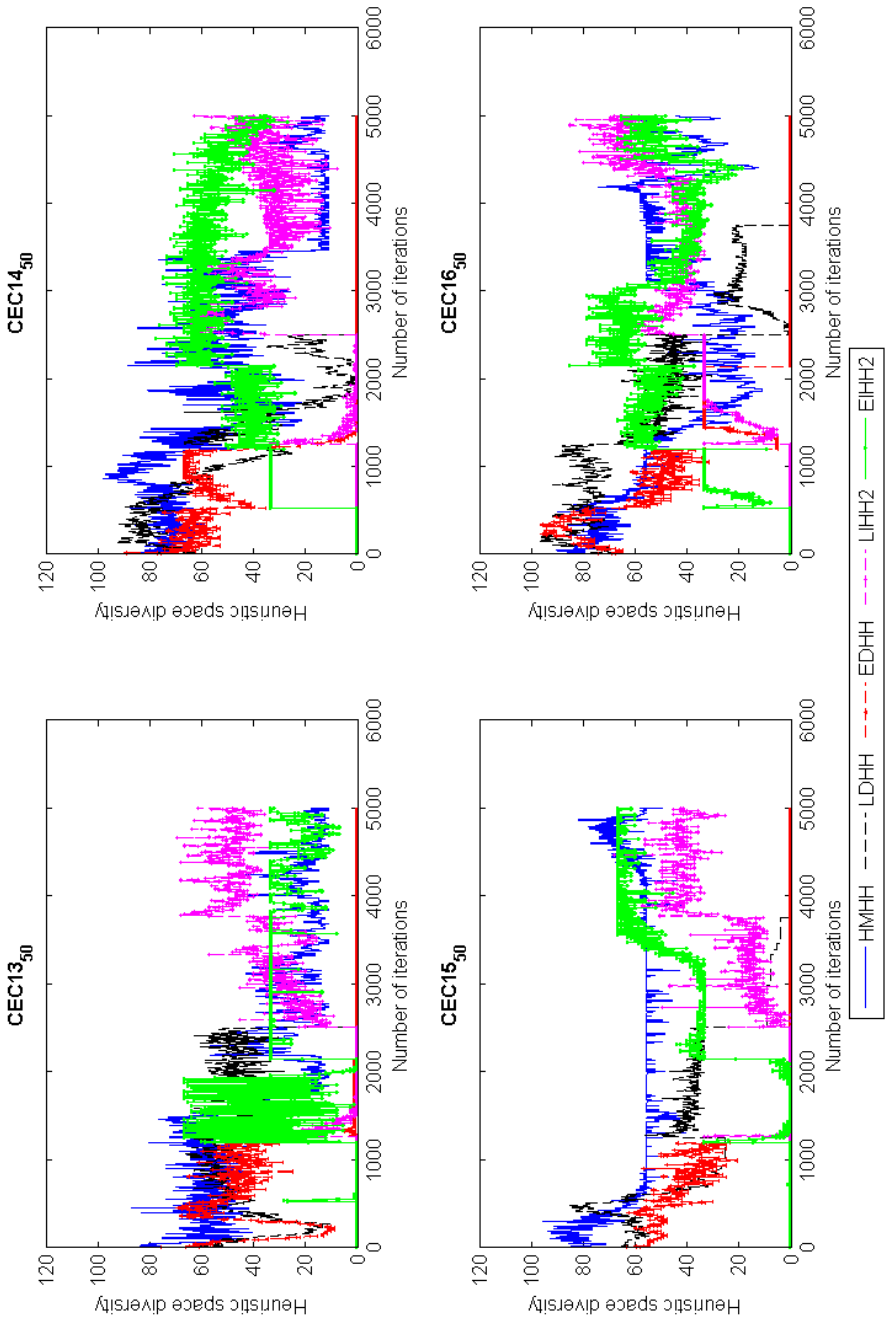

Figure 5.3: Graphs of heuristic space diversity (y-axes) versus iterations (x-axes) on a selected set of problems from the CEC 2005 benchmark problems in 50 dimensions.

Chapter 6

Benchmarking the Heterogeneous Meta-hyper-heuristic

An important step in the development of a new optimization algorithm, is benchmarking the new algorithm against existing similar algorithms. Such a comparative analysis is extremely valuable in determining the contribution made by the newly developed algorithm and determining its strengths and weaknesses relative to existing state-of-the-art algorithms of the same class. Section 6.1 describes the relevant benchmark multi-method algorithms selected for comparison purposes. The comparative analysis is described in Section 6.2 and the chapter is concluded in Section 6.3.

6.1 Investigating meta-hyper-heuristic performance versus other popular multi-method algorithms

Four multi-method algorithms, namely the population-based algorithm portfolio (PAP) of Peng *et al.* [122], the fitness-based area-under-curve bandit operator selection method (FAUC-Bandit) [55], the modified population-based genetic adaptive method for single objective optimization (AMALGAM-SO) [172], and the evolutionary algorithm based on self-adaptive learning population search techniques (EEA-SLPS) [177] were selected for evaluation in this chapter. As of the writing of this thesis, these algorithms were identified to be the most recent and most successful algorithms in the portfolio algorithm,

ensembles, and adaptive operator selection literature. Each of the identified algorithms were implemented with the same set of low level meta-heuristics (LLMs) used throughout the thesis to ensure that the algorithms are compared against a similar baseline. The rest of this section describes the selected algorithms in more detail.

6.1.1 Population-based algorithm portfolio

Peng *et al.* [122] developed the population-based algorithm portfolio (PAP). Their work focuses on reducing the risk of poor algorithm performance by dividing the number of allowable function evaluations between a portfolio of available algorithms. As described in Algorithm 6.1, entities are divided into subpopulations which are adapted in parallel by an assigned constituent algorithm or LLM, where one LLM is used per subpopulation. Each entity has access to only the other entities within the same subpopulation in order to prevent the same genetic material from being adapted repeatedly. At pre-specified migration time intervals, κ, n_q entities are migrated between subpopulations to ensure effective information sharing between the different optimization algorithms.

This information sharing mechanism is described in more detail in Figure 6.1, which describes the migration procedure applied to subpopulation $\boldsymbol{P}_1$, where n_q is three and the number of subpopulations is four. An interim subpopulation $\boldsymbol{P}'_1$ is first created as a copy of $\boldsymbol{P}_1$. The best entity from each of the subpopulations $\boldsymbol{P}_2$ to $\boldsymbol{P}_4$ is then added to $\boldsymbol{P}'_1$. These entities are indicated in blue. The worst three (n_q) entities, indicated in red, are then deleted from $\boldsymbol{P}'_1$. Finally, $\boldsymbol{P}'_1$ replaces $\boldsymbol{P}_1$ before the migration procedure is applied to subpopulation $\boldsymbol{P}_2$, and then $\boldsymbol{P}_3$ and $\boldsymbol{P}_4$.

It should be noted that PAP makes use of a static algorithm selection mechanism. The allocation of entities to LLMs are only performed once at the start of the optimization process and the allocation remains the same throughout the optimization process. This implies that the algorithm is stuck with its initial choices regardless of how the search space and the suitability of the algorithms change over time. The HSD of PAP thus also remains constant throughout the optimization run.

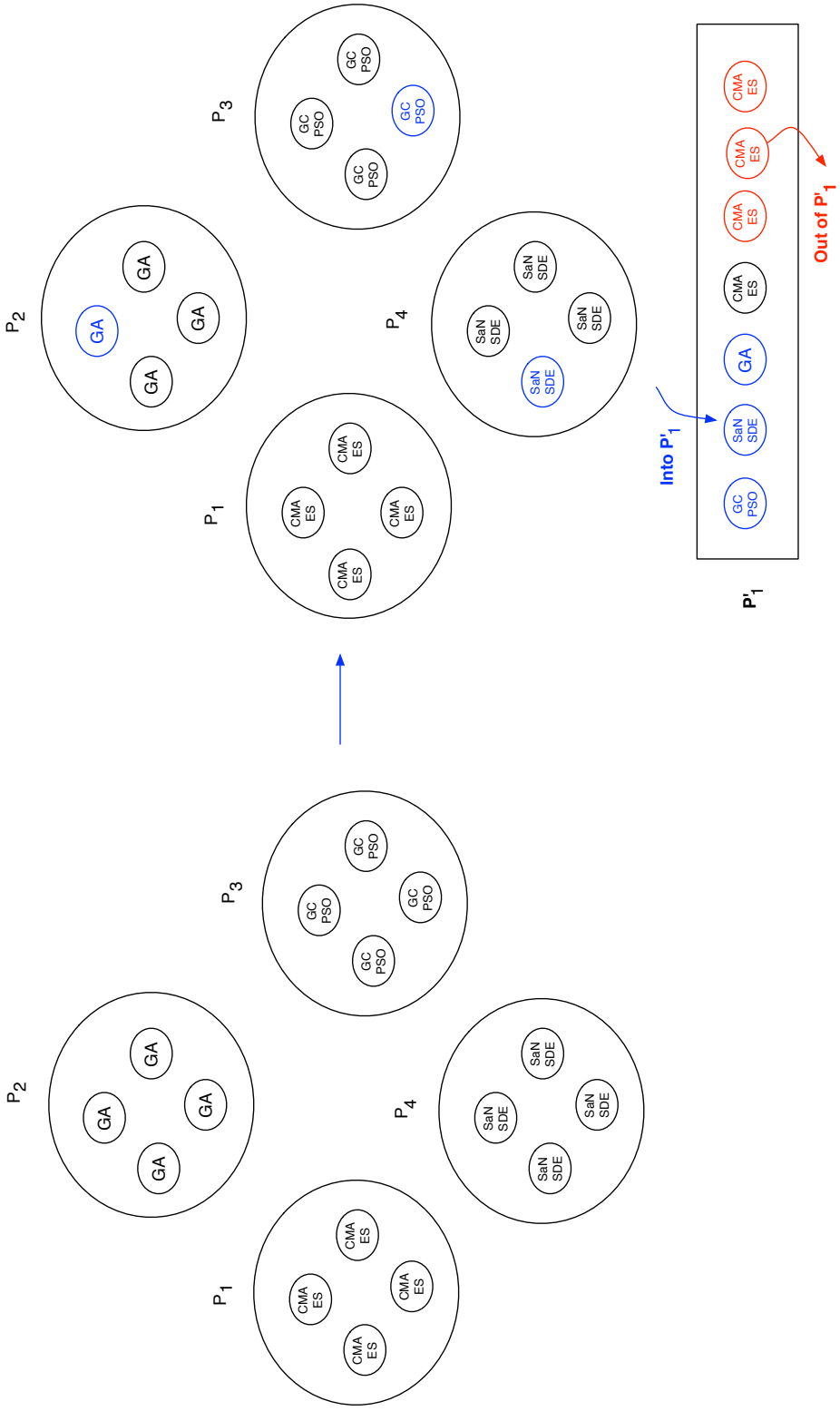

Figure 6.1: The PAP algorithm [122].

Initialize n_p subpopulations, $\boldsymbol{P}_1, \boldsymbol{P}_2, ..., \boldsymbol{P}_{n_p}$
while *no stopping condition is satisfied* **do**
  Evaluate all the entities in $\boldsymbol{P}_1, \boldsymbol{P}_2, ..., \boldsymbol{P}_{n_p}$
  for *all algorithms* m **do**
    Adapt $\boldsymbol{P}_m$ using algorithm m
  end
  if *migration interval, after* κ *iterations, is reached* **then**
    Activate migration procedure as follows:
    for *all subpopulations* m **do**
      For $\boldsymbol{P}_m$, select n_q best individuals from the other $n_p - 1$ subpopulations denoted as set $\boldsymbol{\Lambda}_m$
      Set $\boldsymbol{P}'_m$ to $\boldsymbol{P}_m \cup \boldsymbol{\Lambda}_m$
      Discard the n_q worst individuals in $\boldsymbol{P}'_m$
    end
    for *all populations* m **do**
      Set $\boldsymbol{P}_m$ to $\boldsymbol{P}'_m$
    end
  end
end

Algorithm 6.1: The PAP algorithm [122].

6.1.2 The evolutionary algorithm based on self-adaptive learning population search techniques

Similar to PAP [122], the EEA-SLPS algorithm [177] consists of entities divided into subpopulations. These subpopulations are adapted in parallel by an assigned LLM, where one LLM is used per subpopulation. Each entity has access to only the other entities within the same subpopulation in order to prevent the same genetic material from being adapted repeatedly. However, an information exchange mechanism is used to ensure that each LLM benefits from the learning of the other algorithms. A strong focus of Xue *et al's* [177] work was the investigation of alternative information exchange mechanisms and their impact on portfolio performance. Eighteen mechanisms were evaluated and the best strategy was identified as replacing the worst individual of each

subpopulation by the current best individual of the entire ensemble. This replacement was found to work best at each iteration as indicated in Algorithm 6.2.

Let $\boldsymbol{x}^*(t)$ be the best solution in the entire portfolio at time t.
Initialize n_p subpopulations, $\boldsymbol{P}_1, \boldsymbol{P}_2, ..., \boldsymbol{P}_{n_p}$
while *no stopping condition is satisfied* **do**
  Evaluate all the entities in $\boldsymbol{P}_1, \boldsymbol{P}_2, ..., \boldsymbol{P}_{n_p}$
  for *all algorithms* m **do**
    Adapt $\boldsymbol{P}_m$ using algorithm m
  end
  Activate migration procedure as follows:
  for *all subpopulations* m **do**
    if $\boldsymbol{x}^*(t) \in \boldsymbol{P}_m$ **then**
      Replace the worst solution in $\boldsymbol{P}_m$ by $\boldsymbol{x}^*(t)$
    end
  end
end

Algorithm 6.2: The EEA-SLPS algorithm [177].

6.1.3 The population-based genetic adaptive method for single objective optimization

The single objective version of the highly successful AMALGAM-SO is the third algorithm that was investigated. Similar to the HMHH algorithm, AMALGAM-SO also uses a common population of entities that are adapted over time. At the start of the optimization run, entities are assigned to LLMs. The entities are then adapted by their allocated LLMs until the population reaches "convergence". "Convergence" is obtained by meeting any one of a complex set of stopping conditions and can require a significant number of iterations. When this state is reached, the population size is first doubled, then the number of entities allocated to each LLMs is reconsidered. Each LLM's performance since the previous update is used to determine the number of entities assigned to

it according to the following equation:

$$n_m(t) = \lfloor n_s \frac{Q_{\delta m}(t)}{\sum_{m=1}^{n_a} Q_{\delta m}(t)} \rfloor, \tag{6.1}$$

where $n_m(t)$ is the number of entities allocated to the m^{th} LLM at time t and $Q_{\delta m}(t)$ is the total improvement in fitness function value of all entities assigned to the m^{th} LLM from the previous update of $n_m(t)$ to iteration t. This process of "convergence" and update of entities continues until the maximum number of iterations i_{max} is reached. For the sake of completeness, the algorithm pseudocode is provided in Algorithm 6.3.

Two sets of stopping conditions are used to determine whether the population has converged. The first set is used when the population is dominated by CMAES, in other words $n_1(t) > n_2(t) + n_3(t) + n_4(t)$, where $n_1(t)$ denotes the number of entities assigned to CMAES and $n_2(t)$, $n_3(t)$, and $n_4(t)$ are the entities assigned to the other LLMs. As described in [172], the first set of stopping conditions are as follows:

- Stop if the range of the best objective function values of the last $10 + \lfloor \frac{30 n_x}{n_s} \rfloor$ generations is zero, where n_x denotes the number of dimensions and n_s is the population size, or the ratio of the range of the current function values to the maximum current function value is below 10^{-5}.
- Stop if the standard deviation of the normal distribution is smaller than $10^{-9}\sigma(0)$ in all coordinates, where $\sigma(t)$ denotes the step size of the CMAES algorithm at time t, and the CMAES evolution path described in Equation (2.10) is smaller than $10^{-9}\sigma(0)$ in all components.
- Stop if adding a $0.1\boldsymbol{\sigma}(t)$ in a principal axis direction of the covariance matrix, $\boldsymbol{C}(t)$, does not change $\boldsymbol{w}_x(t)$, the centre of mass of $\boldsymbol{x}(t)$.
- Stop if adding $0.2\boldsymbol{\sigma}(t)$ in each coordinate does not change $\boldsymbol{w}_x(t)$.
- Stop if the condition number of the covariance matrix, $\boldsymbol{C}(t)$, exceeds 10^{14}.

The second set of stopping conditions is used when the population is not dominated by CMAES, in other words $n_1(t) \leq n_2(t) + n_3(t) + n_4(t)$:

Let t_{max} be the maximum number of iterations remaining in the optimization run
Let $n_m(t)$ be the number of entities allocated to the m^{th} LLM at time t
Let $\boldsymbol{R}(t)$ be the combined parent and child populations at time t
$t = 0$
Randomly initialize $n_m(1), \forall m$
while *no stopping condition is satisfied* **do**
  CONVERGED = FALSE
  Initialize the parent population $\boldsymbol{X}(t)$ of size n_x by means of a latin hypercube sampling strategy (Algorithm 6.4)
  while *algorithm has not converged (CONVERGED = FALSE)* **do**
    Create offspring population $\boldsymbol{c}(t+1)$ by applying A to $\boldsymbol{X}(t)$
    Combine parent and offspring populations, $\boldsymbol{X}(t)$ and $\boldsymbol{c}(t+1)$, and sort in order of decreasing fitness to obtain $\boldsymbol{R}(t+1)$
    Select $\boldsymbol{X}(t+1)$ from $\boldsymbol{R}(t+1)$ using the species selection mechanism from Section 5.2
    if *any of the stopping criteria have been met* **then**
      CONVERGED = TRUE
    else
      $t = t + 1$
    end
    if *CONVERGED = TRUE* **then**
      Update $n_m(t), \forall m$ according to Equation (6.1)
      Increase population size n_s to $2n_s$
      Reset $t = 0$ and recompute t_{max}
    end
  end
end

Algorithm 6.3: The AMALGAM-SO algorithm [172].

- Stop if the standard deviation of the best individual, $\boldsymbol{x}^*(t)$, is smaller than $10^{-9}\sigma(0)$ in all coordinates.
- Stop if the ratio of the range of the best function values found so far to the maxi-

mum function value of $\boldsymbol{x}^*(t)$ is below 10^{-5}.

- Stop if the acceptance rate of new solutions in $\boldsymbol{x}^*(t)$ is lower than the acceptance rate, υ. In this study, υ was set to $7.5 \times 10^{-2} n_s$.
- Stop if the range of the best objective function values of the last $50 + \lfloor \frac{100 n_x}{n_s} \rfloor$ generations is zero.

It is important to note that the purpose of this chapter is to investigate the mechanisms used to allocate entities to LLMs. When an algorithm such as AMALGAM-SO is considered, a large number of different algorithmic aspects lead to the success of the algorithm. However, if an objective evaluation of the entity-to-algorithm allocation mechanisms of different algorithms is to be performed, it is necessary to isolate the entity-to-algorithm allocation mechanism. It was thus decided to implement a modified AMALGAM-SO without any performance enhancing aspects not directly related to the entity-allocation mechanism. Algorithmic aspects removed include the species selection mechanism which operates directly on the solution space independently from the heuristic space and the latin hypercube sampling strategy used to initialize a diverse set of initial solutions (Algorithm 6.4).

Create large initial sample, $\boldsymbol{S}(0)$, in normalized search space
Set $\boldsymbol{X}(0)$ to $\boldsymbol{S}(0)\{1\}$
while $|\boldsymbol{X}(0)| < n_s$ **do**
  for *all* $s \in \boldsymbol{S}(0)$ **do**
    for *all* $x \in \boldsymbol{X}(0)$ **do**
      $D(x, s) = \sqrt{\sum_{i=1}^{n_x} (x_i - s_i)^2}$
    end
  end
  Find member r of $\boldsymbol{S}(0)$ with maximum distance to members of $\boldsymbol{X}(0)$
  $\boldsymbol{X}(0) \cup r$
end

Algorithm 6.4: Latin hypercube sampling strategy [172].

A number of aspects need to be addressed with regards to this modified algorithm's performance. Firstly, note that the AMALGAM-SO implementation in [172] is significantly biased towards CMAES. CMAES is well known to be one of the best performing algorithms on the CEC 2005 benchmark problem set [3]. CMAES is provided an unfair advantage by ensuring that between 80% and 90% of the entities in the initial population is allocated to CMAES. During the optimization run it is also always ensured that a minimum of 25% of entities is allocated to CMAES at all times. The minimum number of entities allocated to the other algorithms are set to only 5% of the population size. In the modified AMALGAM-SO this inherent bias was removed by dividing the entities equally between all four the available LLMs at the start of the optimization run and setting the minimum number of entities per algorithm equal for all algorithms.

6.1.4 Fitness-based area-under-curve multi-armed bandit

The fourth algorithm considered is an online operator selection strategy from the adaptive operator selection field [40]. The FAUC-Bandit algorithm [54] consists of a credit assignment mechanism and a LLM selection mechanism as indicated in Figure 6.2. The credit assignment mechanism evaluates the performance of each of the available LLMs based on previous successes and assigns a credit to each LLM. The LLM selection mechanism then utilizes these LLM credits in a dynamic bandit-based LLM selection mechanism to select which LLM should be applied to the entity under consideration.

More specifically, the fitness-based area-under-curve credit assignment mechanisms consider a list of entities ranked according to the greatest improvement obtained, over a given time window W. A receiving operator curve (ROC) is plotted for each LLM by scanning the ordered list of entities. Starting from the origin of the curve and the highest ranking entity, a vertical segment is plotted for every entity that has been generated by the LLM under consideration. If an entity has not been generated by the specific LLM, a horizontal segment is drawn. Ties are resolved by means of diagonal segments. A decay factor, D, is used to bias the selection towards the higher ranked entities. If ϱ_i is the rank position of entity i, the length of the i^{th} segment is defined as $D^{\varrho_i}(W - \varrho_i)$ with $D \in [0, 1]$. Finally, a ROC such as the example in Figure 6.3 is obtained for each algorithm. The area under the ROC of the m^{th} LLM at time t, namely $q_m(t)$ is then

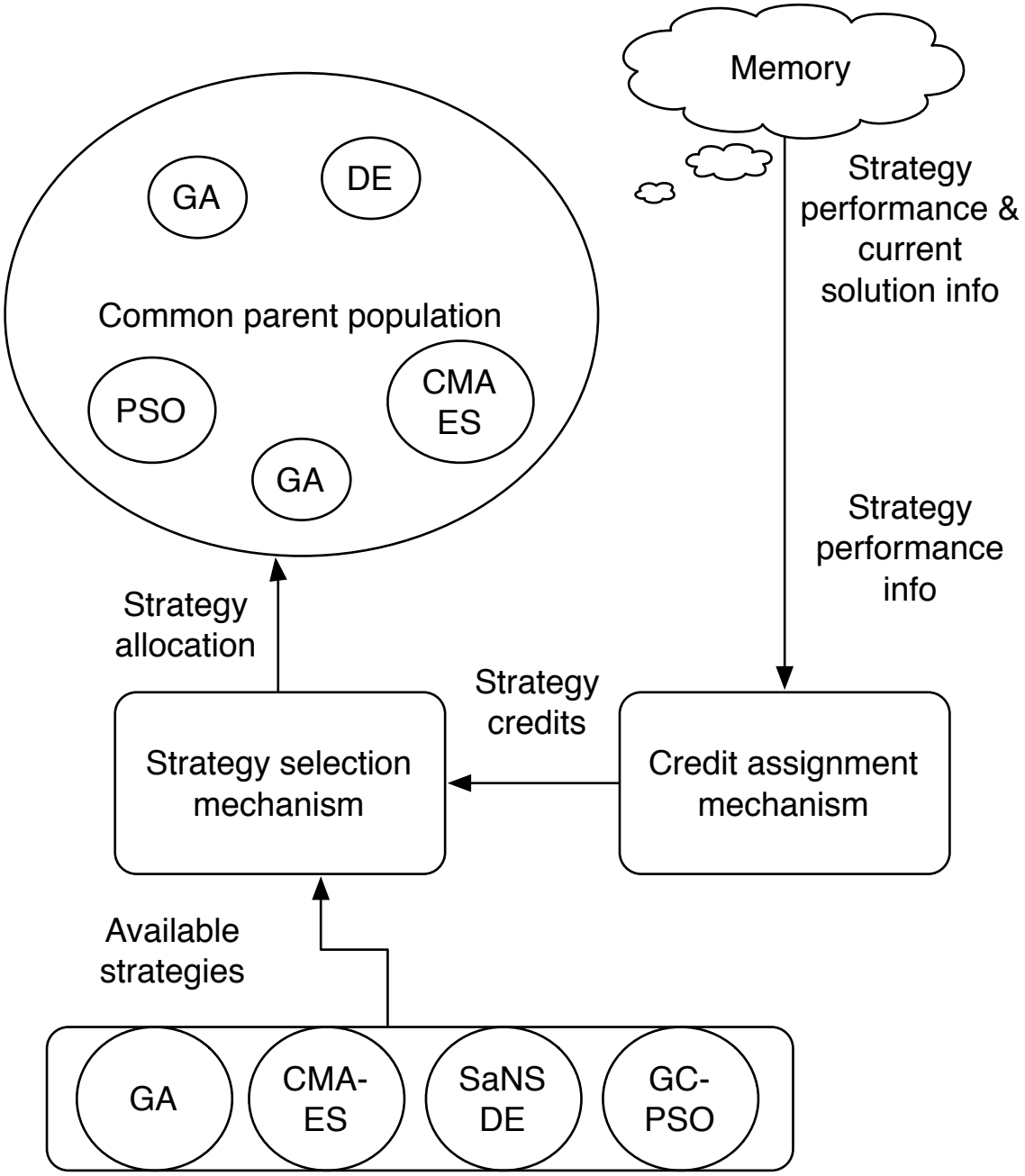

Figure 6.2: The fitness-based area-under-curve bandit algorithm [54].

used as the credit associated with the LLM under consideration.

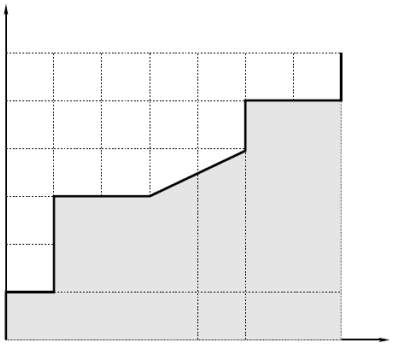

Figure 6.3: ROC curve of ranked entities (x-axis) versus number of entities generated by the LLM under consideration (y-axis) [54].

The LLM selection mechanism is based on a maximization operator:

$$q_m(t) + C\sqrt{\frac{2\log\sum_k \nu_k(t)}{\nu_m(t)}}, \tag{6.2}$$

where C is a user-defined constant balancing exploration and exploitation and $\nu_m(t)$ refers to the number of times the m^{th} LLM has been used up until time t. The LLM with the largest maximization operator is applied to the entity under consideration. For the sake of completeness, high level pseudocode of the algorithm is provided in Algorithm 6.5.

One of the main criticisms of the FAUC-Bandit approach is that the algorithm is extremely slow due to the frequent updating of a computationally complex entity-to-algorithm allocation procedure.

Initialize the parent population $\boldsymbol{X}$
Randomly select an initial LLM $A_i(1)$ from the set of LLMs to apply to the first entity.
while *no stopping condition is satisfied* **do**
  Adapt entity i using LLM $A_i(t)$
  Calculate $Q_{\delta A_i(t)}(t)$, the total improvement in fitness function value of entity i after application of algorithm $A_i(t)$ and add entity-to-LLM allocation information to archive $\boldsymbol{A}$ of size W
  Sort all entries in $\boldsymbol{A}$ according to greatest improvement obtained
  for *all LLMs* m **do**
    Calculate the area under the ROC curve, $q_m(t)$.
    Calculate the maximization value of algorithm m as defined in Equation (6.2).
  end
  Allocate entity $i+1$ to the algorithm m_{i+1} which maximizes Equation (6.2).
  $i = i + 1$
end

Algorithm 6.5: The FAUC-Bandit algorithm [122].

6.2 Comparative analysis of selected multi-method algorithms

The same experimental setup used throughout the thesis was again employed to analyse the performance of the four multi-method algorithms versus the best performing HMHH algorithm from Chapter 5, EIHH1, and the baseline HMHH algorithm. The comparison focused on two aspects. Firstly, the performance of each multi-method algorithm against its four LLMs were evaluated. The idea was to evaluate the performance benefits obtained by each multi-method algorithm when compared against the single-method results of its best performing LLM. Secondly, the relative performance of each of the multi-method algorithms were evaluated.

The control parameters of the six multi-method algorithms used are specified in Table 6.1. Modified AMALGAM-SO, EEA-SLPS, and PAP's parameters were optimized for the CEC 2005 problems by the original authors, thus these parameters were used as-is. For the same reason the LLM control parameters specified in the original

papers ([3],[179]) were used. In the original work, the FAUC-Bandit algorithm's parameters were, however, optimized for the BBOB-2010 noiseless benchmarking problem suite. F-race [10] was thus used to tune the FAUC-Bandit algorithm's control parameters specifically for the 2005 IEEE CEC benchmark problem set. Similar to the analysis of Fialho *et al.* [55], the following parameter values were evaluated: scaling factor $C \in \{0.1, 0.5, 1, 5, 10\}$, decay factor $D \in \{0.5, 1\}$, and window size $W \in \{50, 100, 500\}$, resulting in the evaluation of 30 configurations. The final values for C, W, and D obtained by F-race are also provided in Table 6.1. A similar strategy was used to tune the number of iterations between re-allocation, k, of the HMHH algorithms. Values of $k \in \{1, 2, 5, 10, 20, 50, 100, 500, 1000\}$ were evaluated.

The results of the first experiment, focusing on the benchmark algorithms versus their LLMs, are presented in Tables B.13 to Tables B.17. The Mann-Whitney U test results obtained when each algorithm was compared to each one of its LLMs are recorded in Table 6.2. The results indicate that EIHH1 was the best performing multi-method algorithm when evaluated against its LLMs, with 129 wins, 47 draws, and 28 losses. PAP and EEA-SLPS also performed well with 128 wins, 27 draws, and 49 losses, and 110 wins, 38 draws, and 56 losses, respectively. The fourth best performing algorithm was the baseline HMHH algorithm which outperformed all other algorithms 106 times out of 204 cases. The modified AMALGAM-SO and the FAUC-Bandit algorithm struggled to outperform the set of selected LLMs and only outperformed the GA and the GCPSO algorithm a significant number of times.

It should be noted that the first four benchmark algorithms outperformed three of their four LLMs, but struggled to outperform CMAES. This poor performance can, however, be expected since a portion of the function evaluation budget of a multi-method algorithm needs to be allocated to solve the algorithm selection problem. A larger number of function evaluations, when compared to a single-method algorithm, is required to solve the harder optimization problem of determining which entities to allocate to which LLMs in addition to solving the actual optimization problem. The inefficiency of the multi-method algorithms is then understandable since computational resources are required to first "learn" which LLM is the best algorithm for the problem at hand. This is in contrast to CMAES which uses the entire function evaluation budget on optimization of the actual

Table 6.1: Algorithm parameters.

Parameter	Value used
Common algorithm parameters	
Population size (n_s)	100
Maximum number of iterations (I_{max})	$100n_x$
HMHH and EIHH1	
Number of iterations between re-allocation (k)	5
Size of HMHH tabu list	3
Size of EIHH1 tabu list	2 if $n_a = 4$; 1 if $n_a = 3$ and 0 if $n_a <= 2$
PAP and EEA-SLPS	
Number of entities assigned to CMAES	14
Number of entities assigned to GCPSO	18
Number of entities assigned to GA	18
Number of entities assigned to SaNSDE	50
PAP Migration interval (κ_1)	$I_{max}/20$
EEA-SLPS Migration interval (κ_2)	1
Number of entities involved in PAP migration (n_{q1})	3
Number of entities involved in EEA-SLPS migration (n_{q2})	1
Modified AMALGAM-SO	
Refer to Section 6.1.3	
FAUC-Bandit	
Size of time window (W)	50
Decay factor (D)	0.5
Exploration-exploitation constant (C)	1

problem. It is encouraging, however, to note that EIHH1 performed significantly better against CMAES than any of the other evaluated algorithms.

Furthermore, a closer inspection of the results showed that the good performance of CMAES could be mostly attributed to the first five uni-modal problems. EIHH1 did

Table 6.2: Hypotheses analysis of HMHH, EIHH1, modified AMALGAM-SO, PAP, EEA-SLPS, and FAUC-Bandit versus their LLMs.

	CMAES	SaNSDE	GCPSO	GA	TOTAL
HMHH	2-6-43	23-10-8	40-6-5	41-7-3	106-29-69
EIHH1	2-34-15	35-8-8	48-2-1	44-3-4	129-47-28
PAP	4-7-40	28-18-5	47-1-3	49-1-1	128-27-49
EEA-SLPS	4-8-39	18-20-13	46-2-3	42-8-1	110-38-56
AMALGAM-SO	5-2-44	10-11-30	25-13-13	33-12-6	73-38-93
FAUC-Bandit	2-5-44	1-8-42	19-9-23	24-11-16	46-33-125

show improved performance in comparison with CMAES as problem complexity with regards to multimodalism increased. An inspection of the entity-to-algorithm allocation also indicates that both the EIHH1 and HMHH algorithms were able to, in most cases, identify either CMAES or SaNSDE as the best performing algorithms and bias the search towards them as the optimization run progressed. Figures 6.4 and 6.5 provides examples of this behaviour. Note that SaNSDE significantly outperformed CMAES on this problem. This highlights the main advantage of the HMHH algorithm: identification of the best LLM and an inherent bias towards it resulting in good performance without *a priori* knowledge of the best LLM for the problem to be solved.

The various benchmark algorithms were also compared against each other. Table 6.3 uses Mann-Whitney U tests to compare each benchmark algorithm to each of the other benchmark algorithms. The results indicate that the EIHH1 algorithm is in fact the best performing algorithm out of the six benchmark algorithms evaluated. EIHH1 performed statistically better than the other five benchmark algorithms in 129 out of the 255 tests conducted. The second best performing algorithm was PAP with 140 wins, 60 draws, and 45 losses. PAP was followed, in worsening order, by HMHH, EEA-SLPS, the modified AMALGAM-SO, and FAUC-Bandit.

The HSD of the benchmark algorithms for problem 17 from the CEC 2005 problem set in 10, 30 and, 50 dimensions is plotted in Figure 6.6. Only the graphs of this problem are provided due to space constraints. The graphs of the other problems are similar and

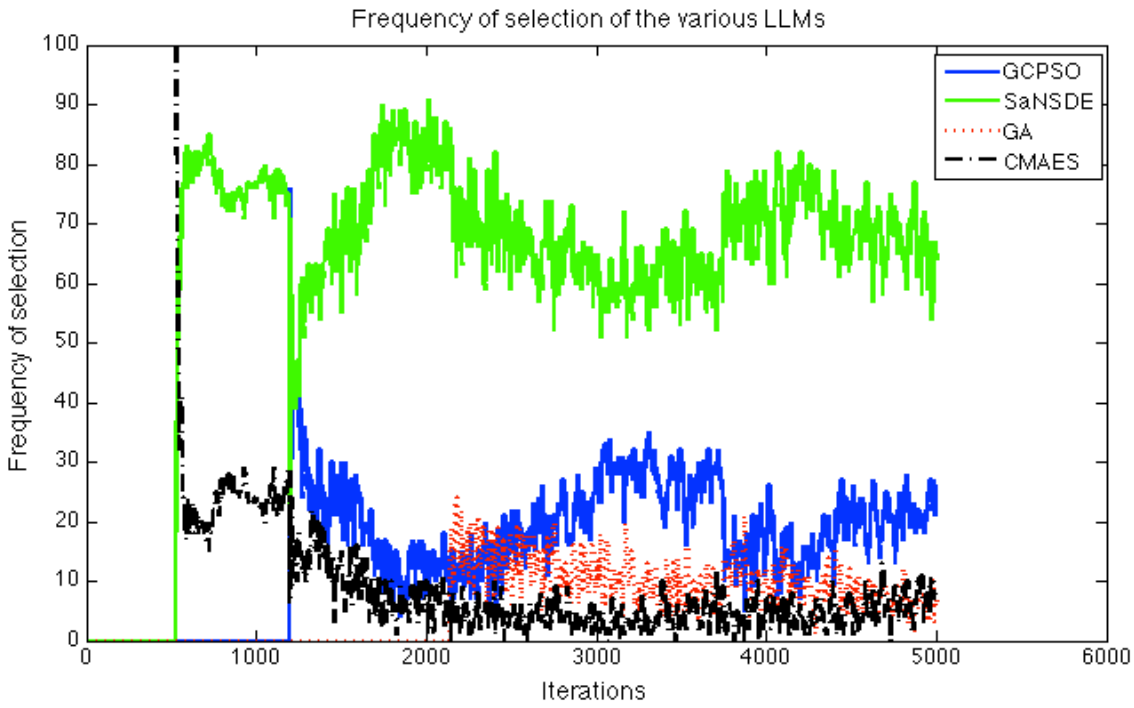

Figure 6.4: Frequency of use of each of the LLMs in the EIHH1 algorithm on the 12^{th} CEC 2005 problem in 50 dimensions. Frequency of use is determined by the number of entities allocated to the LLM under consideration per iteration.

are provided in Appendix C.

As can be seen in Figure 6.6, PAP and EEA-SLPS both maintained a constant heuristic space diversity throughout the optimization run. Although an information exchange mechanism is used and entities are in effect re-allocated to different algorithms throughout the optimization run, the number of entities allocated to each algorithm remained the same. The advantage of maintaining a high heuristic space diversity for a longer period of time is greater exploration of the heuristic space. The algorithms are, however, never allowed to converge to a point where all entities are allocated to a single LLM which could affect the results obtained.

The modified AMALGAM-SO only updates the entity-to-algorithm allocation once the algorithm has met the stopping conditions defined in Section 6.1.3. HSD thus remains constant over time until the population "converges" and then the HSD drops rapidly

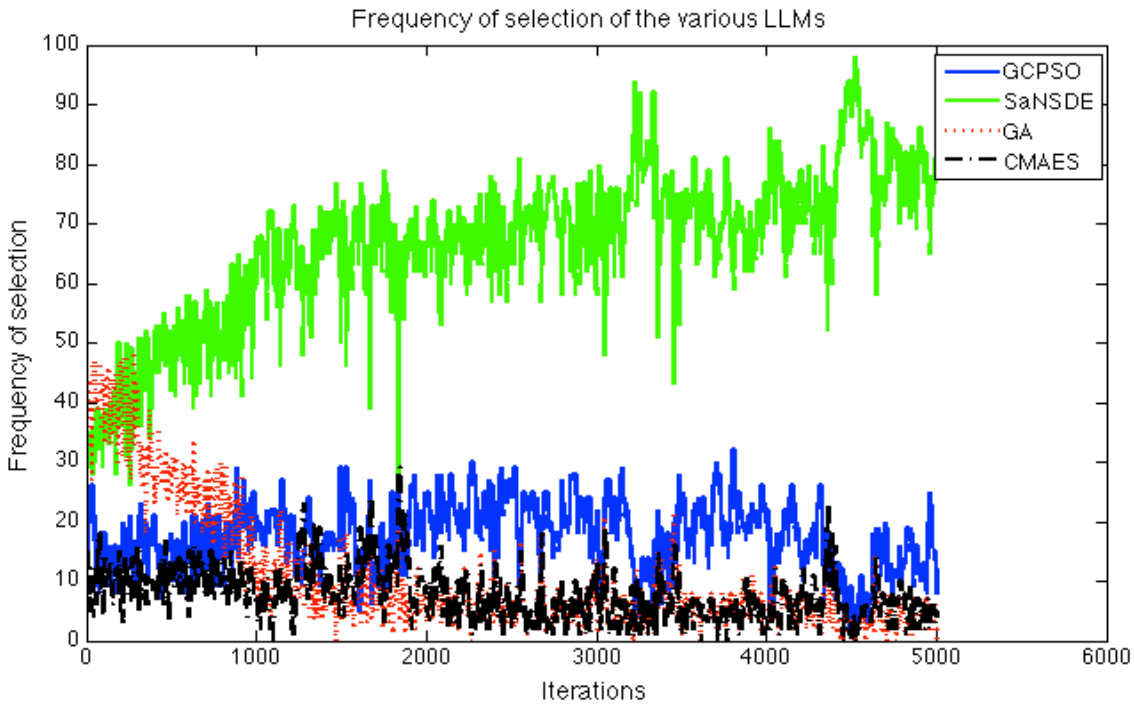

Figure 6.5: Frequency of use of each of the LLMs in the HMHH algorithm on the 12^{th} CEC 2005 problem in 50 dimensions. Frequency of use is determined by the number of entities allocated to the LLM under consideration per iteration.

when algorithms are re-allocated to entities. This rapid decrease in HSD can be seen at around iterations 1100, 1150, 1750, and 2000 for problem CEC17_{10}. It should also be noted that, because the modified AMALGAM-SO makes use of a growing population size, more iterations are allowed to ensure comparison with the other algorithms over the same number of function evaluations. This explains the later termination of AMALGAM-SO that can be seen in the graph of problem CEC17_{10} and some of the other problem graphs in Appendix C.

Even though the modified AMALGAM-SO started off with a significantly high heuristic space diversity, no knowledge exchange mechanism exists between the re-allocation of entities to algorithms. This implies that the separate entity-to-algorithm allocation groups did not benefit from the learning of the other entity-to-algorithm allocation groups. This lack of knowledge sharing is suspected to be a definite contributor to

the poor results obtained.

FAUC-Bandit does not take any information into account between iterations with regards to the heuristic space diversity of the population, resulting in a sporadic HSD over time with no clear strategy with regards to exploration and exploitation of the heuristic space. As discussed in the previous chapter, no intervention is made with regards to HSD in the HMHH algorithm. EIHH1 makes use of an exponentially increasing HSD which along with the *a priori* knowledge of single-method LLM ranking on the benchmark problem set, shows great promise.

6.3 Summary

In this chapter, six popular multi-method algorithms, namely PAP, FAUC-Bandit, a modified AMALGAM-SO, EEA-SLPS, EIHH1, and HMHH have been investigated. The same set of LLMs were used for each algorithm and a comparison was performed between the multi-method benchmark algorithms and each individual LLM. The multi-method algorithms were also compared against each other. EIHH1 was shown to outperform the other five multi-method algorithms and also performed well relative to its LLMs.

Table 6.3: Hypotheses analysis of the multi-method algorithms when compared to each other.

	HMHH	EIHH1	PAP	EEA-SLPS	AMALGAM-SO	FAUC-Bandit	TOTAL
HMHH	NA	6-11-34	15-14-22	19-17-15	29-11-11	39-10-2	108-63-84
EIHH1	34-11-6	NA	34-8-9	35-8-8	38-3-10	44-5-2	185-35-35
PAP	22-14-15	9-8-34	NA	24-26-1	39-7-5	46-5-0	140-60-55
EEA-SLPS	15-17-19	8-8-35	1-26-24	NA	31-13-7	42-7-2	97-71-87
AMALGAM-SO	11-11-29	10-3-38	5-7-39	7-13-31	NA	27-11-13	60-45-150
FAUC-Bandit	2-10-39	2-5-44	0-5-46	2-7-42	13-11-27	NA	19-38-198

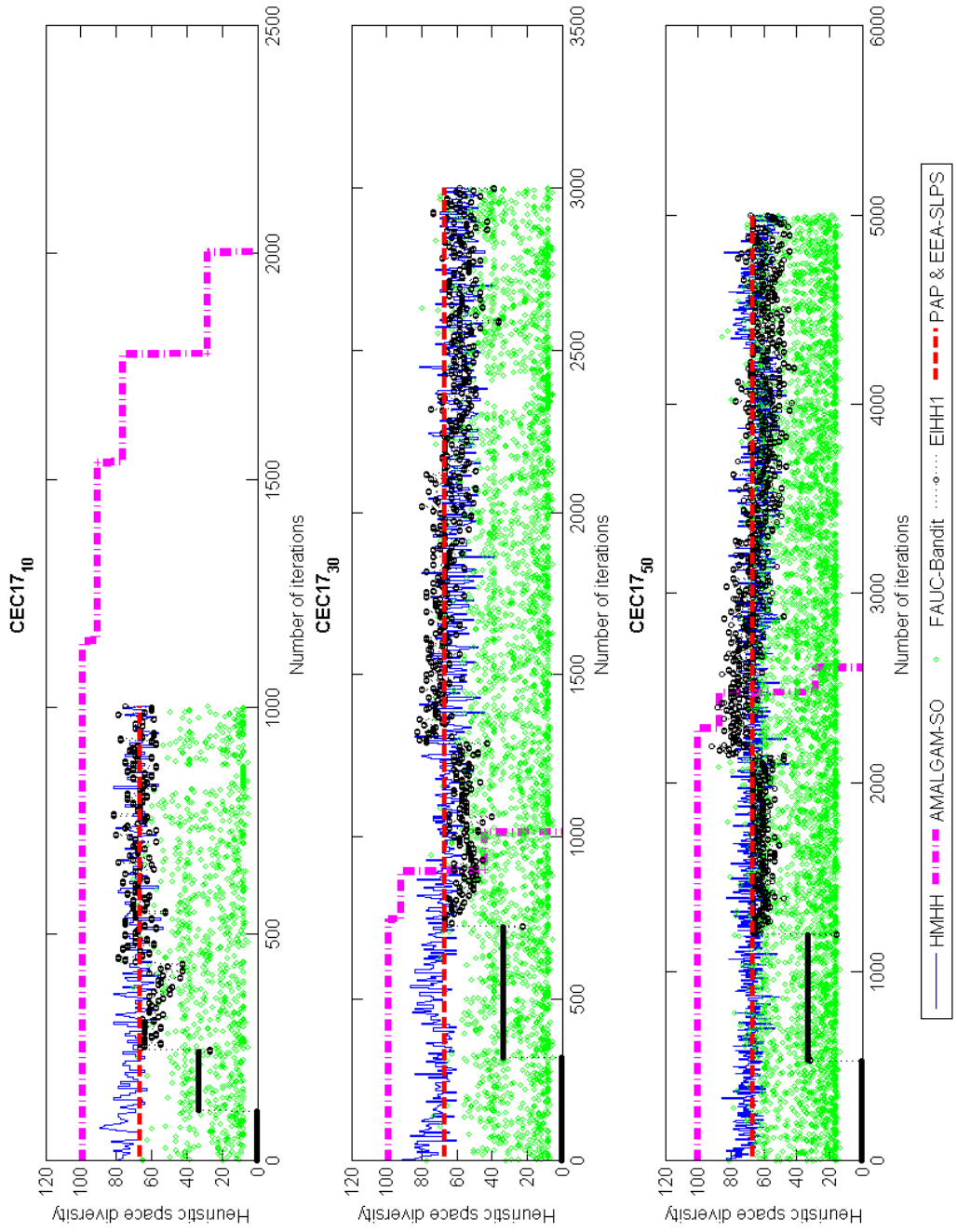

Figure 6.6: Comparison of heuristic space diversity (y-axes) versus iterations (x-axes) for HMHH, EIHH1, PAP, EEA-SLPS, modified AMALGAM-SO, and FAUC-Bandit on problem 17 of the CEC 2005 benchmark problems in 10, 30, and 50 dimensions.

Chapter 7

Conclusion

This thesis touched on a number of issues of interest in using meta-heuristics as low level algorithms in a hyper-heuristic context ranging from an evaluation of alternative low level meta-heuristic (LLM) selection methods to the use of diversity management to obtain improved performance. This chapter summarizes the main findings of this study before identifying and discussing opportunities for future research.

7.1 Summary

The purpose of this thesis was to investigate the use of meta-heuristics as low level algorithms in a hyper-heuristic context. An analysis of various state-of-the-art single-method optimization algorithms as well as the latest multi-method algorithm literature, led to the development of the heterogeneous meta-hyper-heuristic framework.

The development of an appropriate selection mechanism to perform the candidate solution-to-LLM allocation was considered to be one of the most important choices in the development of the meta-hyper-heuristic algorithm. Subsequently, six solution-to-LLM strategies inspired from the evolutionary algorithm (EA) and hyper-heuristic literature, were developed: the random selection strategy, the roulette-wheel selection strategy, the tournament selection strategy, the rank-based selection strategy, the Boltzman selection strategy, and the tabu search-based selection strategy. An empirical comparison of the six selection strategies resulted in the tabu search-based strategy being identified as the

most suitable.

After a suitable selection strategy was developed, the use of local search strategies to improve the performance of the meta-hyper-heuristic algorithm was investigated. Four strategies were developed: LS1HH, where local search is applied to the best solution at each iteration, LS2HH, where local search is applied to a random solution at each iteration, LS3HH, where roulette-wheel selection is used to select a solution to undergo local search, and LS4HH, where local search is added to the set of LLMs. LS1HH, LS2HH, and LS3HH performed statistically significantly better than LS4HH. At this stage of the algorithm development process, the set of LLMs was redefined since the influence of this set on hyper-heuristic performance was shown to be significant.

After the basic HMHH algorithm configuration was established, the performance benefits of diversity management in a hyper-heuristic framework was investigated. Constant and adaptive solution space diversity management strategies were proposed which utilized either a local search for reducing solution space diversity (SSD) or a species selection mechanisms for increasing solution space diversity. The results, however, indicated that additional solution space diversity intervention mechanisms had a relatively insignificant impact on hyper-heuristic performance.

Hyper-heuristics, however, lend themselves to another type of diversity control, namely the management of heuristic space diversity (HSD). The concept of heuristic space diversity was explored and a HSD metric was proposed. Six HSD control strategies were then proposed and compared. These strategies include the linearly decreasing HSD strategy (LDHH), the exponentially decreasing HSD strategy (EDHH), the linearly increasing HSD strategy with *a priori* knowledge, the exponentially increasing HSD strategy with *a priori* knowledge, the linearly increasing HSD strategy with no *a priori* knowledge, and the exponentially increasing HSD strategy with no *a priori* knowledge. The results of the comparison indicated that a significant performance improvement could be obtained by controlling the HSD of the HMHH algorithm. The exponentially increasing HSD strategy was shown to outperform the decreasing, linearly increasing, and uncontrolled HSD strategies. This performance improvement was even more significant with the availability of *a priori* knowledge of LLM performance on the benchmark problem set in question.

Finally, in a benchmarking exercise, the best performing HSD strategy, EIHH1, statistically significantly outperformed six state-of-the-art multi-method algorithms, namely the population-based algorithm portfolio (PAP) of Peng *et al.* [122], the fitness-based area-under-curve bandit operator selection method (FAUC-Bandit) [55], the modified population-based genetic adaptive method for single objective optimization (AMALGAM-SO) [172], and the evolutionary algorithm based on self-adaptive learning population search techniques (EEA-SLPS) [177]. The comparison was performed with each algorithm using the same set of LLMs to ensure a fair comparison and also included evaluations of each multi-method algorithm against its constituent LLMs.

Overall, it was shown that the use of meta-heuristics in a hyper-heuristic can lead to the development of an effective multi-method algorithm. A number of opportunities for improvement and future research does, however, exist and is listed in the next section.

7.2 Future research opportunities

The opportunities for future research are discussed in detail throughout the rest of this section.

Alternative LLMs:

The performance of a hyper-heuristic algorithm is dependent on the performance of its constituent algorithms. In this thesis, four LLMs were used in the final HMHH, namely a CMAES, SaNSDE, GCPSO, and a GA with floating-point representation, tournament selection, blend crossover and self-adaptive Gaussian mutation. As technology advances and the body of optimization research grows, more and more algorithms will be developed which will probably outperform the LLMs used in this thesis. A future update of the HMHH algorithm with new state-of-the-art meta-heuristics could lead to significant performance improvements.

Alternative benchmark problem sets:

In this thesis problems of 10 to 50 dimensions were considered. No conclusions can thus be drawn with regard to algorithm performance on problems of dimensions higher than 50. However, real world problems often have thousands of decision variables and recently various benchmark problem sets focusing on large scale global optimization have been

released. Investigating the performance of the HMHH algorithm on a more diverse set of larger benchmark problems including problems with 1000 dimensions, is an important future research area.

More in-depth analysis of the HMHH algorithm:
The HMHH algorithm significantly outperformed four state-of-the-art multi-method algorithms. A more in-depth investigation into the reasons behind this performance improvement could make for interesting future research.

Acceptance strategy:
The deterministic acceptance strategies "accept all moves" for the PSO and CMAES algorithms, and "improvement only" for the GA and DE algorithms were used in this thesis to accept the moves in solution space generated by the selected LLMs. A number of additional acceptance strategies were reviewed in Section 2.2.4. Simulated annealing [1, 6], late acceptance [15], and variants of threshold acceptance [101] acceptance strategies could also lead to improved performance.

Entity-to-algorithm allocation strategies based on prediction of future performance:
Most of the entity-to-algorithm allocation strategies investigated in this thesis were based on past LLM performance. However, further investigation of a prediction mechanism aimed at estimating future LLM performance and using this as input to the entity-to-algorithm allocation strategy could be a promising future research direction. Examples of this approach can be found in [157] and [183].

Variations on the heuristic space diversity metric:
The heuristic space diversity metric proposed in this thesis is directly based on the entity-to-algorithm allocation of the algorithm. Future work could focus on incorporating the behaviours or performance of the various LLMs into this metric. Using the complementarity definition of Peng *et al.* [122] and Tang *et al.* [157] as HSD metric could also be explored.

Adaptive heuristic space diversity management strategy:
All of the HSD strategies developed in this thesis made use of predetermined time intervals at which changes were made to the set of available LLMs. Instead of adjusting HSD at these predetermined time intervals, an adaptive strategy could also be employed

where HSD is modified based on specific characteristics of the solution, objective, and heuristic space.

More in-depth investigation of the nature of the heuristic space:
Efforts to analyze the heuristic space by means of landscape analysis have been conducted [113, 114, 118]. However, after 15 years of hyper-heuristic research, very little is still known about the nature of the heuristic space. Significant opportunities for future research still exists to understand the heuristic space and utilize the knowledge towards improved hyper-heuristic performance.

More in-depth investigation of the various HMHH algorithms' ability to address specific problem characteristics:
Further useful insights can be obtained by considering specific problem characteristics and the effect these characteristics have on algorithm performance. This can be achieved by grouping the benchmark problems according to predefined characteristics and then conducting experiments to test hypotheses about the effectiveness of different algorithms in dealing with specific problem characteristics.

Extension of the HMHH algorithm to constrained, multi-objective, and dynamic environments:
Many real world problems are characterized by constraints, a set of conflicting objective functions, and environments that change over time. Significant future research opportunities lie in extending the HMHH algorithm to function in these contexts.

Application of the HMHH algorithm to real-world problems:
The scope of this thesis was limited to continuous optimization problems and perturbative hyper-heuristics. However, due to the generality of hyper-heuristics the LLMs in, for example, the EIHH framework, can be replaced by domain specific heuristics embedded in either a constructive or perturbative hyper-heuristic. The EIHH hyper-heuristic selection strategy could then prove useful in other domains such as timetabling, scheduling, bin packing, vehicle routing and many other real world applications. An investigation into this hypothesis could be an excellent future research opportunity.

Bibliography

[1] C.H. Antunes, P. Lima, E. Oliveira, and D.F. Pires. A multi-objective simulated annealing approach to reactive power compensation. *Engineering Optimization*, 43(10):1063–1077, 2011.

[2] A. Auger and N. Hansen. Performance evaluation of an advanced local search evolutionary algorithm. In *Proceedings of the IEEE Congress on Evolutionary Computation*, volume 2, pages 1777–1784. IEEE, 2005.

[3] A. Auger and N. Hansen. A restart cma evolution strategy with increasing population size. *Proceedings of the IEEE Congress on Evolutionary Computation*, 2:1769–1776, 2005.

[4] M. Ayob and G. Kendall. A monte carlo hyper-heuristic to optimise component placement sequencing for multi head placement machine. In *Proceedings of the International Conference on Intelligent Technologies*, volume 3, pages 132–141, 2003.

[5] R. Bai, E.K. Burke, M. Gendreau, G. Kendall, and B. McCollum. Memory length in hyper-heuristics: An empirical study. In *Proceedings of the IEEE Symposium on Computational Intelligence in Scheduling*, pages 173–178. IEEE, 2007.

[6] R. Bai and G. Kendall. An investigation of automated planograms using a simulated annealing based hyper-heuristic. In *Metaheuristics: Progress as Real Problem Solvers*, pages 87–108. Springer, 2005.

[7] J.E. Baker. Adaptive selection methods for genetic algorithms. In *Proceedings of the International Conference on Genetic Algorithms and their Applications*, pages 101–111. Hillsdale, New Jersey, 1985.

[8] R.E. Bellman. A problem in the sequential design of experiments. *Sankhyā: The Indian Journal of Statistics*, 16:221–229, 1956.

[9] M. Biazzini, B. Bánhelyi, A. Montresor, and M. Jelasity. Distributed hyper-heuristics for real parameter optimization. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, pages 1339–1346. ACM, 2009.

[10] M. Birattari, T. Stützle, L. Paquete, and K. Varrentrapp. A racing algorithm for configuring metaheuristics. *Proceedings of the Genetic and Evolutionary Computation Conference*, 2:11–18, 2002.

[11] T.M. Blackwell and P.J. Bentley. Improvised music with swarms. *Proceedings of the IEEE Congress on Evolutionary Computation*, 2:1462–1467, 2002.

[12] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE Transactions on Evolutionary Computation*, 10(6):646–657, 2006.

[13] A. Brindle. *Genetic Algorithms for Function Optimization*. PhD thesis, Edmonton: University of Alberta, Department of Computer Science, 1981.

[14] J. Brownlee. *Clever algorithms: nature-inspired programming recipes*. 2011.

[15] E.K. Burke and Y. Bykov. A late acceptance strategy in hill-climbing for exam timetabling problems. In *Proceedings of the PATAT Conference*, 2008.

[16] E.K. Burke, M. Gendreau, M. Hyde, G. Kendall, B. McCollum, G. Ochoa, A.J. Parkes, and S. Petrovic. The cross-domain heuristic search challenge–an international research competition. In *Learning and Intelligent Optimization*, pages 631–634. Springer, 2011.

[17] E.K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and R. Qu. Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society*, 64:1695–1724, 2010.

[18] E.K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and J.R. Woodward. A classification of hyper-heuristic approaches. In *Handbook of Metaheuristics*, pages 449–468. Springer, 2010.

[19] E.K. Burke and G. Kendall. *Search methodologies, introductory tutorials in optimization and decision support techniques*. Springer, 2006.

[20] E.K. Burke, G. Kendall, and E. Soubeiga. A tabu-search hyperheuristic for timetabling and rostering. *Journal of Heuristics*, 9(6):451–470, 2003.

[21] E.K. Burke, J.D. Landa-Silva, and E. Soubeiga. Multi-objective hyper-heuristic approaches for space allocation and timetabling. In *Metaheuristics: Progress as Real Problem Solvers*, pages 129–158. Springer, 2005.

[22] E.K. Burke, J.D. Landa-Silva, and E. Soubeiga. Multi-objective hyper-heuristic approaches for space allocation and timetabling. In *Metaheuristics: Progress as Real Problem Solvers*, pages 129–158. Springer, 2005.

[23] E.K. Burke, B. McCollum, A. Meisels, S. Petrovic, and R. Qu. A graph-based hyper-heuristic for educational timetabling problems. *European Journal of Operational Research*, 176(1):177–192, 2007.

[24] A. Caponio, G.L. Cascella, F. Neri, N Salvatore, and M. Sumner. A fast adaptive memetic algorithm for online and offline control design of pmsm drives. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 37(1):28–41, 2007.

[25] A. Caponio, F. Neri, and V. Tirronen. Super-fit control adaptation in memetic differential evolution frameworks. *Soft Computing*, 13(8-9):811–831, 2009.

[26] Fabio Caraffini, Ferrante Neri, Giovanni Iacca, and Aran Mol. Parallel memetic structures. *Information Sciences*, 227:60–82, 2013.

[27] Fabio Caraffini, Ferrante Neri, and Lorenzo Picinali. An analysis on separability for memetic computing automatic design. *Information Sciences*, 265:1–22, 2014.

[28] K. Chakhlevitch and P. Cowling. Choosing the fittest subset of low level heuristics in a hyperheuristic framework. In *Evolutionary Computation in Combinatorial Optimization*, pages 23–33. Springer, 2005.

[29] P.C. Chen, G. Kendall, and G Van den Berghe. An ant based hyper-heuristic for the travelling tournament problem. In *Proceedings of the IEEE Symposium on Computational Intelligence in Scheduling*, pages 19–26. IEEE, 2007.

[30] X. Chen, Y. Ong, M. Lim, and K.C. Tan. A multi-facet survey on memetic computation. *IEEE Transactions on Evolutionary Computation*, 15(5):591–607, 2011.

[31] A. Chotard, A. Auger, and Hansen. N. Cumulative step-size adaptation on linear functions. *Parallel Problem Solving from Nature*, pages 72–81, 2012.

[32] C. Cobos, M. Mendoza, and E. Leon. A hyper-heuristic approach to design and tuning heuristic methods for web document clustering. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1350–1358. IEEE, 2011.

[33] D. Corne and P. Ross. Peckish initialisation strategies for evolutionary timetabling. In *Practice and Theory of Automated Timetabling*, pages 227–240. Springer, 1996.

[34] P. Cowling and K. Chakhlevitch. Hyperheuristics for managing a large collection of low level heuristics to schedule personnel. In *Proceedings of the Congress on Evolutionary Computation*, volume 2, pages 1214–1221. IEEE, 2003.

[35] P. Cowling, G. Kendall, and E. Soubeiga. A hyperheuristic approach to scheduling a sales summit. In *Practice and Theory of Automated Timetabling III*, pages 176–190. Springer, 2001.

[36] P. Cowling, G. Kendall, and E. Soubeiga. A parameter-free hyperheuristic for scheduling a sales summit. In *Proceedings of the 4th Metaheuristic International Conference*, volume 1101, pages 127–131. Citeseer, 2001.

[37] P. Cowling, G. Kendall, and E. Soubeiga. Hyperheuristics: A tool for rapid prototyping in scheduling and optimisation. In *Applications of Evolutionary Computing*, pages 1–10. Springer, 2002.

[38] T.G. Crainic and M. Toulouse. Parallel strategies for meta-heuristics. In F. Glover and G. Kochenberger, editors, *Handbook in Meta-heuristics*, pages 475–513. Kluwer Academic Publishers, 2003.

[39] M. Črepinšek, S. Liu, and M. Mernik. Exploration and exploitation in evolutionary algorithms: a survey. *ACM Computing Surveys*, 45(3):35, 2013.

[40] L. Davis. Adapting operator probabilities in genetic algorithms. *Proceedings of the Conference on Genetic and Evolutionary Computation*, pages 61–69, 1989.

[41] P. Demeester, B. Bilgin, P.D. Causmaecker, and G. Van den Berghe. A hyperheuristic approach to examination timetabling problems: benchmarks and a new problem from practice. *Journal of Scheduling*, 15(1):83–103, 2012.

[42] T.G. Dietterich. Ensemble methods in machine learning. In *Multiple Classifier Systems*, pages 1–15. Springer, 2000.

[43] J.H. Drake, E Özcan, and E.K. Burke. Modified choice function heuristic selection for the multidimensional knapsack problem. *Genetic and Evolutionary Computing*, pages 225–234, 2015.

[44] G. Dueck. New optimization heuristics: the great deluge algorithm and the record-to-record travel. *Journal of Computational Physics*, 104(1):86–92, 1993.

[45] R.C. Eberhart and Y. Shi. Particle swarm optimization: developments, applications, and resources. *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 81–86, 2001.

[46] A.P. Engelbrecht. *Computational Intelligence: an Introduction*. John Wiley and Sons, Ltd, 2003.

[47] A.P. Engelbrecht. *Fundamentals of Computational Swarm Intelligence*. Wiley, 2005.

[48] A.P. Engelbrecht. Heterogeneous particle swarm optimization. In *Swarm Intelligence*, pages 191–202. Springer, 2010.

[49] A.P. Engelbrecht. Scalability of a heterogeneous particle swarm optimizer. *Proceedings of the IEEE Symposium on Swarm Intelligence*, pages 1–8, 2011.

[50] A.P. Engelbrecht and A. Ismail. Training product unit neural networks. *Stability and Control: Theory and Applications*, 2(1-2):59–74, 1999.

[51] L.J. Eshelman. Real-coded genetic algorithms and interval-schemata. *Foundations of Genetic Algorithms*, 2:187–202, 1993.

[52] Feoktistov. *Differential Evolution in Search of Solutions*, volume 5 of *Optimization and its Applications*. Springer, 2006.

[53] A. Fialho, L. Da Costa, M. Schoenauer, and M. Sebag. Extreme value based adaptive operator selection. In *Parallel Problem Solving from Nature*, pages 175–184. Springer, 2008.

[54] A. Fialho, R. Ros, M. Schoenauer, and M. Sebag. Comparison-based adaptive strategy selection with bandits in differential evolution. In *Parallel Problem Solving from Nature*, pages 194–203. Springer, 2010.

[55] A. Fialho, M. Schoenauer, and M. Sebag. Fitness-auc bandit adaptive strategy selection vs. the probability matching one within differential evolution: an empirical comparison on the bbob-2010 noiseless testbed. *Proceedings of the Conference on Genetic and Evolutionary Computation (Workshop on Black-Box Optimization Benchmarking)*, pages 1535–1542, 2010.

[56] A. Fialho, M. Schoenauer, and M. Sebag. Toward comparison-based adaptive operator selection. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, pages 767–774. ACM, 2010.

[57] A.S. Fraser. Simulation of genetic systems by automatic digital computers i: Introduction. *Australian Journal of Biological Sciences*, 13(2):150–162, 1960.

[58] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 1986.

[59] F. Glover. Tabu search methods in artificial intelligence and operations research. *ORSA Artificial Intelligence Newsletter 1*, 1987.

[60] F. Glover and G.A. Kochenberger. *Handbook of Metaheuristics*. Springer, 2003.

[61] D.E. Goldberg. Probability matching, the magnitude of reinforcement, and classifier system bidding. *Machine Learning*, 5(4):407–425, 1990.

[62] C.P. Gomes and B. Selman. Algorithm portfolios. *Artificial Intelligence*, 126(1):43–62, 2001.

[63] W. Gong, A. Fialho, and Z. Cai. Adaptive strategy selection in differential evolution. *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, pages 409–416, 2010.

[64] J. Grobler. Particle swarm optimization and differential evolution for multi-objective multiple machine scheduling. Master's thesis, University of Pretoria, 2008.

[65] J. Grobler, A.P. Engelbrecht, G. Kendall, and V.S.S. Yadavalli. Investigating the impact of alternative evolutionary selection strategies on multi-method global optimization. *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 2337–2344, 2011.

[66] J. Grobler, A.P. Engelbrecht, G. Kendall, and V.S.S. Yadavalli. The entity-to-algorithm allocation problem: extending the analysis. *Proceedings of the IEEE Symposium on Computational Intelligence in Ensemble Learning*, 2014.

[67] D. Hadka and P. Reed. Borg: An auto-adaptive many-objective evolutionary computing framework. *Evolutionary Computation*, 21(2):231–259, 2013.

[68] L.K. Hansen and P. Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990.

[69] E. Hart and K. Sim. On the life-long learning capabilities of nelli: a hyper-heuristic optimization system. *Parallel Problem Solving from Nature*, 2014b.

[70] K. Hart, E.and Sim and B. Paechter. A lifelong learning hyper-heuristic method for bin packing. *Evolutionary Computation*, 2014.

[71] W.E. Hart, N. Krasnogor, and J.E. Smith. *Recent advances in memetic algorithms*, volume 166. Springer, 2005.

[72] F. Heppner and U. Grenander. A stochastic nonlinear model for coordinated bird flocks. In S. Krasner, editor, *The Ubiquity of Chaos*. AAAS Publications, 1990.

[73] J.H. Holland. Outline for a logical theory of adaptive systems. *Journal of the ACM*, 9(3):297–314, 1962.

[74] J.H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. University of Michigan Press, 1975.

[75] R. Hooke and T. A. Jeeves. "direct search" solution of numerical and statistical problems. *Journal of the ACM*, 8(2):212–229, 1961.

[76] A.S. Jain and S. Meeran. Deterministic job-shop scheduling: Past, present and future. *European Journal of Operational Research*, 113(2):390–434, 1999.

[77] B.A. Julstrom. What have you done for me lately? adapting operator probabilities in a steady-state genetic algorithm. *Proceedings of the* 6^{th} *International Conference on Genetic Algorithms*, pages 81–87, 1995.

[78] G. Kendall and N. M. Hussin. A tabu search hyper-heuristic approach to the examination timetabling problem at the mara university of technology. *Proceedings of the* 5^{th} *Conference on Practice and Theory of Automated Timetabling*, 2005.

[79] G. Kendall and M. Mohamad. Channel assignment in cellular communication using a great deluge hyper-heuristic. In *Proceedings of the 12th IEEE International Conference on Networks*, volume 2, pages 769–773. IEEE, 2004.

[80] G. Kendall and M. Mohamad. Channel assignment optimisation using a hyper-heuristic. In *Proceedings of the IEEE Conference on Cybernetics and Intelligent Systems*, volume 2, pages 791–796. IEEE, 2004.

[81] J. Kennedy. Bare bones particle swarms. In *Proceedings of the IEEE Swarm Intelligence Symposium*, pages 80–87. IEEE, 2003.

[82] J. Kennedy and R. Eberhart. Particle swarm optimization. *Proceedings of the IEEE International Confererence on Neural Networks*, 4:1942–1948, 1995.

[83] J. Kennedy, R.C. Eberhart, and Y. Shi. *Swarm Intelligence*. Morgan Kaufmann Publishers, 2001.

[84] J. Kennedy and R. Mendes. Population structure and particle swarm performance. *Proceedings of the IEEE Congress on Evolutionary Computation*, 2:1671–1676, 2002.

[85] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simmulated annealing. *Science*, 220(4598):671–680, 1983.

[86] P. Korošec, J. Šilc, and B. Filipič. The differential ant-stigmergy algorithm. *Information Sciences*, 192:82–97, 2012.

[87] N. Krasnogor and J. Smith. A memetic algorithm with self-adaptive local search: Tsp as a case study. *Proceedings of the Conference on Genetic and Evolutionary Computation*, pages 987–994, 2000.

[88] N. Krasnogor and J. Smith. A tutorial for competent memetic algorithms: model, taxonomy, and design issues. *IEEE Transactions on Evolutionary Computation*, 9(5):474–488, 2005.

[89] M.W.S. Land. *Evolutionary algorithms with local search for combinatorial optimization*. PhD thesis, University of California, San Diego, 1998.

[90] M.N. Le, Y. Ong, Y. Jin, and B. Sendhoff. Lamarckian memetic algorithms: local optimum and connectivity structure analysis. *Memetic Computing*, 1(3):175–190, 2009.

[91] K. Li, A. Fialho, S. Kwong, and Q. Zhang. Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 18(1):114–130, 2014.

[92] F.G. Lobo and D.E. Goldberg. Decision making in a hybrid genetic algorithm. In *Proceedings of the IEEE International Conference onEvolutionary Computation*, pages 121–125. IEEE, 1997.

[93] E. López-Camacho, H. Terashima-Marin, P. Ross, and G. Ochoa. A unified hyper-heuristic framework for solving bin packing problems. *Expert Systems with Applications*, 2014.

[94] X. Lu, K. Tang, S. Bernhard, and X. Yao. A new self-adaptation scheme for differential evolution. *Neurocomputing*, 146:2–16, 2014. Bridging Machine learning and Evolutionary Computation (BMLEC) Computational Collective Intelligence.

[95] M. Maashi, E Özcan, and G. Kendall. A multi-objective hyper-heuristic based on choice function. *Expert Systems with Applications*, pages 4475–4493, 2014.

[96] R. Mallipeddi, P.N. Suganthan, G.K. Pan, and M.F. Tasgetiren. Differential evolution algorithm with ensemble of parameters and mutation strategies. *Applied Soft Computing*, 2010.

[97] J. Maturana, F. Lardeux, and F. Saubion. Autonomous operator management for evolutionary algorithms. *Journal of Heuristics*, 16(6):881–909, 2010.

[98] M. Maza and B. Tidor. An analysis of selection procedures with particular attention paid to proportional and boltzmann selection. In *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 124–131. Morgan Kaufmann Publishers Inc., 1993.

[99] K. McClymont and E.C. Keedwell. Markov chain hyper-heuristic (mchh): an online selective hyper-heuristic for multi-objective continuous problems. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, pages 2003–2010. ACM, 2011.

[100] D. Meignan, A. Koukam, and J.C. Créput. Coalition-based metaheuristic: a self-adaptive metaheuristic using reinforcement learning and mimetism. *Journal of Heuristics*, 16(6):859–879, 2010.

[101] M. Mısır, W. Vancroonenburg, and G Van den Berghe. A selection hyper-heuristic for scheduling deliveries of ready-mixed concrete. In *Proceedings of the 9th Metaheuristic International Conference*, 2011.

[102] M. Mısır, K. Verbeeck, P. De Causmaecker, and G. Van den Berghe. Design and analysis of an evolutionary selection hyper-heuristic framework. Technical report, KAHO Sint-Lieven, 2013.

[103] M. Montazeri, M.S. Baghshah, and A. Enhesari. Hyper-heuristic algorithm for finding efficient features in diagnose of lung cancer disease. *Journal of Basic and Applied Scientific Research*, 3(10):134–140, 2013.

[104] P. Moscato. On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms. Technical report, Caltech Concurrent Computation Program, 1989.

[105] A. Nareyek. Choosing search heuristics by non-stationary reinforcement learning. In *Metaheuristics: Computer Decision-making*, pages 523–544. Springer, 2004.

[106] J.A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7(4):308–313, 1965.

[107] F.V. Nepomuceno and A.P. Engelbrecht. A self-adaptive heterogeneous pso inspired by ants. In *Swarm Intelligence*, pages 188–195. Springer, 2012.

[108] F.V. Nepomuceno and A.P. Engelbrecht. A self-adaptive heterogeneous pso for real-parameter optimization. *Proceedings of the IEEE Conference on Evolutionary Computation*, pages 361–368, 2013.

[109] F. Neri, V. Tirronen, T. Karkkainen, and T. Rossi. Fitness diversity based adaptation in multimeme algorithms: A comparative study. *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 2374–2381, 2007.

[110] F. Neri, J. Toivanen, G.L. Cascella, and Y. Ong. An adaptive multimeme algorithm for designing hiv multidrug therapies. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 4(2):264–278, 2007.

[111] Q.H. Nguyen, Y. Ong, and M.H. Lim. A probabilistic memetic framework. *IEEE Transactions on Evolutionary Computation*, 13(3):604–623, 2009.

[112] Q.H. Nguyen, Y.S. Ong, and N. Krasnogor. A study on the design issues of memetic algorithm. *Proceedings of the IEEE Conference on Evolutionary Computation*, 2007.

[113] Gabriela Ochoa, Rong Qu, and Edmund K Burke. Analyzing the landscape of a graph based hyper-heuristic for timetabling problems. *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 341–348, 2009.

[114] Gabriela Ochoa, José Antonio Vázquez-Rodríguez, Sanja Petrovic, and Edmund Burke. Dispatching rules for production scheduling: a hyper-heuristic landscape analysis. *Proceedings of the 2009 Congress on Evolutionary Computation*, pages 1873–1880, 2009.

[115] D. Oeulhadj and S. Petrovic. A cooperative distributed hyper-heuristic framework for scheduling. *Proceedings of the IEEE Conference on Systems, Man and Cybernetics*, 2008.

[116] O. Olorunda and A.P. Engelbrecht. An analysis of heterogeneous cooperative algorithms. *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1562–1569, 2009.

[117] Y. Ong, M. Lim, N. Zhu, and K. Wong. Classification of adaptive memetic algorithms: a comparative study. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 36(1):141–152, 2006.

[118] José C Ortiz-Bayliss, Hugo Terashima-Marın, Ender Ozcan, Andrew J Parkes, and Santiago E Conant-Pablos. Exploring heuristic interactions in constraint satisfaction problems: A closer look at the hyper-heuristic space. *Proceedings of the 2013 Congress on Evolutionary Computation*, page In Press, 2013.

[119] D. Ouelhadj and S. Petrovic. A cooperative hyper-heuristic search framework. *Journal of Heuristics*, 16(6):835–857, 2010.

[120] E. Özcan, B. Bilgin, and E.E. Korkmaz. Hill climbers and mutational heuristics in hyperheuristics. In *Parallel Problem Solving from Nature*, pages 202–211. Springer, 2006.

[121] E. Özcan, B. Bilgin, and E.E. Korkmaz. A comprehensive analysis of hyper-heuristics. *Intelligent Data Analysis*, 12(1):3–23, 2008.

[122] F. Peng, K. Tang, G. Chen, and X. Yao. Population-based algorithm portfolios for numerical optimization. *IEEE Transactions on Evolutionary Computation*, 14(5):782–800, 2010.

[123] K.V. Price, R.M. Storn, and J.A. Lampinen. *Differential evolution, a practical approach to global optimization*. Natural Computing Series. Springer, 2005.

[124] A.K. Qin and P.N. Suganthan. Self-adaptive differential evolution algorithm for numerical optimization. *Proceedings of the IEEE Congress on Evolutionary Computation*, 2:1785–1791, 2005.

[125] R. Qu and E.K. Burke. Hybridizations within a graph-based hyper-heuristic framework for university timetabling problems. *Journal of the Operational Research Society*, 60(9):1273–1285, 2008.

[126] R.L. Rardin. *Optimization in Operations Research*. Prentice Hall, 1998.

[127] A. Ratnaweera, H. Watson, and S. K. Halgamuge. Particle swarm optimizer with time varying acceleration coefficients. *Proceedings of the International Conference on Soft Computing and Intelligent Systems*, 2002.

[128] P. Rattadilok, A. Gaw, and R.S.K. Kwan. Distributed choice function hyper-heuristics for timetabling and scheduling. In *Practice and Theory of Automated Timetabling*, pages 51–67. Springer, 2005.

[129] I. Rechenberg. Cybernetic solution path of an experimental problem. Technical report, Ministry of Aviation, Royal Aircraft Establishment, 1965.

[130] I. Rechenberg. *Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologishen Evolution*. Frammann-Holzboog Verlag, Stuttgart, 1973.

[131] Z. Ren, H. Jiang, J. Xuan, and Z. Luo. Ant based hyper heuristics with space reduction: a case study of the p-median problem. In *Parallel Problem Solving from Nature*, pages 546–555. Springer, 2010.

[132] Z. Ren, H. Jiang, J. Xuan, and Z. Luo. Hyper-heuristics with low level parameter adaptation. *Evolutionary Computation*, 20(2):189–227, 2012.

[133] C.W. Reynolds. Flocks, herds, and schools: a distributed behavioral model. *Computer Graphics*, 21(4):25–34, 1987.

[134] J.R. Rice. The algorithm selection problem. *Advances in Computing*, 1976.

[135] P. Ross, S. Schulenburg, J.G. Marín-Blázquez, and E. Hart. Hyper-heuristics: learning to combine simple heuristics in bin-packing problems. In *GECCO*, pages 942–948, 2002.

[136] B. Ruzek and M. Kvasnicka. Determination of the earthquake hypocenter: a challenge for the differential evolution algorithm. Section 7.5 in Differential Evolution, A Practical Approach to Global Optimization (Price, K.V. and Storn, R.M. and Lampinen, J.A.) Springer., 2005.

[137] N. Sabar and G. Kendall. Population based monte carlo tree search hyper-heuristic for combinatorial optimization problems. *Information Sciences*, 2014.

[138] N.R. Sabar, M. Ayob, G. Kendall, and R. Qu. Grammatical evolution hyper-heuristic for combinatorial optimization problems. *IEEE Transactions on Evolutionary Computation*, 17(6):840–861, 2013.

[139] N.R. Sabar, M. Ayob, G. Kendall, and R. Qu. The automatic design of hyper-heuristic framework with gene expression prgoramming for combinatorial optimization problems. *IEEE Transactions on Evolutionary Computation*, 2014.

[140] N.R. Sabar, M. Ayob, G. Kendall, and R. Qu. A dynamic multi-armed bandit-gene expression programming hyper-heuristic for combinatorial optimization problems. *IEEE Transactions on Cybernetics*, 2014.

[141] S. Salcedo-Sanz, J.M. Matías-Román, S. Jiménez-Fernández, A.. Portilla-Figueras, and L Cuadra. An evolutionary-based hyper-heuristic approach for the jawbreaker puzzle. *Applied Intelligence*, pages 1–11, 2013.

[142] A. Salman, I. Ahmad, and S. Al-Madani. Particle swarm optimization for task assignment problem. *Microprocessors and Microsystems*, 26(8):363–371, 2002.

[143] E. Segredo, C. Segura, and C. León. Memetic algorithms and hyperheuristics applied to a multiobjectivised two-dimensional packing problem. *Journal of Global Optimization*, pages 1–26, 2013.

[144] K. Sim and E. Hart. Generating single and multiple cooperative hyper-heuristics for the one-dimensional bin-packing problem using a single node genetic programming island model. *Proceedings of the Conference on Genetic and Evolutionary Computation*, 2013.

[145] K. Sim and E. Hart. An improved immune-inspired hyper-heuristic for combinatorial optimisation problems. *Proceedings of the Conference on Genetic and Evolutionary Computation*, 2014.

[146] K. Sim, E. Hart, and B. Paechter. Learning to solve bin packing problems with an immune inspired hyper-heuristic. *Proceedings of the 12th European Conference on Artificial Life*, pages 856–863, 2013.

[147] J.E. Smith. Coevolving memetic algorithms: a review and progress report. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 37(1):6–17, 2007.

[148] J.E. Smith. Estimating meme fitness in adaptive memetic algorithms for combinatorial problems. *Evolutionary Computation*, 20(2):165–188, 2012.

[149] K. Smith-Miles, D. Baatar, B. Wreford, and R. Lewis. Toward objective measures of algorithm performance across instance space. *Computers & Operations Research*, 45:12–24, 2014.

[150] P. Spanevello and M. Montes de Oca. Experiments on adaptive heterogeneous pso algorithms. *Proceedings of the Doctoral Symposium on Engineering Stochastic Local Search Algorithms*, pages 36–40, 2009.

[151] M. Srinivas and L.M. Patnaik. Genetic algorithms: A survey. *Computer*, 27(6):17–26, 1994.

[152] R. Storn. On the usage of differential evolution for function optimization. *Proceedings of the Biennial Conference of the North American Fuzzy Information Processing Society*, pages 519–523, 1996.

[153] R. Storn and K. Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, 1997.

[154] D. Sudholt. Local search in evolutionary algorithms: the impact of the local search frequency. In *Algorithms and Computation*, pages 359–368. Springer, 2006.

[155] P.N. Suganthan, N. Hansen, J.J. Liang, Y.P. Deb, K.and Chen, A. Auger, and S. Tiwari. Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization. Technical report, Nanyang Technological University and Kanpur Genetic Algorithms Laboratory, 2005.

[156] J. Tang, M.H. Lim, and Y. Ong. Diversity-adaptive parallel memetic algorithm for solving large scale combinatorial optimization problems. *Soft Computing*, 11(9):873–888, 2007.

[157] K. Tang, F. Peng, G. Chen, and Yao. Population-based algorithm portfolios with automated constituent algorithm selection. *Information Sciences*, (279):94–104, 2014.

[158] D. Thierens. An adaptive pursuit strategy for allocating operator probabilities. In *Proceedings of the Conference on Genetic and Evolutionary Computation*, pages 1539–1546. ACM, 2005.

[159] V. Tirronen, F. Neri, T. Kärkkäinen, K. Majava, and T. Rossi. An enhanced memetic differential evolution in filter design for defect detection in paper production. *Evolutionary Computation*, 16(4):529–555, 2008.

[160] C.A. Tovey. Tutorial on computational complexity. *Interfaces*, pages 30–61, 2002.

[161] I.C. Trelea. The particle swarm optimization algorithm: convergence analysis and parameter selection. *Information Processing Letters*, 85(6):317–325, 2003.

[162] A. Tuson and P. Ross. Adapting operator settings in genetic algorithms. *Evolutionary Computation*, 6(2):161–184, 1998.

[163] J. Tvrdík. Modifications of differential evolution with composite trial vector generation strategies. In *Soft Computing Models in Industrial and Environmental Applications*, pages 113–122. Springer, 2013.

[164] F. Van Den Bergh. *An analysis of particle swarm optimizers*. PhD thesis, University of Pretoria, 2006.

[165] F. Van den Bergh and A.P. Engelbrecht. Cooperative learning in neural networks using particle swarm optimizers. *South African Computer Journal*, 26:84–90, 2000.

[166] F. Van den Bergh and A.P Engelbrecht. A new locally convergent particle swarm optimiser. *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, 3:6–12, 2002.

[167] F. Van den Bergh and A.P. Engelbrecht. A study of particle swarm optimization particle trajectories. *Information Sciences*, 176(8):937–971, 2006.

[168] J.A. Vazquez-Rodriguez, S. Petrovic, and A. Salhi. A combined meta-heuristic with hyper-heuristic approach to the scheduling of the hybrid flow shop with sequence dependent setup ttime and uniform machines. *Proceedings of the* 3^{rd} *Multi-*

disciplinary International Scheduling Conference: Theory and Applications, pages 506–513, 2007.

[169] N. Veerapen, Y. Hamadi, and F. Saubion. Using local search with adaptive operator selection to solve the progressive party problem. Technical report, Microsoft Research, 2013.

[170] G. Venter and J. Sobieszczanski-Sobieski. Multidisciplinary optimization of a transport aircraft wing using particle swarm optimization. *Proceedings of the* 9^{th} *AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, 2002.

[171] J.S. Vesterstrom, J. Riget, and T. Krink. Division of labor in particle swarm optimisation. In *Proceedings of the IEEE Congress on Evolutionary Computation*, volume 2, pages 1570–1575. IEEE, 2002.

[172] J.A. Vrugt, B.A. Robinson, and J.M. Hyman. Self-adaptive multimethod search for global optimization in real-parameter spaces. *IEEE Transactions on Evolutionary Computation*, 13(2):243–259, 2009.

[173] Y. Wang, Z. Cai, and Q. Zhang. Differential evolution with composite trial vector generation strategies and control parameters. *IEEE Transactions on Evolutionary Computation*, 15(1):55–66, 2011.

[174] J.M. Whitacre, T.Q. Pham, and R.A. Sarker. Use of statistical outlier detection method in adaptive evolutionary algorithms. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, pages 1345–1352. ACM, 2006.

[175] D. Whitley, v.S. Gordon, and K. Mathias. Lamarckian evolution, the baldwin effect and function optimization. In *Parallel Problem Solving from Nature*, pages 5–15. Springer, 1994.

[176] D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

[177] Y. Xue, S. Zhong, Y. Zhuang, and B. Xu. An ensemble algorithm with self-adaptive learning techniques for high-dimensional numerical optimization. *Applied Mathematics and Computation*, 231:329–346, 2014.

[178] Z. Yang, J. He, and X. Yao. Making a difference to differential evolution. In Z. Michalewicz and P. Siarry, editors, *Advances in Metaheuristics for Hard Optimization*, pages 397–414. Springer, 2008.

[179] Z. Yang, K. Tang, and X. Yao. Self-adaptive differential evolution with neighborhood search. *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1110–1116, 2008.

[180] Z. Yang, K. Tang, and X. Yao. Scalability of generalized adaptive differential evolution for large-scale continuous optimization. *Soft Computing*, 15(11):2141–2155, 2011.

[181] X. Yao, Y. Liu, and Lin G. Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation*, 3(2):82–102, 1999.

[182] E.L. Yu and P.N. Suganthan. Ensemble of niching algorithms. *Information Sciences*, 180(15):2815–2833, 2010.

[183] S.Y. Yuen, C.K. Chow, and X. Zhang. Which algorithm should i choose at any point of the search: an evolutionary portfolio approach. In *Proceedings of the Annual Conference on Genetic and Evolutionary Computation*, pages 567–574. ACM, 2013.

[184] C. Zhang, P. Li, Z. Guan, and Y. Rao. A tabu search algorithm with a new neighborhood structure for the job shop scheduling problem. *Computers and Operations Research*, 34(11):3229–3242, 2006.

[185] J. Zhang and A.C. Sanderson. Jade: adaptive differential evolution with optional external archive. *IEEE Transactions on Evolutionary Computation*, 13(5):945–958, 2009.

[186] J. Zhong, M. Shen, J. Zhang, H. Chung, Y. Shi, and Y. Li. A differential evolution algorithm with dual populations for solving periodic railway timetable scheduling problem. *IEEE Transactions on Evolutionary Computation*, 2012. In press.

Appendix A

Benchmark Problem Set

All algorithms presented in this thesis were evaluated on the first 17 problems of the 2005 IEEE Congress of Evolutionary Computation benchmark problem set [155]. This problem benchmark set is summarized in this appendix.

A.1 F_1: Shifted Sphere Function

$$F_1(\boldsymbol{x}) = \sum_{i=1}^{n_x} z_i^2 + f_{bias_1} \tag{A.1}$$

where

$$\boldsymbol{z} = \boldsymbol{x} - \boldsymbol{o}, \tag{A.2}$$

$$x \in [-100, 100]^{n_x} \text{ and} \tag{A.3}$$

$$f_{bias_1} = -450 \tag{A.4}$$

A.2 F_2: Shifted Schwefel's Problem 1.2

$$F_2(\boldsymbol{x}) = \sum_{i=1}^{n_x} (\sum_{j=1}^{i} z_j)^2 + f_{bias_2} \tag{A.5}$$

where

$$\boldsymbol{z} = \boldsymbol{x} - \boldsymbol{o}, \tag{A.6}$$

$$x \in [-100, 100]^{n_x} \text{ and} \tag{A.7}$$

$$f_{bias_2} = -450 \tag{A.8}$$

A.3 F_3: Shifted Rotated High Conditioned Elliptic Function

$$F_3(\boldsymbol{x}) = \sum_{i=1}^{n_x} (10^6)^{\frac{i-1}{n_x - 1}} z_i^2 + f_{bias_3} \tag{A.9}$$

where

$$\boldsymbol{z} = (\boldsymbol{x} - \boldsymbol{o})\boldsymbol{M}, \tag{A.10}$$

$$x \in [-100, 100]^{n_x}, \tag{A.11}$$

$$f_{bias_3} = -450 \tag{A.12}$$

and $\boldsymbol{M}$ is an orthogonal matrix.

A.4 F_4: Shifted Schwefel's Problem 1.2 With Noise in Fitness

$$F_4(\boldsymbol{x}) = \sum_{i=1}^{n_x} \left(\sum_{j=1}^{i} z_j \right)^2 \times (1 + 0.4|N(0,1)|) + f_{bias_4} \tag{A.13}$$

where

$$\boldsymbol{z} = \boldsymbol{x} - \boldsymbol{o}, \tag{A.14}$$

$$x \in [-100, 100]^{n_x} \text{ and} \tag{A.15}$$

$$f_{bias_4} = -450 \tag{A.16}$$

A.5 F_5: Schwefel's Problem 2.6 with Global Optimum on Bounds

$$f(\boldsymbol{x}) = \max\{|x_1 + 2x_2 - 7|, |2x_1 + x_2 - 5|\}, \text{ where} \tag{A.17}$$

$$\boldsymbol{x}^* = [1, 3] \tag{A.18}$$

Extension to n_x dimensions gives:

$$F_5(\boldsymbol{x}) = \max\{|\boldsymbol{A}_i\boldsymbol{x} - \boldsymbol{B}_i|\} + f_{bias_5} \tag{A.19}$$

where

$$x \in [-100, 100]^{n_x}, \tag{A.20}$$

$$f_{bias_4} = -310, \tag{A.21}$$

$\boldsymbol{A}$ is a $n_x \times n_x$ matrix consisting of integer random numbers in the range $[-500, 500]$, $\det(\boldsymbol{A}) = \boldsymbol{0}$, $\boldsymbol{B}_i = \boldsymbol{A}_i o$, $\boldsymbol{o}$ is a $n_x \times 1$ vector, and o_i are random numbers in the range $[-100, 100]$.

A.6 F_6: Shifted Rosenbrock's Function

$$F_6(\boldsymbol{x}) = \sum_{i=1}^{n_x - 1} (100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2) + f_{bias_6} \tag{A.22}$$

where

$$\boldsymbol{z} = \boldsymbol{x} - \boldsymbol{o} + 1, \tag{A.23}$$

$$x \in [-100, 100]^{n_x} \text{ and} \tag{A.24}$$

$$f_{bias_6} = 390 \tag{A.25}$$

A.7 F_7: Shifted Rotated Griewank's Function Without Bounds

$$F_7(\boldsymbol{x}) = \sum_{i=1}^{n_x} \frac{z_i^2}{4000} - \prod_{i=1}^{n_x} \cos\left(\frac{z_i}{\sqrt{i}}\right) + 1 + f_{bias_7} \tag{A.26}$$

where

$$\boldsymbol{z} = (\boldsymbol{x} - \boldsymbol{o})\boldsymbol{M}, \tag{A.27}$$

$$\boldsymbol{M} = \boldsymbol{M}'(1 + 0.3|N(0,1)|), \tag{A.28}$$

$$\boldsymbol{x}(0) \in [0, 600]^{n_x}, \tag{A.29}$$

$$f_{bias_7} = -180 \tag{A.30}$$

and $\boldsymbol{M}'$ is a linear transformation matrix of condition number 3.

A.8 F_8: Shifted Rotated Ackley's Function with Global Optimum on Bounds

$$F_8(\boldsymbol{x}) = -20exp(-0.2\sqrt{\frac{1}{n_x}\sum_{i=1}^{n_x} z_i^2}) - exp(\frac{1}{n_x}\sum_{i=1}^{n_x}\cos(2\pi z_i)) + 20 + e + f_{bias_8} \tag{A.31}$$

where

$$\boldsymbol{z} = (\boldsymbol{x} - \boldsymbol{o})\boldsymbol{M}, \tag{A.32}$$

$$x \in [-32, 32]^{n_x}, \tag{A.33}$$

$$f_{bias_8} = -140 \tag{A.34}$$

and $\boldsymbol{M}$ is a linear transformation matrix of condition number 100.

A.9 F_9: Shifted Rastrigin's Function

$$F_9(\boldsymbol{x}) = \sum_{i=1}^{n_x}(z_i^2 - 10\cos(2\pi z_i) + 10) + f_{bias_9} \tag{A.35}$$

where

$$\boldsymbol{z} = \boldsymbol{x} - \boldsymbol{o}, \tag{A.36}$$

$$x \in [-5, 5]^{n_x} \text{ and} \tag{A.37}$$

$$f_{bias_9} = -330 \tag{A.38}$$

A.10 F_{10}: Shifted Rotated Rastrigin's Function

$$F_{10}(\boldsymbol{x}) = \sum_{i=1}^{n_x}(z_i^2 - 10\cos(2\pi z_i) + 10) + f_{bias_{10}} \tag{A.39}$$

where

$$\boldsymbol{z} = (\boldsymbol{x} - \boldsymbol{o})\boldsymbol{M}, \tag{A.40}$$

$$x \in [-5, 5]^{n_x}, \tag{A.41}$$

$$f_{bias_{10}} = -330 \tag{A.42}$$

and $\boldsymbol{M}$ is a linear transformation matrix of condition number 2.

A.11 F_{11}: Shifted Rotated Weierstrass Function

$$F_{11}(\boldsymbol{x}) = \sum_{i=1}^{n_x}(\sum_{k=0}^{k_{max}}[a^k \cos(2\pi b^k(z_i + 0.5))]) - n_x \sum_{k=0}^{k_{max}}[a^k \cos(2\pi b^k 0.5)] + f_{bias_{11}} \tag{A.43}$$

where

$$\boldsymbol{z} = (\boldsymbol{x} - \boldsymbol{o}) \times \boldsymbol{M}, \tag{A.44}$$

$$x \in [-0.5, 0.5]^{n_x}, \tag{A.45}$$

$$a = 0.5, \tag{A.46}$$

$$b = 3, \tag{A.47}$$

$$k_{max} = 20, \tag{A.48}$$

$$f_{bias_{11}} = 90 \tag{A.49}$$

and $\boldsymbol{M}$ is a linear transformation matrix with condition number of 5.

A.12 F_{12}: Schwefel's Problem 2.13

$$F_{12}(\boldsymbol{x}) = \sum_{i=1}^{n_x}(\boldsymbol{A}_i - \boldsymbol{B}_i(\boldsymbol{x}))^2 + f_{bias_{12}} \tag{A.50}$$

where

$$\boldsymbol{A}_i = \sum_{j=1}^{n_x} (a_{ij} \sin \alpha_j + b_{ij} \cos \alpha_j), \tag{A.51}$$

$$\boldsymbol{B}_i(x) = \sum_{j=1}^{n_x} (a_{ij} \sin x_j + b_{ij} \cos x_j) \qquad \forall i \in \{1, \ldots, n_x\}, \tag{A.52}$$

$$x \in [-\pi, \pi]^{n_x}, \tag{A.53}$$

$$f_{bias_{12}} = -460, \tag{A.54}$$

$\boldsymbol{A}$, $\boldsymbol{B}$ are two $n_x \times n_x$ matrices, a_{ij} and b_{ij} are integer random numbers in the range $[-100, 100]$, and $\boldsymbol{\alpha}$ are random numbers in the range $[-\pi, \pi]$.

A.13 F_{13}: Shifted Expanded Griewank's Plus Rosenbrock's Function ($F8F2$)

$$\begin{aligned} F_{13}(\boldsymbol{x}) =& F8(F2(z_1, z_2)) + F8(F2(z_2, z_3)) + \ldots + \\ & F8(F2(z_{n_x-1}, z_{n_x})) + F8(F2(z_{n_x}, z_1)) + f_{bias_{13}} \end{aligned} \tag{A.55}$$

where

$$\boldsymbol{z} = \boldsymbol{x} - \boldsymbol{o} + 1, \tag{A.56}$$

$$F8(\boldsymbol{x}) = \sum_{i=1}^{n_x} \frac{x_i^2}{4000} - \prod_{i=1}^{n_x} \cos(\frac{x_i}{\sqrt{i}}) + 1, \tag{A.57}$$

$$F2(\boldsymbol{x}) = \sum_{i=1}^{n_x - 1} (100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2), \tag{A.58}$$

$$x \in [-5, 5]^{n_x} \text{ and} \tag{A.59}$$

$$f_{bias_{13}} = -130. \tag{A.60}$$

A.14 F_{14}: Shifted Rotated Expanded Schaffer's F_6 Function

$$F_{14}(\boldsymbol{x}) = F(z_1, z_2) + F(z_2, z_3) + \ldots + F(z_{n_x-1}, z_{n_x}) + F(z_{n_x}, z_1) + f_{bias_{14}} \tag{A.61}$$

where

$$F(x, y) = 0.5 + \frac{(\sin^2(\sqrt{x^2 + y^2}) - 0.5)}{(1 + 0.001(x^2 + y^2))^2}, \tag{A.62}$$

$$\boldsymbol{z} = (\boldsymbol{x} - \boldsymbol{o}) \times \boldsymbol{M}, \tag{A.63}$$

$$x \in [-100, 100]^{n_x}, \tag{A.64}$$

$$f_{bias_{14}} = -300. \tag{A.65}$$

and $\boldsymbol{M}$ is a linear transformation matrix with condition number of 3.

A.15 F_{15}: Hybrid Composition Function

$$F_{15}(\boldsymbol{x}) = \sum_{i=1}^{n} \{w_i \times [f_i'((\boldsymbol{x} - \boldsymbol{o}_i)\lambda_i \times \boldsymbol{M}_i) + bias_i]\} + f_{bias_{15}} \tag{A.66}$$

where

$$w1_i = exp(-\frac{\sum_{k=1}^{n_x}(x_k - o_{ik})^2}{2n_x\sigma_i^2}), \tag{A.67}$$

(A.68)

$$w2_i = \begin{cases} w1_i & \text{if } w1_i = \max(\boldsymbol{w1}), \\ w1_i(1 - \max(\boldsymbol{w1})^{10}) & \text{if } w1_i \ = \max(\boldsymbol{w1}), \end{cases}$$

$$w_i = \frac{w2_i}{\sum_{i=1}^{n} w2_i}, \tag{A.69}$$

$$x \in [-5, 5]^{n_x}, \tag{A.70}$$

$$f_{bias_{15}} = 120. \tag{A.71}$$

$f_{1-2}(\boldsymbol{x})$: Rastrigin's function

$$f_i(\boldsymbol{x}) = \sum_{i=1}^{n_x}(x_i^2 - 10\cos(2\pi x_i) + 10) \tag{A.72}$$

$f_{3-4}(\boldsymbol{x})$: Weierstrass function

$$f_i(\boldsymbol{x}) = \sum_{i=1}^{n_x}(\sum_{k=0}^{k_{max}}[a^k \cos(2\pi b^k(x_i + 0.5))]) - n_x \sum_{k=0}^{k_{max}}[a^k \cos(2\pi b^k 0.5)], \tag{A.73}$$

$$a = 0.5, \tag{A.74}$$

$$b = 3, \tag{A.75}$$

$$k_{max} = 20 \tag{A.76}$$

$f_{5-6}(\boldsymbol{x})$: Griewank's function

$$f_i(\boldsymbol{x}) = \sum_{i=1}^{n_x} \frac{x_i^2}{4000} - \prod_{i=1}^{n_x} \cos(\frac{x_i}{\sqrt{i}}) + 1 \tag{A.77}$$

$f_{7-8}(\boldsymbol{x})$: Ackley's function

$$f_i(\boldsymbol{x}) = -20exp(-0.2\sqrt{\frac{1}{n_x}\sum_{i=1}^{n_x} x_i^2}) - exp(\frac{1}{n_x}\sum_{i=1}^{n_x}\cos(2\pi x_i)) + 20 + e \tag{A.78}$$

$f_{9-10}(\boldsymbol{x})$: Sphere function

$$f_i(\boldsymbol{x}) = \sum_{i=1}^{n_x} x_i^2, \tag{A.79}$$

$$\sigma_i = 1 \qquad \forall i \in \{1, \ldots n_x\}, \tag{A.80}$$

$$\boldsymbol{\lambda} = [1, 1, 10, 10, 5/60, 5/60, 5/32, 5/32, 5/100, 5/100] \tag{A.81}$$

and $\boldsymbol{M}_i$ are all identity matrices.

A.16 F_{16}: Rotated Version of Hybrid Composition Function F_{15}

$$F_{16}(\boldsymbol{x}) = \sum_{i=1}^{n}\{w_i \times [f_i'((\boldsymbol{x} - \boldsymbol{o}_i)\lambda_i \times \boldsymbol{M}_i) + bias_i]\} + f_{bias_{16}} \tag{A.82}$$

where $\boldsymbol{M}_i$ are linear transformation matrices with condition number 2 and all other settings are the same as for F_{15}.

A.17 F_{17}: F_{16} with Noise in Fitness

$$F_{17}(\boldsymbol{x}) = G(\boldsymbol{x})(1 + 0.2|N(0,1)|) + f_{bias_{17}} \tag{A.83}$$

where

$$G(\boldsymbol{x}) = F_{16} - f_{bias_{16}} \tag{A.84}$$

and all other settings are the same as for F_{16}.

Appendix B

Results

Throughout this thesis each algorithm was evaluated over 30 independent simulation runs on each of the 17 CEC 2005 problems in dimensions 10, 30, and 50. These results are presented in this appendix. The symbols μ and σ denote the mean and standard deviation associated with the corresponding performance measure and $\#FEs$ denotes the number of function evaluations which were needed to reach the global optimum within a specified accuracy.

Table B.1: Results of alternative selection strategy evaluation: RAND and BOLT.

Prob (Dims)	RAND				BOLT			
	FFV		# FEs		FFV		# FEs	
	μ	σ	μ	σ	μ	σ	μ	σ
1(10)	$1.00E-06$	0	12133	3.9421	$1.00E-06$	0	11773	4.5329
1(30)	$1.00E-06$	0	55670	21.97	$1.00E-06$	0	55860	24.497
1(50)	$1.00E-06$	0	$1.2551E+05$	45.509	$1.00E-06$	0	$1.3172E+05$	81.211
2(10)	$1.00E-06$	0	22677	11.796	$1.00E-06$	0	19180	11.433
2(30)	$1.00E-06$	0	$2.0905E+05$	93.899	$1.00E-06$	0	$1.6691E+05$	77.496
2(50)	$1.00E-06$	0	500000	0	0.000001	0	$4.8073E+05$	161.4
3(10)	$2.39E+04$	22789	100000	0	3407.6	4247.8	100000	0
3(30)	$3.3659E+05$	$1.5152E+05$	300000	0	44624	27185	$3.00E+05$	0
3(50)	$7.3423E+05$	$2.4609E+05$	$5.00E+05$	0	$1.2362E+05$	40507	$5.00E+05$	0
4(10)	$1.00E-06$	0	24847	19.278	$1.00E-06$	0	20337	9.1406
4(30)	0.012333	0.013565	300000	0	0.000001	0	$2.5561E+05$	295.81
4(50)	205.86	155.45	$5.00E+05$	0	15.676	14.74	$5.00E+05$	0
5(10)	$1.00E-06$	0	14767	13.732	$1.00E-06$	0	15087	8.2284
5(30)	1926.7	950.27	$3.00E+05$	0	603.57	526.04	$3.00E+05$	0
5(50)	5501	1277.5	$5.00E+05$	0	**3508.9**	737.17	$5.00E+05$	0
6(10)	0.00033333	0.0018257	52233	147.87	0.00033333	0.0018257	39973	132.6
6(30)	1.259	2.8495	$2.5992E+05$	447.59	0.26567	1.0097	$2.2974E+05$	455.88
6(50)	4.7007	5.3583	$4.7203E+05$	462.25	7.3677	17.985	$4.6228E+05$	648.4
7(10)	**1267**	$4.6252E-13$	100000	0	**1267**	$4.6252E-13$	100000	0
7(30)	**4696.3**	$2.7751E-12$	300000	0	**4696.3**	$2.7751E-12$	300000	0
7(50)	**6195.3**	0	500000	0	**6195.3**	0	500000	0
8(10)	**20.068**	0.10825	$1.00E+05$	0	20.078	0.11394	$1.00E+05$	0
8(30)	20.305	0.28615	$3.00E+05$	0	20.422	0.34431	$3.00E+05$	0
8(50)	20.801	0.40599	$5.00E+05$	0	20.883	0.25636	$5.00E+05$	0
9(10)	0.39967	0.75825	70580	249.18	0.465	0.88821	73627	241.73
9(30)	2.575	1.0648	300000	0	2.2353	1.1727	300000	0
9(50)	10.725	3.6969	$5.00E+05$	0	10.203	2.8382	500000	0
10(10)	13.908	5.3673	$1.00E+05$	0	12.878	8.5088	$1.00E+05$	0
10(30)	60.173	30.423	$3.00E+05$	0	48.433	23.21	$3.00E+05$	0
10(50)	83.543	45.391	$5.00E+05$	0	77.082	45.766	$5.00E+05$	0
11(10)	4.2416	2.2042	$1.00E+05$	0	4.1225	2.1277	$1.00E+05$	0
11(30)	24.517	5.9022	$3.00E+05$	0	25.686	6.2736	$3.00E+05$	0
11(50)	52.244	8.1035	$5.00E+05$	0	50.327	12.524	$5.00E+05$	0
12(10)	164.05	488.11	62747	330.19	126.6	383.64	61440	372.87
12(30)	1940.7	2299.6	$3.00E+05$	0	**1744.7**	2693.7	$3.00E+05$	0
12(50)	11767	9228.3	$5.00E+05$	0	**11438**	13494	$5.00E+05$	0
13(10)	0.42667	0.19838	$1.00E+05$	0	**0.34433**	0.17853	$1.00E+05$	1.278
13(30)	1.3217	0.61569	$3.00E+05$	0	1.494	0.50914	$3.00E+05$	0
13(50)	3.0537	0.99364	$5.00E+05$	0	**3.1593**	1.2158	$5.00E+05$	0
14(10)	3.4243	0.3228	$1.00E+05$	0	3.2127	0.47059	$1.00E+05$	0
14(30)	12.829	0.549	$3.00E+05$	0	12.752	0.40445	$3.00E+05$	0
14(50)	22.361	0.61917	$5.00E+05$	0	22.4	0.44187	$5.00E+05$	0
15(10)	138.33	171.46	77770	311.9	283.31	176.43	93610	205.6
15(30)	323.43	100.71	300000	0	**292**	129.29	$3.00E+05$	0
15(50)	291.43	109.94	$5.00E+05$	0	287.01	90.412	$5.00E+05$	0
16(10)	127.16	21.95	$1.00E+05$	0	123.19	22.532	$1.00E+05$	0
16(30)	**172.72**	139.15	$3.00E+05$	0	208.16	166.01	$3.00E+05$	0
16(50)	161.43	137.75	$5.00E+05$	0	**141.81**	125.8	$5.00E+05$	0
17(10)	129.9	20.159	$1.00E+05$	0	121.53	15.194	$1.00E+05$	0
17(30)	191.11	171.16	$3.00E+05$	0	185.49	160.91	$3.00E+05$	0
17(50)	153.99	140.37	$5.00E+05$	0	**101.45**	94.849	$5.00E+05$	0

Table B.2: Results of alternative selection strategy evaluation: TOUR and ROUL.

Prob (Dims)	TOUR				ROUL			
	FFV		# FEs		FFV		# FEs	
	μ	σ	μ	σ	μ	σ	μ	σ
1(10)	$1.00E-06$	0	12000	17.838	0.000001	0	**$1.10E+04$**	23.715
1(30)	$1.00E-06$	0	**32530**	48.32	0.000001	0	$3.97E+04$	167.4
1(50)	$1.00E-06$	0	**49257**	41.384	0.000001	0	$7.70E+04$	713.61
2(10)	$1.00E-06$	0	19573	28.412	0.0066675	0.034674	$2.23E+04$	267.41
2(30)	$1.00E-06$	0	$1.0964E+05$	106.07	89.105	164.55	$2.4301E+05$	711.26
2(50)	$1.00E-06$	0	$2.7483E+05$	174.2	1371.1	2146.1	$4.8821E+05$	160.33
3(10)	$1.00E-06$	0	51427	108.23	73381	$2.3949E+05$	32753	342.48
3(30)	1674.3	4081.6	$2.9991E+05$	4.9295	$2.0525E+06$	$1.6702E+06$	$3.00E+05$	0
3(50)	$1.0102E+05$	72787	500000	0	$2.7677E+06$	$2.9294E+06$	$5.00E+05$	0
4(10)	$1.00E-06$	0	19903	22.3	1.9597	10.696	$2.01E+04$	225.05
4(30)	$1.00E-06$	0	$1.6536E+05$	126.26	9335.9	7293.7	$3.00E+05$	0
4(50)	0.041334	0.14731	500000	0	26659	9608.3	$5.00E+05$	0
5(10)	$1.00E-06$	0	**11877**	18.765	0.023001	0.12598	$2.56E+04$	329.5
5(30)	**385.52**	485.25	300000	0	3477.9	1590.8	$3.00E+05$	0
5(50)	4390.9	1152.2	500000	0	9849.4	3088.7	$5.00E+05$	0
6(10)	0.00066667	0.0025371	34543	133.38	6.1673	14.902	$6.76E+04$	387.6
6(30)	0.66367	1.5085	$2.026E+05$	589.31	15.669	26.575	$2.8612E+05$	482.69
6(50)	1.5083	2.0368	**$4.2338E+05$**	876.75	251.16	1172	$4.8926E+05$	588.25
7(10)	**1267**	$4.6252E-13$	$1.00E+05$	0	**1267**	0.050742	$1.00E+05$	0
7(30)	**4696.3**	$2.7751E-12$	$3.00E+05$	0	**4696.3**	$2.7751E-12$	$3.00E+05$	0
7(50)	**6195.3**	0	$5.00E+05$	0	6195.4	0.58424	$5.00E+05$	0
8(10)	20.07	0.10685	$1.00E+05$	0	20.193	0.13031	$1.00E+05$	0
8(30)	**20.156**	0.11174	$3.00E+05$	0	20.712	0.22643	$3.00E+05$	0
8(50)	20.78	0.39956	$5.00E+05$	0	20.963	0.16961	$5.00E+05$	0
9(10)	0.13667	0.33647	**48307**	267.34	0.926	1.4707	51203	407.26
9(30)	5.022	2.2953	$3.00E+05$	0	12.587	18.626	$1.9681E+05$	818.69
9(50)	25.985	11.035	$5.00E+05$	0	42.706	34.941	500000	0
10(10)	18.477	8.1232	100000	0	13.877	9.0904	100000	0
10(30)	1847.5	9614.7	$2.9487E+05$	157.95	94.226	21.929	300000	0
10(50)	110.1	49.509	$5.00E+05$	0	203.8	59.842	500000	0
11(10)	6.4184	1.5567	100000	0	6.5747	1.4775	$1.00E+05$	0
11(30)	28.466	5.2396	$3.00E+05$	0	31.55	5.3227	$3.00E+05$	0
11(50)	50.889	7.2092	$5.00E+05$	0	57.48	4.4024	$5.00E+05$	0
12(10)	152.05	397.6	**51143**	343.45	759.03	962.31	87037	297.33
12(30)	6340.1	7897.4	$2.8579E+05$	329.64	12948	9977.6	$3.00E+05$	0
12(50)	51571	36242	$4.9726E+05$	149.89	81345	50154	$5.00E+05$	0
13(10)	0.353	0.13522	$1.00E+05$	0	0.52533	0.22598	100000	0
13(30)	1.7873	0.56335	$3.00E+05$	0	2.877	1.093	$3.00E+05$	0
13(50)	5.36	2.4864	$5.00E+05$	0	7.5513	3.2891	$5.00E+05$	0
14(10)	3.463	0.33085	$1.00E+05$	0	3.516	0.3841	100000	0
14(30)	13.03	0.44696	$3.00E+05$	0	12.975	0.44704	$3.00E+05$	0
14(50)	22.488	0.62213	$5.00E+05$	0	22.532	0.43019	$5.00E+05$	0
15(10)	251.39	430.91	**$7.51E+04$**	326.97	295.59	175.46	$9.49E+04$	192.62
15(30)	350.37	111.02	$3.00E+05$	0	318.92	127.35	$2.8797E+05$	457.96
15(50)	**282.18**	92.661	$4.9338E+05$	362.77	333.87	85.557	$5.00E+05$	0
16(10)	141.8	24.462	$1.00E+05$	0	131.44	36.34	$1.00E+05$	0
16(30)	212.27	137.94	$3.00E+05$	0	254.86	162.86	$3.00E+05$	0
16(50)	183.93	117.44	$5.00E+05$	0	209	113.18	$5.00E+05$	0
17(10)	141.53	24.311	$1.00E+05$	0	129.54	22.625	100000	0
17(30)	207.26	168.39	$3.00E+05$	0	241.11	166.85	$3.00E+05$	0
17(50)	134.75	109.44	$5.00E+05$	0	193.69	100.78	$5.00E+05$	0

Table B.3: Results of alternative selection strategy evaluation: RANK and TSHH.

Prob (Dims)	RANK				TSHH			
	FFV		# FEs		FFV		# FEs	
	μ	σ	μ	σ	μ	σ	μ	σ
1(10)	$1.00E-06$	0	11573	5.2976	$1.00E-06$	0	13110	7.8008
1(30)	$1.00E-06$	0	54647	24.761	$1.00E-06$	0	37740	10.071
1(50)	$1.00E-06$	0	$1.2791E+05$	74.657	$1.00E-06$	0	59243	11.869
2(10)	0.000001	0	$1.84E+04$	8.3422	0.000001	0	$\mathbf{1.54E+04}$	7.6417
2(30)	0.000001	0	$1.6229E+05$	85.968	0.000001	0	$\mathbf{6.94E+04}$	83.232
2(50)	0.000001	0	$4.6883E+05$	235.01	0.000001	0	$\mathbf{1.283E+05}$	133.57
3(10)	5786.5	8405.3	$1.00E+05$	0	0.000001	0	$\mathbf{2.24E+04}$	25.929
3(30)	$1.5473E+05$	87108	$3.00E+05$	0	**0.000001**	0	$1.5136E+05$	167.12
3(50)	$4.0364E+05$	$2.0383E+05$	$5.00E+05$	0	**0.000001**	0	$3.2973E+05$	218.71
4(10)	0.000001	0	$2.01E+04$	11.184	0.000001	0	$\mathbf{1.75E+04}$	9.4127
4(30)	0.000001	0	$2.4797E+05$	307.44	0.000001	0	$\mathbf{9.47E+04}$	131.24
4(50)	22.01	22.715	$5.00E+05$	0	**0.000001**	0	$1.9176E+05$	230.62
5(10)	0.000001	0	$1.55E+04$	8.5164	0.000001	0	$1.68E+04$	14.891
5(30)	416.17	485.42	$3.00E+05$	0	548.65	592.86	$3.00E+05$	0
5(50)	3861.9	898.84	$5.00E+05$	0	5295.5	1444.8	$5.00E+05$	0
6(10)	0.13367	0.72646	$4.29E+04$	191.56	0.001	0.0030513	$\mathbf{3.23E+04}$	112.88
6(30)	0	0	$2.1922E+05$	393.09	0.398	1.2144	$\mathbf{1.8793E+05}$	490.91
6(50)	6.9587	17.669	$4.5162E+05$	589.5	1.9007	2.3675	$4.2378E+05$	859.41
7(10)	**1267**	$4.6252E-13$	$1.00E+05$	0	**1267**	$4.6252E-13$	$1.00E+05$	0
7(30)	**4696.3**	$2.7751E-12$	$3.00E+05$	0	**4696.3**	$2.7751E-12$	$3.00E+05$	0
7(50)	**6195.3**	0	$5.00E+05$	0	**6195.3**	0	$5.00E+05$	0
8(10)	20.069	0.12506	$1.00E+05$	0	20.079	0.13181	$1.00E+05$	0
8(30)	20.513	0.34172	$3.00E+05$	0	20.177	0.12282	$3.00E+05$	0
8(50)	20.756	0.37242	$5.00E+05$	0	**20.572**	0.34532	$5.00E+05$	0
9(10)	0.13833	0.33579	63760	254.55	0.13533	0.337	63990	233.63
9(30)	2.3093	1.3892	$2.9512E+05$	189.69	10.609	4.0744	$2.9889E+05$	60.615
9(50)	**9.467**	3.0466	500000	0	35.658	10.958	500000	0
10(10)	**10.793**	5.1619	$1.00E+05$	0	17.944	9.5247	$1.00E+05$	0
10(30)	**44.331**	19.002	$3.00E+05$	0	73.34	23.689	$3.00E+05$	0
10(50)	**68.692**	28.398	$5.00E+05$	0	146.95	51.373	$5.00E+05$	0
11(10)	**3.6913**	2.2	$9.51E+04$	187.51	4.8408	1.7899	$1.00E+05$	0
11(30)	26.457	6.0631	$3.00E+05$	0	**24.491**	5.142	$3.00E+05$	0
11(50)	55.252	6.2803	$5.00E+05$	0	**49.56**	8.5591	$5.00E+05$	0
12(10)	443.11	687.88	$6.87E+04$	354.7	257.98	543.17	$6.50E+04$	390.76
12(30)	1896.8	2739.8	$2.9808E+05$	104.98	5283	4418.4	$3.00E+05$	0
12(50)	11925	9612.8	$5.00E+05$	0	53453	39068	$5.00E+05$	0
13(10)	0.404	0.19342	$1.00E+05$	0	0.5	0.16688	$1.00E+05$	0
13(30)	**1.269**	0.49143	$3.00E+05$	0	3.1493	1.1606	$3.00E+05$	0
13(50)	3.221	1.33	$5.00E+05$	0	7.5257	3.7335	$5.00E+05$	0
14(10)	**3.088**	0.44933	$1.00E+05$	0	3.5283	0.3483	$1.00E+05$	0
14(30)	**12.704**	0.54604	$3.00E+05$	0	13.133	0.49901	$3.00E+05$	0
14(50)	**22.348**	0.56163	$5.00E+05$	0	22.639	0.6383	$5.00E+05$	0
15(10)	237.62	202.61	88553	219.44	280.45	192.27	96013	158.17
15(30)	344.53	124.61	300000	0	323.78	104.54	300000	0
15(50)	317.73	99.69	$5.00E+05$	0	294.62	104.51	$5.00E+05$	0
16(10)	**114.94**	12.749	$1.00E+05$	0	121	18.555	$1.00E+05$	0
16(30)	214.04	182.23	$3.00E+05$	0	246.63	162.46	$3.00E+05$	0
16(50)	154.39	131.47	$5.00E+05$	0	154.46	108.91	$5.00E+05$	0
17(10)	**117.08**	12.767	$1.00E+05$	0	130.06	21.297	$1.00E+05$	0
17(30)	183.12	176.64	$3.00E+05$	0	**151.95**	120.65	$3.00E+05$	0
17(50)	163.56	148.27	$5.00E+05$	0	162.41	139.74	$5.00E+05$	0

Table B.4: Results of alternative local search selection strategy evaluation on the 2005 IEEE CEC benchmark problem set: LS1HH and LS2HH.

Prob (Dims)	LS1HH				LS2HH			
	FFV		# FEs		FFV		# FEs	
	μ	σ	μ	σ	μ	σ	μ	σ
1(10)	$1.00E-06$	0	**13131**	499.95	$1.00E-06$	0	13205	590.59
1(30)	$1.00E-06$	0	**69212**	4269.7	$1.00E-06$	0	69832	4304.7
1(50)	$1.00E-06$	0	$1.7709E+05$	20587	$1.00E-06$	0	$1.7458E+05$	18988
2(10)	$1.00E-06$	0	15325	1118.4	$1.00E-06$	0	15773	1145.2
2(30)	$1.00E-06$	0	$\mathbf{1.1457E+05}$	16881	$1.00E-06$	0	$1.1821E+05$	19876
2(50)	$1.00E-06$	0	$3.8645E+05$	$1.0245E+05$	$1.00E-06$	0	$3.9112E+05$	$1.0488E+05$
3(10)	$1.00E-06$	0	24546	3222.3	$1.00E-06$	0	**24383**	3428.6
3(30)	810.08	3062	$\mathbf{2.9586E+05}$	18327	811.5	2115.4	$2.9699E+05$	15519
3(50)	$1.6086E+05$	78665	$5.00E+05$	0	$\mathbf{1.2457E+05}$	93799	$5.00E+05$	0
4(10)	$1.00E-06$	0	**17701**	1235.6	$1.00E-06$	0	17808	1383.5
4(30)	$1.00E-06$	0	$2.1283E+05$	44233	$1.00E-06$	0	$\mathbf{2.0331E+05}$	34332
4(50)	20.47	43.422	$4.9967E+05$	1908.1	**8.1087**	17.909	$5.00E+05$	523.99
5(10)	$1.00E-06$	0	**18974**	1325.3	$1.00E-06$	0	19088	1034.1
5(30)	307.76	256.03	$3.00E+05$	0	299.83	353.17	$3.00E+05$	0
5(50)	**3437**	808.59	$5.00E+05$	0	3738.4	769.24	$5.00E+05$	0
6(10)	0.00033333	0.0018257	40633	10451	0.133	0.72658	**40452**	15190
6(30)	2.343	2.8194	$\mathbf{2.8505E+05}$	34959	3.7687	12.187	$2.9229E+05$	15650
6(50)	**33.918**	39.14	$4.9919E+05$	5017.1	40.04	47.379	$5.00E+05$	0
7(10)	**0.107**	0.086389	$1.00E+05$	0	0.131	0.11161	$1.00E+05$	0
7(30)	0.0063333	0.0085029	$1.7798E+05$	$1.1646E+05$	0.01	0.013131	$2.0306E+05$	$1.1305E+05$
7(50)	0.0023333	0.0067891	$2.3673E+05$	$1.3455E+05$	0.004	0.0093218	$2.6732E+05$	$1.5592E+05$
8(10)	**20.036**	0.079719	$1.00E+05$	0	20.087	0.13664	$1.00E+05$	0
8(30)	**20.178**	0.12962	$3.00E+05$	0	20.232	0.12098	$3.00E+05$	0
8(50)	20.898	0.3121	$5.00E+05$	0	20.858	0.34138	$5.00E+05$	0
9(10)	0.20133	0.39601	56521	28626	0.137	0.42669	**46927**	25418
9(30)	**3.0743**	1.5724	$2.9871E+05$	5481.7	3.2047	1.4938	$3.00E+05$	0
9(50)	**17.708**	4.5149	$5.00E+05$	0	20.042	6.5921	$5.00E+05$	0
10(10)	18.665	9.2106	$1.00E+05$	0	18.475	7.8114	$1.00E+05$	0
10(30)	63.803	25.807	$3.00E+05$	0	61.737	24.805	$3.00E+05$	0
10(50)	294.99	61.918	$5.00E+05$	0	313.54	39.575	$5.00E+05$	0
11(10)	5.488	1.6099	$1.00E+05$	0	**5.484**	1.4235	$1.00E+05$	0
11(30)	26.027	6.2231	$3.00E+05$	0	**25.349**	5.9663	$3.00E+05$	0
11(50)	42.707	10.078	$5.00E+05$	0	**41.928**	10.158	$5.00E+05$	0
12(10)	585.49	771.15	64927	38884	416.18	645.96	67772	37142
12(30)	**3076.7**	2840.5	$3.00E+05$	0	4955.8	5179.1	$3.00E+05$	0
12(50)	**17044**	12000	$5.00E+05$	0	21114	15077	$5.00E+05$	0
13(10)	**0.44033**	0.12944	$1.00E+05$	0	0.47067	0.15993	$1.00E+05$	0
13(30)	5.556	4.9635	$3.00E+05$	0	**3.3303**	3.0567	$3.00E+05$	0
13(50)	**10.079**	8.7701	$5.00E+05$	0	11.19	8.9088	$5.00E+05$	0
14(10)	**3.5607**	0.36167	$1.00E+05$	0	3.623	0.32138	$1.00E+05$	0
14(30)	13.125	0.34038	$3.00E+05$	0	**13.068**	0.43086	$3.00E+05$	0
14(50)	22.673	0.44734	$5.00E+05$	0	22.695	0.39549	$5.00E+05$	0
15(10)	291.2	192.47	86387	26640	246.86	212.57	83231	27190
15(30)	**292.06**	132.22	$2.9854E+05$	8657.7	294.68	114.47	$3.00E+05$	0
15(50)	300.73	86.785	$5.00E+05$	0	299.09	114.19	$5.00E+05$	0
16(10)	136.96	21.556	$1.00E+05$	0	127.94	14.729	$1.00E+05$	0
16(30)	180.9	140.38	$3.00E+05$	0	**152.71**	119.62	$3.00E+05$	0
16(50)	**241.85**	111.51	$5.00E+05$	0	249.92	91.765	$5.00E+05$	0
17(10)	**130.72**	24.751	$1.00E+05$	0	132.96	21.29	$1.00E+05$	0
17(30)	**149.32**	161.11	$3.00E+05$	0	237.57	194	$3.00E+05$	0
17(50)	308.73	77.866	$5.00E+05$	0	**286.36**	85.684	$5.00E+05$	0

Table B.5: Results of alternative local search selection strategy evaluation on the 2005 IEEE CEC benchmark problem set: LS3HH and LS4HH.

Prob (Dims)	LS3HH				LS4HH			
	FFV		# FEs		FFV		# FEs	
	μ	σ	μ	σ	μ	σ	μ	σ
1(10)	$1.00E-06$	0	13527	772.74	$1.00E-06$	0	17349	2037.9
1(30)	$1.00E-06$	0	70063	4643.4	$1.00E-06$	0	$1.1726E+05$	17620
1(50)	$1.00E-06$	0	**$1.7249E+05$**	18461	$1.00E-06$	0	$2.7417E+05$	75811
2(10)	$1.00E-06$	0	**15074**	1378.5	$1.00E-06$	0	23836	4870.7
2(30)	$1.00E-06$	0	$1.1908E+05$	21837	0.411	1.3529	$2.5866E+05$	37765
2(50)	$1.00E-06$	0	**$3.5727E+05$**	$1.0133E+05$	2594.7	2039.5	$5.00E+05$	288.12
3(10)	$1.00E-06$	0	24457	2855.7	$1.00E-06$	0	55849	18574
3(30)	2494.6	7033.2	$2.9729E+05$	9499.2	$4.7311E+05$	$5.4541E+05$	$3.00E+05$	127.48
3(50)	$1.3351E+05$	66530	$5.00E+05$	0	$1.3999E+06$	$2.1202E+06$	$5.00E+05$	98.611
4(10)	$1.00E-06$	0	18045	967.98	$1.00E-06$	0	29194	6913
4(30)	$1.00E-06$	0	$2.063E+05$	43004	1136.4	1075.5	$3.00E+05$	207.47
4(50)	16.21	36.188	$4.9745E+05$	10417	30599	10319	$5.00E+05$	380.49
5(10)	$1.00E-06$	0	19033	1371.8	$1.00E-06$	0	23927	2009.6
5(30)	**287.27**	305.33	$3.00E+05$	0	1237.8	749.04	$3.00E+05$	63.568
5(50)	3805.9	1110.2	$5.00E+05$	0	5699.2	1395.3	$5.00E+05$	139.38
6(10)	0.26567	1.0097	40900	18079	0.13267	0.72665	49792	12304
6(30)	1.6943	2.7528	$2.9246E+05$	16037	16.3	26.287	$3.00E+05$	396.98
6(50)	63.946	70.188	$5.00E+05$	0	61.178	60.443	$5.00E+05$	113.97
7(10)	0.169	0.15121	$1.00E+05$	0	0.12933	0.080727	97179	15984
7(30)	0.0043333	0.008172	$1.5782E+05$	$1.1044E+05$	0.0086667	0.012794	$1.92E+05$	$1.1024E+05$
7(50)	0.0036667	0.0096431	$2.7225E+05$	$1.5275E+05$	0.0033333	0.008023	$2.8849E+05$	$1.5355E+05$
8(10)	20.086	0.11574	$1.00E+05$	0	20.065	0.099956	$1.00E+05$	75.28
8(30)	20.214	0.11448	$3.00E+05$	0	20.761	0.20356	$3.00E+05$	267.72
8(50)	**20.846**	0.32715	$5.00E+05$	0	21.183	0.034873	$5.00E+05$	524.46
9(10)	0.072	0.24686	54745	24158	0.072	0.24686	51278	22297
9(30)	3.3373	1.3947	$3.00E+05$	0	20.825	6.918	$3.00E+05$	246.29
9(50)	18.211	4.1952	$5.00E+05$	0	100.85	25.588	$5.00E+05$	401.21
10(10)	16.073	7.4463	$1.00E+05$	0	**15.374**	6.7704	$1.00E+05$	94.822
10(30)	**54.826**	22.228	$3.00E+05$	0	64.222	32.241	$3.00E+05$	139.77
10(50)	**290.01**	62.998	$5.00E+05$	0	386.85	32.217	$5.00E+05$	387.23
11(10)	5.5902	2.1572	$1.00E+05$	0	6.4702	1.4468	$1.00E+05$	57.744
11(30)	26.4	4.3447	$3.00E+05$	0	28.631	5.6437	$3.00E+05$	245.84
11(50)	45.466	11.535	$5.00E+05$	0	67.731	9.2689	$5.00E+05$	357.03
12(10)	290.46	602.81	**51896**	36729	365.97	628.79	70247	28282
12(30)	3825.1	5366.7	$3.00E+05$	0	9299.1	8224.2	$3.00E+05$	72.512
12(50)	20098	13926	$5.00E+05$	0	$1.9387E+05$	74653	$5.00E+05$	336.18
13(10)	0.48833	0.13847	$1.00E+05$	0	0.484	0.17379	$1.00E+05$	80.376
13(30)	3.6953	3.2199	$3.00E+05$	0	11.464	4.1071	$3.00E+05$	279.01
13(50)	14.089	10.555	$5.00E+05$	0	34.4	4.0116	$5.00E+05$	397.13
14(10)	3.5897	0.33956	$1.00E+05$	0	3.6493	0.29484	$1.00E+05$	99.151
14(30)	13.099	0.43684	$3.00E+05$	0	13.257	0.38212	$3.00E+05$	220.9
14(50)	**22.596**	0.55778	$5.00E+05$	0	23.107	0.38498	$5.00E+05$	447.31
15(10)	274.57	205.3	83308	28721	302.11	184.42	87039	25765
15(30)	307.45	104.01	$3.00E+05$	0	323.65	104.18	$3.00E+05$	137.15
15(50)	**298.89**	114.33	$5.00E+05$	0	609.82	277.93	$2.5073E+05$	$2.54E+05$
16(10)	**126.12**	19.484	$1.00E+05$	0	142.04	28.859	$1.00E+05$	79.067
16(30)	231.03	164.31	$3.00E+05$	0	187.49	134.05	$3.00E+05$	179.61
16(50)	270	92.302	$5.00E+05$	0	630.28	290.48	$2.1749E+05$	$2.518E+05$
17(10)	131.98	23.589	$1.00E+05$	0	135.44	25.872	$1.00E+05$	66.806
17(30)	204.52	180.29	$3.00E+05$	0	261.87	167.5	$3.00E+05$	226.97
17(50)	301.97	75.472	$5.00E+05$	0	618.96	285.7	$2.3404E+05$	$2.5339E+05$

Table B.6: Results of the investigation into alternative solution space diversity control mechanisms: TSHH (no local search) and LSHH (constant local search).

Prob (Dims)	TSHH				LSHH			
	FFV		# FEs		FFV		# FEs	
	μ	σ	μ	σ	μ	σ	μ	σ
1(10)	$1.00E-06$	0	**11870**	538.93	$1.00E-06$	0	13205	590.59
1(30)	$1.00E-06$	0	**52983**	2723.3	$1.00E-06$	0	69832	4304.7
1(50)	$1.00E-06$	0	$\mathbf{1.1563E+05}$	9985.8	$1.00E-06$	0	$1.7458E+05$	18988
2(10)	$1.00E-06$	0	14013	1246.7	$1.00E-06$	0	15773	1145.2
2(30)	$1.00E-06$	0	**90727**	15876	$1.00E-06$	0	$1.1821E+05$	19876
2(50)	$1.00E-06$	0	$\mathbf{2.4547E+05}$	78036	$1.00E-06$	0	$3.9112E+05$	$1.0488E+05$
3(10)	$1.00E-06$	0	22750	3003	$1.00E-06$	0	24383	3428.6
3(30)	295.35	998.74	$2.7856E+05$	27782	811.5	2115.4	$2.9699E+05$	15519
3(50)	**78960**	71769	$5.00E+05$	0	$1.2457E+05$	93799	$5.00E+05$	0
4(10)	$1.00E-06$	0	**15853**	1341.8	$1.00E-06$	0	17808	1383.5
4(30)	$1.00E-06$	0	$1.5021E+05$	27962	$1.00E-06$	0	$2.0331E+05$	34332
4(50)	1.193	5.5772	$\mathbf{4.6643E+05}$	60163	8.1087	17.909	$5.00E+05$	523.99
5(10)	$1.00E-06$	0	17110	883.31	$1.00E-06$	0	19088	1034.1
5(30)	174.5	278.17	$3.00E+05$	0	299.83	353.17	$3.00E+05$	0
5(50)	3237	760.19	$5.00E+05$	0	3738.4	769.24	$5.00E+05$	0
6(10)	0	0	**33417**	7388.8	0.133	0.72658	40452	15190
6(30)	0.13267	0.72665	$\mathbf{2.4344E+05}$	33935	3.7687	12.187	$2.9229E+05$	15650
6(50)	4.8457	13.791	$\mathbf{4.7795E+05}$	35158	40.04	47.379	$5.00E+05$	0
7(10)	0.162	0.13632	$1.00E+05$	0	**0.131**	0.11161	$1.00E+05$	0
7(30)	0.0036667	0.0080872	$\mathbf{1.2987E+05}$	$1.1343E+05$	0.01	0.013131	$2.0306E+05$	$1.1305E+05$
7(50)	0.0033333	0.0095893	$\mathbf{2.0263E+05}$	$1.6704E+05$	0.004	0.0093218	$2.6732E+05$	$1.5592E+05$
8(10)	20.06	0.086681	$1.00E+05$	0	20.087	0.13664	$1.00E+05$	0
8(30)	**20.169**	0.12041	$3.00E+05$	0	20.232	0.12098	$3.00E+05$	0
8(50)	**20.796**	0.37683	$5.00E+05$	0	20.858	0.34138	$5.00E+05$	0
9(10)	0.005	0.0050855	**43320**	19202	0.137	0.42669	46927	25418
9(30)	**2.3763**	1.3964	$2.9771E+05$	10079	3.2047	1.4938	$3.00E+05$	0
9(50)	14.373	3.3496	$5.00E+05$	0	20.042	6.5921	$5.00E+05$	0
10(10)	15.556	9.1746	$1.00E+05$	0	18.475	7.8114	$1.00E+05$	0
10(30)	**55.768**	19.838	$3.00E+05$	0	61.737	24.805	$3.00E+05$	0
10(50)	**57.474**	48.572	$5.00E+05$	0	313.54	39.575	$5.00E+05$	0
11(10)	5.0464	2.2525	97283	14880	5.484	1.4235	$1.00E+05$	0
11(30)	**24.378**	5.1224	$3.00E+05$	0	25.349	5.9663	$3.00E+05$	0
11(50)	44.831	9.4654	$5.00E+05$	0	**41.928**	10.158	$5.00E+05$	0
12(10)	302.86	546.06	69543	39317	416.18	645.96	67772	37142
12(30)	2611.5	3622.3	$2.9488E+05$	20162	4955.8	5179.1	$3.00E+05$	0
12(50)	21252	14310	$5.00E+05$	0	21114	15077	$5.00E+05$	0
13(10)	**0.43267**	0.16282	99050	5203.4	0.47067	0.15993	$1.00E+05$	0
13(30)	**2.0187**	0.44262	$3.00E+05$	0	3.3303	3.0567	$3.00E+05$	0
13(50)	**4.1393**	0.92703	$5.00E+05$	0	11.19	8.9088	$5.00E+05$	0
14(10)	3.6397	0.29122	$1.00E+05$	0	**3.623**	0.32138	$1.00E+05$	0
14(30)	13.14	0.44011	$3.00E+05$	0	**13.068**	0.43086	$3.00E+05$	0
14(50)	**22.527**	0.3707	$5.00E+05$	0	22.695	0.39549	$5.00E+05$	0
15(10)	258.98	210.57	80170	30406	246.86	212.57	83231	27190
15(30)	373.72	108.43	$3.00E+05$	0	**294.68**	114.47	$3.00E+05$	0
15(50)	**259.17**	114.32	$5.00E+05$	0	299.09	114.19	$5.00E+05$	0
16(10)	138.39	25.216	$1.00E+05$	0	**127.94**	14.729	$1.00E+05$	0
16(30)	153.27	128.88	$3.00E+05$	0	**152.71**	119.62	$3.00E+05$	0
16(50)	**74.477**	45.497	$5.00E+05$	0	249.92	91.765	$5.00E+05$	0
17(10)	133.28	22.828	$1.00E+05$	0	132.96	21.29	$1.00E+05$	0
17(30)	200.71	169.72	$3.00E+05$	0	237.57	194	$3.00E+05$	0
17(50)	168.33	164.8	$5.00E+05$	0	286.36	85.684	$5.00E+05$	0

Table B.7: Results of the investigation into alternative solution space diversity control mechanisms: ALSHH (adaptive local search).

Prob (Dims)	ALSHH			
	FFV		# FEs	
	μ	σ	μ	σ
1(10)	$1.00E-06$	0	12087	537.38
1(30)	$1.00E-06$	0	53170	3025.3
1(50)	$1.00E-06$	0	$1.1587E+05$	10855
2(10)	$1.00E-06$	0	**13907**	1016.7
2(30)	$1.00E-06$	0	93030	18560
2(50)	$1.00E-06$	0	$2.577E+05$	71368
3(10)	$1.00E-06$	0	**21817**	3112.5
3(30)	2006.8	10120	$\mathbf{2.656E+05}$	39777
3(50)	88482	57891	$5.00E+05$	0
4(10)	$1.00E-06$	0	16117	1480.9
4(30)	$1.00E-06$	0	$\mathbf{1.4849E+05}$	32113
4(50)	1.821	8.6887	$4.7791E+05$	50187
5(10)	$1.00E-06$	0	**16927**	1170.6
5(30)	**139.52**	177.54	$3.00E+05$	0
5(50)	**3064.4**	877.39	$5.00E+05$	0
6(10)	0.13333	0.72652	36263	15400
6(30)	0.67233	1.5052	$2.5901E+05$	34525
6(50)	5.356	4.7524	$4.9236E+05$	26607
7(10)	0.13167	0.17916	$1.00E+05$	30.731
7(30)	0.006	0.0093218	$1.5659E+05$	$1.1934E+05$
7(50)	0.005	0.011671	$2.1313E+05$	$1.761E+05$
8(10)	**20.059**	0.11931	$1.00E+05$	9.3526
8(30)	20.193	0.1094	$3.00E+05$	39.224
8(50)	20.94	0.34657	$5.00E+05$	47.324
9(10)	0.23333	0.49446	53361	30000
9(30)	2.5743	1.5592	$3.00E+05$	0
9(50)	**11.182**	4.1157	$5.00E+05$	0
10(10)	**14.644**	8.0407	$1.00E+05$	26.704
10(30)	59.464	31.099	$3.00E+05$	32.641
10(50)	60.881	51.26	$5.00E+05$	20.197
11(10)	5.8024	2.2642	**97273**	14953
11(30)	25.704	6.5042	$3.00E+05$	28.768
11(50)	43.005	9.0337	$5.00E+05$	31.043
12(10)	242.55	562.16	**60480**	39101
12(30)	**2479.8**	3643.6	$3.00E+05$	19.931
12(50)	**17120**	14039	$5.00E+05$	9.1287
13(10)	0.46067	0.19216	$1.00E+05$	19.931
13(30)	2.288	1.4814	$3.00E+05$	0
13(50)	4.257	0.8859	$5.00E+05$	0
14(10)	3.6253	0.35519	$1.00E+05$	0
14(30)	13.106	0.47118	$3.00E+05$	0
14(50)	22.631	0.39482	$5.00E+05$	31.36
15(10)	182.33	193.2	**74777**	32918
15(30)	327.12	102.2	$3.00E+05$	26.801
15(50)	273.54	94.572	$5.00E+05$	29.642
16(10)	134.35	23.174	$1.00E+05$	0
16(30)	227.38	161.47	$3.00E+05$	45.758
16(50)	116.97	138.04	$5.00E+05$	21.804
17(10)	**126.32**	19.389	$1.00E+05$	0
17(30)	**173.3**	154.54	$3.00E+05$	0
17(50)	**153.15**	132.86	$5.00E+05$	0

Table B.8: Results of the investigation into alternative solution space diversity control mechanisms: DIVHH (constant species selection) and ADIVHH (adaptive species selection).

Prob (Dims)	DIVHH				ADIVHH			
	FFV		# FEs		FFV		# FEs	
	μ	σ	μ	σ	μ	σ	μ	σ
1(10)	$1.00E-06$	0	**8756.7**	469.54	$1.00E-06$	0	8876.7	472.47
1(30)	$1.00E-06$	0	**30677**	1471.7	$1.00E-06$	0	30810	1859.1
1(50)	$1.00E-06$	0	62497	2540.5	$1.00E-06$	0	**61763**	2742.8
2(10)	$1.00E-06$	0	**12737**	728	$1.00E-06$	0	12777	914.89
2(30)	$1.00E-06$	0	96830	9455.1	$1.00E-06$	0	95827	8084.9
2(50)	$1.00E-06$	0	$3.0483E+05$	21347	$1.00E-06$	0	$3.0276E+05$	20046
3(10)	$1.00E-06$	0	22370	2558.7	$1.00E-06$	0	**22037**	2379.6
3(30)	15709	17425	$3.00E+05$	0	14322	11644	$3.00E+05$	0
3(50)	**78276**	38647	$5.00E+05$	0	83440	43728	$5.00E+05$	0
4(10)	$1.00E-06$	0	**13650**	871.29	$1.00E-06$	0	13783	1212.3
4(30)	$1.00E-06$	0	$\mathbf{1.3636E+05}$	16722	$1.00E-06$	0	$1.3743E+05$	34916
4(50)	741.16	1837.8	$5.00E+05$	0	110.87	305.21	$5.00E+05$	0
5(10)	$1.00E-06$	0	12513	1433.8	$1.00E-06$	0	**12047**	898.94
5(30)	572.37	670.35	$3.00E+05$	0	402.87	477.12	$3.00E+05$	0
5(50)	3585.9	654.71	$5.00E+05$	0	3565.9	831.91	$5.00E+05$	0
6(10)	0.00033333	0.0018257	**29677**	11907	0.002	0.0040684	32763	14187
6(30)	1.1987	1.8617	$2.3845E+05$	53605	0.98833	1.8469	$\mathbf{2.2272E+05}$	65776
6(50)	6.2293	5.6979	$\mathbf{4.7594E+05}$	57513	7.933	14.953	$4.7642E+05$	52760
7(10)	0.165	0.09666	$1.00E+05$	0	0.19267	0.16737	$1.00E+05$	0
7(30)	0.0083333	0.014162	$1.5818E+05$	$1.2622E+05$	0.010333	0.014016	$1.7338E+05$	$1.2886E+05$
7(50)	0.00066667	0.0025371	$\mathbf{1.7877E+05}$	$1.4677E+05$	0.0046667	0.010417	$2.0712E+05$	$1.6519E+05$
8(10)	20.051	0.097129	$1.00E+05$	24.598	**20.05**	0.10417	$1.00E+05$	19.205
8(30)	20.208	0.15028	$3.00E+05$	13.817	20.221	0.1364	$3.00E+05$	20.833
8(50)	21.077	0.13417	$5.00E+05$	0	20.805	0.40291	$5.00E+05$	0
9(10)	0.006	0.0049827	**37743**	15066	0.007	0.0046609	39643	16243
9(30)	2.1807	1.9921	$2.8907E+05$	32679	2.478	1.9473	$2.9427E+05$	19808
9(50)	18.333	12.165	$4.9994E+05$	310.38	20.999	11.252	$5.00E+05$	0
10(10)	13.469	5.5986	$1.00E+05$	0	**12.928**	6.7108	$1.00E+05$	0
10(30)	83.46	27.429	$3.00E+05$	0	71.604	24.588	$3.00E+05$	1.8257
10(50)	210.67	54.125	$5.00E+05$	10.148	186.37	39.198	$5.00E+05$	19.491
11(10)	6.0496	2.0357	$1.00E+05$	10.283	5.8025	1.5892	$1.00E+05$	4.3417
11(30)	30.394	3.3909	$3.00E+05$	33.192	29.559	3.6047	$3.00E+05$	27.926
11(50)	59.229	3.9607	$5.00E+05$	31.55	58.916	5.47	$5.00E+05$	24.962
12(10)	191.16	510.98	**53063**	39042	279.44	548.06	59723	40319
12(30)	4211.1	4194.2	$3.00E+05$	8.9763	2984	4433.5	$2.9576E+05$	23260
12(50)	28073	24371	$5.00E+05$	0	30380	18795	$5.00E+05$	0
13(10)	**0.40467**	0.14994	$1.00E+05$	0	0.51333	0.14182	$1.00E+05$	0
13(30)	5.0677	1.5478	$3.00E+05$	0	5.0143	1.9442	$3.00E+05$	0
13(50)	21.222	7.227	$5.00E+05$	0	19.962	5.7001	$5.00E+05$	0
14(10)	**3.5637**	0.33974	$1.00E+05$	0	3.6447	0.42816	$1.00E+05$	0
14(30)	**13.083**	0.45656	$3.00E+05$	0	13.147	0.32714	$3.00E+05$	0
14(50)	**21.731**	0.92441	$5.00E+05$	0	22.135	0.91735	$5.00E+05$	0
15(10)	237.03	198.35	81013	29724	259.89	196.18	85629	28758
15(30)	**313.45**	134.77	$3.00E+05$	11.043	360.93	97.258	$3.00E+05$	17.499
15(50)	286.76	95.286	$5.00E+05$	3.6515	330.69	98.527	$5.00E+05$	3.6515
16(10)	**125.53**	17.529	$1.00E+05$	31.675	126.41	21.736	$1.00E+05$	30.926
16(30)	234.75	148.07	$3.00E+05$	30.859	163.57	123.48	$3.00E+05$	35.207
16(50)	244.59	141.1	$5.00E+05$	27.916	238.26	123.21	$5.00E+05$	34.032
17(10)	135.59	19.435	$1.00E+05$	0	**132.76**	25.888	$1.00E+05$	0
17(30)	221.92	163.39	$3.00E+05$	0	**195.5**	139.97	$3.00E+05$	0
17(50)	177.52	106.45	$5.00E+05$	0	191.53	118.35	$5.00E+05$	0

Table B.9: Results of the investigation into alternative heuristic space diversity control mechanisms: Baseline HMHH algorithm (no heuristic space diversity control strategy).

Prob (Dims)	HMHH (baseline)			
	FFV		# FEs	
	μ	σ	μ	σ
1(10)	$1.00E-06$	0	13310	694.98
1(30)	$1.00E-06$	0	45040	1783.8
1(50)	$1.00E-06$	0	**74587**	2362.3
2(10)	$1.00E-06$	0	31577	2659
2(30)	$1.00E-06$	0	$2.2039E+05$	49695
2(50)	0.0013341	0.0043412	$4.9119E+05$	17885
3(10)	238.87	910.04	84027	11776
3(30)	$1.3519E+05$	73374	$3.00E+05$	0
3(50)	$4.1224E+05$	$1.9264E+05$	$5.00E+05$	0
4(10)	$1.00E-06$	0	43940	9759.9
4(30)	0.94533	2.8539	$3.00E+05$	0
4(50)	340.53	332.25	$5.00E+05$	0
5(10)	$1.00E-06$	0	17980	1572
5(30)	1360.6	728.2	$3.00E+05$	0
5(50)	4906	1030.8	$5.00E+05$	0
6(10)	0.119	0.41785	65910	19543
6(30)	4.2753	22.125	$2.5104E+05$	45175
6(50)	8.0497	21.09	$4.7265E+05$	45948
7(10)	0.44533	0.34199	$1.00E+05$	0
7(30)	0.0046667	0.0081931	$1.6146E+05$	$1.153E+05$
7(50)	0.003	0.0059596	$2.3284E+05$	$1.7801E+05$
8(10)	20.061	0.10594	$1.00E+05$	0
8(30)	20.166	0.10969	$3.00E+05$	0
8(50)	**20.262**	0.099158	$5.00E+05$	0
9(10)	0.038	0.17798	38270	19385
9(30)	2.245	1.8063	$2.8956E+05$	30956
9(50)	14.237	4.7209	$5.00E+05$	0
10(10)	16.157	7.0261	$1.00E+05$	0
10(30)	72.135	30.203	$3.00E+05$	0
10(50)	89.756	21.106	$5.00E+05$	0
11(10)	6.8758	1.7015	$1.00E+05$	0
11(30)	27.311	4.4467	$3.00E+05$	0
11(50)	52.332	6.9296	$5.00E+05$	0
12(10)	466.17	620.43	88740	19820
12(30)	4489.4	5300.9	$3.00E+05$	0
12(50)	60290	54326	$5.00E+05$	0
13(10)	0.50233	0.22999	$1.00E+05$	0
13(30)	**1.8733**	0.441	$3.00E+05$	0
13(50)	**3.8647**	1.1894	$5.00E+05$	0
14(10)	3.653	0.2848	$1.00E+05$	0
14(30)	13.14	0.44261	$3.00E+05$	0
14(50)	22.517	0.55336	$5.00E+05$	0
15(10)	263.91	212.3	**76663**	33922
15(30)	347.11	97.654	$3.00E+05$	0
15(50)	302.11	109.61	$5.00E+05$	0
16(10)	137.61	25.198	$1.00E+05$	0
16(30)	195.48	138.18	$3.00E+05$	0
16(50)	159.99	128.26	$5.00E+05$	0
17(10)	134.52	17.935	$1.00E+05$	0
17(30)	**163.15**	110.67	$3.00E+05$	0
17(50)	130.69	108.95	$5.00E+05$	0

Table B.10: Results of the investigation into alternative heuristic space diversity control mechanisms: LDHH (linearly decreasing heuristic space diversity) and EDHH (exponentially decreasing heuristic space diversity).

Prob (Dims)	LDHH				EDHH			
	FFV		# FEs		FFV		# FEs	
	μ	σ	μ	σ	μ	σ	μ	σ
1(10)	$1.00E-06$	0	**12017**	642.78	$1.00E-06$	0	12090	579.15
1(30)	$1.00E-06$	0	**43653**	1500.7	$1.00E-06$	0	46723	2038.7
1(50)	$1.00E-06$	0	84380	9942.9	$1.00E-06$	0	77327	4280.2
2(10)	$1.00E-06$	0	**13743**	959.41	$1.00E-06$	0	14393	902.84
2(30)	$1.00E-06$	0	**97973**	14510	$1.00E-06$	0	$2.0176E+05$	95248
2(50)	0.0030009	0.016431	$3.7491E+05$	$1.6617E+05$	0.55267	1.3763	$4.11E+05$	$1.6417E+05$
3(10)	$1.00E-06$	0	20120	2020.8	$1.00E-06$	0	**19550**	1444.1
3(30)	55462	48441	$2.9701E+05$	16395	70567	75788	$3.00E+05$	0
3(50)	$2.1491E+05$	$1.4672E+05$	$5.00E+05$	0	$5.266E+05$	$3.2535E+05$	$5.00E+05$	0
4(10)	$1.00E-06$	0	**15550**	1344.9	$1.00E-06$	0	16003	1023.7
4(30)	0.0033343	0.018257	$1.502E+05$	$1.0167E+05$	1.1467	3.7805	$1.6407E+05$	$1.0009E+05$
4(50)	37.163	181.92	$2.837E+05$	$1.7079E+05$	295.84	1291.7	$\mathbf{2.6099E+05}$	$1.4466E+05$
5(10)	$1.00E-06$	0	17313	1044.1	$1.00E-06$	0	**17077**	1054
5(30)	1086.8	849.77	$3.00E+05$	0	1200.2	662.45	$3.00E+05$	0
5(50)	5052.4	1274.2	$5.00E+05$	0	5238.2	1421.9	$5.00E+05$	0
6(10)	0.26533	1.0098	37347	18132	0.016	0.083896	**35933**	13900
6(30)	1.1653	1.9058	$2.3937E+05$	54813	1.076	1.7304	$2.7588E+05$	41820
6(50)	21.662	26.963	$4.8019E+05$	53900	35.225	49.587	$5.00E+05$	0
7(10)	0.15467	0.12678	$1.00E+05$	0	0.106	0.093055	93167	17726
7(30)	0.004	0.010372	$1.4953E+05$	$1.1659E+05$	0.0066667	0.01561	$1.4694E+05$	$1.1871E+05$
7(50)	0.0016667	0.0064772	$\mathbf{1.8404E+05}$	$1.4432E+05$	0.0056667	0.011351	$2.1627E+05$	$1.7421E+05$
8(10)	**20.034**	0.077442	99290	3888.8	20.114	0.16596	$1.00E+05$	0
8(30)	**20.15**	0.13011	$3.00E+05$	0	20.239	0.1972	$3.00E+05$	0
8(50)	20.622	0.34904	$5.00E+05$	0	20.829	0.3542	$5.00E+05$	0
9(10)	0.0046667	0.0050742	**33440**	15441	0.52933	0.88824	54163	35998
9(30)	9.8607	5.3668	$2.9727E+05$	14953	18.916	8.0898	$3.00E+05$	0
9(50)	29.842	16.353	$5.00E+05$	0	48.514	19.672	$5.00E+05$	0
10(10)	19.136	9.4447	$1.00E+05$	0	15.582	9.6233	98393	8800.1
10(30)	73.098	31.88	$3.00E+05$	0	73.336	28.963	$3.00E+05$	0
10(50)	40.668	16.476	$5.00E+05$	0	**29.142**	20.344	$5.00E+05$	0
11(10)	6.9141	1.3129	$1.00E+05$	0	6.1124	2.0343	98277	9439.1
11(30)	21.74	7.7398	$3.00E+05$	0	19.711	5.2357	$3.00E+05$	0
11(50)	49.067	9.0971	$5.00E+05$	0	**43.424**	13.12	$5.00E+05$	0
12(10)	156.24	420.61	69417	35661	131.04	401.7	**60357**	37188
12(30)	4646.4	4889	$3.00E+05$	0	10348	13200	$3.00E+05$	0
12(50)	43832	26157	$5.00E+05$	0	56752	44038	$5.00E+05$	0
13(10)	0.54667	0.19359	$1.00E+05$	0	0.585	0.22664	$1.00E+05$	0
13(30)	2.7937	0.87927	$3.00E+05$	0	2.2563	0.63171	$3.00E+05$	0
13(50)	4.9403	2.2133	$5.00E+05$	0	4.794	3.0517	$5.00E+05$	0
14(10)	3.457	0.54519	$1.00E+05$	0	3.5107	0.42496	$1.00E+05$	0
14(30)	13.204	0.31822	$3.00E+05$	0	13.185	0.38084	$3.00E+05$	0
14(50)	22.283	0.91453	$5.00E+05$	0	22.524	1.0094	$5.00E+05$	0
15(10)	222.08	194.97	79050	33379	251.95	179.67	91060	23362
15(30)	340.26	89.804	$3.00E+05$	0	330.33	102.54	$3.00E+05$	0
15(50)	297.23	99.908	$5.00E+05$	0	307.76	101.03	$5.00E+05$	0
16(10)	135.35	27.021	94230	6630.6	134.17	26.114	78710	20528
16(30)	161.8	103.94	$3.00E+05$	0	189.7	154.73	$3.00E+05$	0
16(50)	122.27	118.27	$5.00E+05$	0	**102.04**	125.19	$4.9486E+05$	13023
17(10)	137.86	18.663	$1.00E+05$	0	137.21	25.201	$1.00E+05$	0
17(30)	198.1	160.95	$3.00E+05$	0	207.01	180.66	$3.00E+05$	0
17(50)	169.22	159.14	$5.00E+05$	0	**120.22**	129.83	$5.00E+05$	0

Table B.11: Results of the investigation into alternative heuristic space diversity control mechanisms: LIHH1 (linear increasing heuristic space diversity with *a priori* knowledge) and EIHH1 (exponentially increasing heuristic space diversity with *a priori* knowledge).

Prob (Dims)	LIHH1				EIHH1			
	FFV		# FEs		FFV		# FEs	
	μ	σ	μ	σ	μ	σ	μ	σ
1(10)	$1.00E-06$	0	8623.3	244.5	$1.00E-06$	0	**8563.3**	277.28
1(30)	$1.00E-06$	0	19253	599.27	$1.00E-06$	0	**19193**	549.57
1(50)	$1.00E-06$	0	26793	739.96	$1.00E-06$	0	**26753**	670.94
2(10)	$1.00E-06$	0	9173.3	294.7	$1.00E-06$	0	**9106.7**	398.21
2(30)	$1.00E-06$	0	26760	766.36	$1.00E-06$	0	**26410**	806.59
2(50)	$1.00E-06$	0	**52543**	736.57	$1.00E-06$	0	52723	903.89
3(10)	$1.00E-06$	0	13350	483.34	$1.00E-06$	0	**13277**	397.13
3(30)	$1.00E-06$	0	**61567**	1265.3	$1.00E-06$	0	61603	1201.9
3(50)	$1.00E-06$	0	$\mathbf{1.5574E+05}$	2671.5	$1.00E-06$	0	$1.6336e+05$	6549
4(10)	$1.00E-06$	0	**9510**	347.75	$1.00E-06$	0	9573.3	374.1
4(30)	$1.00E-06$	0	29330	927.79	$1.00E-06$	0	**29267**	986.58
4(50)	$1.00E-06$	0	59993	1287.6	$1.00E-06$	0	**59840**	1043.1
5(10)	$1.00E-06$	0	17520	608.79	$1.00E-06$	0	**17447**	567.35
5(30)	$1.00E-06$	0	$\mathbf{2.5721e+05}$	68420	0.00033427	0.0018254	$2.7194e+05$	70931
5(50)	**0.039999**	0.059073	$5.00E+05$	0	190.58	730.17	$5.00E+05$	0
6(10)	0.00066667	0.0025371	**18940**	744.91	0.00033333	0.0018257	19210	716
6(30)	0.133	0.72658	$\mathbf{1.2291e+05}$	46644	0	0	$2.0872e+05$	49555
6(50)	0.0013333	0.0057135	$\mathbf{3.7927e+05}$	93024	3.8803	12.716	$4.741e+05$	37361
7(10)	0.001	0.0030513	**7533.3**	339.71	0.00066667	0.0025371	7640	439.91
7(30)	0.00033333	0.0018257	**16603**	673.38	0	0	16657	622.94
7(50)	0	0	**24980**	703.39	0	0	25013	676.57
8(10)	20.07	0.11418	$1.00E+05$	0	20.054	0.096654	$1.00E+05$	0
8(30)	20.209	0.15768	$3.00E+05$	0	**20.122**	0.10621	$3.00E+05$	0
8(50)	21.102	0.14223	$5.00E+05$	0	21.123	0.0312	$5.00E+05$	0
9(10)	0.495	0.76553	79563	30822	0.233	0.41917	58827	27886
9(30)	7.8523	4.5941	$3.00E+05$	0	3.4547	3.7103	$2.7578e+05$	42827
9(50)	24.662	7.7736	$5.00E+05$	0	21.058	6.896	$5.00E+05$	0
10(10)	1.6147	1.0913	**88090**	30887	1.8793	1.2046	88200	30602
10(30)	12.719	5.4544	$3.00E+05$	0	**10.553**	4.5767	$3.00E+05$	0
10(50)	28.78	10.823	$5.00E+05$	0	**26.417**	10.946	$5.00E+05$	0
11(10)	1.223	1.2786	74537	39566	1.0214	1.149	**66540**	41709
11(30)	**8.9018**	3.2821	$3.00E+05$	0	10.785	6.4151	$3.00E+05$	0
11(50)	**19.799**	4.1018	$5.00E+05$	0	21.232	7.9897	$5.00E+05$	0
12(10)	174.03	416.19	72990	41976	317.66	627.11	63770	45137
12(30)	5850.1	3788.7	$2.9121e+05$	48127	4822.2	4782.6	$3.00E+05$	0
12(50)	26343	16432	$5.00E+05$	0	35970	22535	$5.00E+05$	0
13(10)	0.643	0.21991	$1.00E+05$	0	0.44167	0.14283	$1.00E+05$	0
13(30)	2.4147	0.67678	$3.00E+05$	0	**1.8363**	0.55461	$3.00E+05$	0
13(50)	4.597	0.67692	$5.00E+05$	0	**4.141**	0.82065	$5.00E+05$	0
14(10)	2.8263	0.6434	$1.00E+05$	0	**2.6953**	0.37906	$1.00E+05$	0
14(30)	**10.059**	0.82305	$3.00E+05$	0	10.288	0.96526	$3.00E+05$	0
14(50)	**19.482**	0.64356	$5.00E+05$	0	19.92	0.74443	$5.00E+05$	0
15(10)	340.36	102.85	$1.00E+05$	0	366.66	88.409	$1.00E+05$	0
15(30)	**276.66**	89.763	$3.00E+05$	0	**276.66**	104	$3.00E+05$	0
15(50)	**253.35**	93.695	$5.00E+05$	0	256.66	77.386	$5.00E+05$	0
16(10)	**100.73**	12.229	94410	17096	100.8	13.241	$1.00E+05$	0
16(30)	179.42	167.61	$3.00E+05$	0	**120.7**	142.49	$3.00E+05$	0
16(50)	**129.03**	143.34	$5.00E+05$	0	148.7	163.59	$5.00E+05$	0
17(10)	97.72	8.1602	$1.00E+05$	0	**96.525**	7.7402	$1.00E+05$	0
17(30)	183.73	205.24	$3.00E+05$	0	190.59	177.28	$3.00E+05$	0
17(50)	143.73	142.08	$5.00E+05$	0	**78.116**	78.053	$5.00E+05$	0

Table B.12: Results of the investigation into alternative heuristic space diversity control mechanisms: LIHH2 (linear increasing heuristic space diversity without *a priori* knowledge) and EIHH2 (exponentially increasing heuristic space diversity without *a priori* knowledge).

Prob (Dims)	LIHH2				EIHH2			
	FFV		# FEs		FFV		# FEs	
	μ	σ	μ	σ	μ	σ	μ	σ
1(10)	$1,00E-06$	0	18870	8668.3	$1,00E-06$	0	16610	8578.8
1(30)	$1,00E-06$	0	60323	30956	$1,00E-06$	0	49367	28510
1(50)	$1,00E-06$	0	$1.053e+05$	62707	$1,00E-06$	0	89440	55856
2(10)	$1,00E-06$	0	32310	21104	$1,00E-06$	0	28097	12646
2(30)	$1,00E-06$	0	$1.7347e+05$	85128	$1,00E-06$	0	$1.0933e+05$	72461
2(50)	0.0090008	0.031985	$3.4577e+05$	$1.8287e+05$	$1,00E-06$	0	$\mathbf{2.9271e+05}$	$1.5095e+05$
3(10)	3.7033	17.172	57867	37168	$1,00E-06$	0	43277	19645
3(30)	51869	86498	$2.1972e+05$	$1.0788e+05$	6796	17006	$\mathbf{1.6836e+05}$	$1.0592e+05$
3(50)	$2.1453e+05$	$2.7217e+05$	$\mathbf{3.975e+05}$	$1.5265e+05$	99777	$1.0943e+05$	$4.2713e+05$	$1.2437e+05$
4(10)	$1,00E-06$	0	39220	23734	$1,00E-06$	0	29923	13977
4(30)	$1,00E-06$	0	$1.4722e+05$	97022	$1,00E-06$	0	$\mathbf{1.3888e+05}$	73296
4(50)	344.97	930.78	$3.9416e+05$	$1.6501e+05$	0.578	2.5913	$2.9951e+05$	$1.712e+05$
5(10)	$1,00E-06$	0	46650	20429	$1,00E-06$	0	31960	7965
5(30)	1064.4	1734.8	$2.8774e+05$	34956	**670.92**	905.42	$2.8844e+05$	45133
5(50)	3908.7	2977.8	$5,00E+05$	0	**3395.5**	2249.4	$4.605e+05$	$1.0539e+05$
6(10)	0.25133	0.95807	57533	24690	0.00066667	0.0025371	41690	18418
6(30)	1.8913	5.2325	$\mathbf{2.2451e+05}$	77133	0.901	2.9717	$2.3416e+05$	52399
6(50)	6.1297	8.7732	$4.6432e+05$	76351	5.24	7.869	$\mathbf{4.4701e+05}$	80799
7(10)	0.67867	1.1493	72220	43161	0.61767	1.0662	**67773**	40968
7(30)	0.0043333	0.0089763	$\mathbf{1.2978e+05}$	$1.1301e+05$	0.0083333	0.012888	$1.5247e+05$	$1.1976e+05$
7(50)	0.0023333	0.0067891	$2.3375e+05$	$1.7057e+05$	0.003	0.0079438	$2.0818e+05$	$1.9341e+05$
8(10)	20.057	0.10968	$1,00E+05$	0	**20.034**	0.079333	$1,00E+05$	0
8(30)	20.226	0.27014	$3,00E+05$	0	20.235	0.26664	$3,00E+05$	0
8(50)	20.463	0.38992	$5,00E+05$	0	20.558	0.36105	$5,00E+05$	0
9(10)	0.067333	0.24812	46027	28073	0.038	0.17798	39850	20348
9(30)	3.4097	3.542	$\mathbf{2.6593e+05}$	68493	2.446	1.9105	$2.8476e+05$	37453
9(50)	**13.851**	8.9144	$4.5523e+05$	$1.1611e+05$	16.123	14.872	$4.8928e+05$	58698
10(10)	17.456	15.677	97077	16012	**14.653**	17.797	$1,00E+05$	0
10(30)	**52.991**	38.962	$3,00E+05$	0	57.979	48.81	$3,00E+05$	0
10(50)	110.71	60.034	$5,00E+05$	0	110.28	62.968	$5,00E+05$	0
11(10)	3.8435	2.491	91433	23930	3.7648	2.8672	**89520**	27470
11(30)	**19.651**	9.1953	$3,00E+05$	0	24.223	9.253	$3,00E+05$	0
11(50)	44.635	14.737	$5,00E+05$	0	43.44	18.5	$5,00E+05$	0
12(10)	254.83	564.94	73370	32129	220.08	561.14	63337	29948
12(30)	5033.1	5266.4	$3,00E+05$	0	**3296.3**	5065	$2.9662e+05$	13275
12(50)	**25332**	17809	$5,00E+05$	0	26310	27152	$5,00E+05$	0
13(10)	**0.43733**	0.15909	$1,00E+05$	0	0.50733	0.16607	$1,00E+05$	0
13(30)	2.216	0.90388	$3,00E+05$	0	2.0943	0.48971	$3,00E+05$	0
13(50)	4.7787	2.0317	$5,00E+05$	0	6.6917	4.4423	$5,00E+05$	0
14(10)	**3.1147**	0.6036	$1,00E+05$	0	3.1803	0.50128	$1,00E+05$	0
14(30)	**12.096**	1.2419	$3,00E+05$	0	12.492	0.99427	$3,00E+05$	0
14(50)	**21.387**	1.7448	$5,00E+05$	0	21.534	1.2894	$5,00E+05$	0
15(10)	225.03	195.33	83517	27458	230.52	221.47	78130	27953
15(30)	307.57	116.67	$3,00E+05$	0	**284.64**	126.64	$2.9793e+05$	11338
15(50)	298.08	97.404	$5,00E+05$	0	**283.46**	95.071	$5,00E+05$	0
16(10)	125.43	26.513	$1,00E+05$	0	**121.64**	24.171	$1,00E+05$	0
16(30)	**126.52**	115.06	$3,00E+05$	0	181.14	161.42	$3,00E+05$	0
16(50)	171.22	173.72	$5,00E+05$	0	156.36	107.54	$5,00E+05$	0
17(10)	120.71	22.704	$1,00E+05$	0	**119.64**	24.8	$1,00E+05$	0
17(30)	169.03	163.13	$3,00E+05$	0	228.75	178.19	$3,00E+05$	0
17(50)	226.73	138.82	$5,00E+05$	0	150.06	130.81	$5,00E+05$	0

Table B.13: Results comparison: HMHH and EIHH1.

Prob (Dims)	HMHH				EIHH1			
	FFV		# FEs		FFV		# FEs	
	μ	σ	μ	σ	μ	σ	μ	σ
1(10)	$1.00E-06$	0	11870	538.93	$1.00E-06$	0	**8563.3**	277.28
1(30)	$1.00E-06$	0	52983	2723.3	$1.00E-06$	0	**19193**	549.57
1(50)	$1.00E-06$	0	$1.16E+05$	9985.8	$1.00E-06$	0	**26753**	670.94
2(10)	$1.00E-06$	0	14013	1246.7	$1.00E-06$	0	**9106.7**	398.21
2(30)	$1.00E-06$	0	90727	15876	$1.00E-06$	0	**26410**	806.59
2(50)	$1.00E-06$	0	$2.45E+05$	78036	$1.00E-06$	0	**52723**	903.89
3(10)	$1.00E-06$	0	22750	3003	$1.00E-06$	0	**13277**	397.13
3(30)	295.35	998.74	$2.79E+05$	27782	$1.00E-06$	0	**61603**	1201.9
3(50)	78960	71769	$5.00E+05$	0	$1.00E-06$	0	$\mathbf{1.63E+05}$	6549
4(10)	$1.00E-06$	0	15853	1341.8	$1.00E-06$	0	**9573.3**	374.1
4(30)	$1.00E-06$	0	$1.50E+05$	27962	$1.00E-06$	0	**29267**	986.58
4(50)	1.193	5.5772	$4.66E+05$	60163	$1.00E-06$	0	**59840**	1043.1
5(10)	$1.00E-06$	0	**17110**	883.31	$1.00E-06$	0	17447	567.35
5(30)	174.5	278.17	$3.00E+05$	0	0.00033427	0.0018254	$\mathbf{2.72E+05}$	70931
5(50)	3237	760.19	$5.00E+05$	0	**190.58**	730.17	$5.00E+05$	0
6(10)	0	0	33417	7388.8	0.00033333	0.0018257	**19210**	716
6(30)	0.13267	0.72665	$2.43E+05$	33935	0	0	$\mathbf{2.09E+05}$	49555
6(50)	4.8457	13.791	$4.78E+05$	35158	3.8803	12.716	$\mathbf{4.74E+05}$	37361
7(10)	0.162	0.13632	$1.00E+05$	0	0.00066667	0.0025371	**7640**	439.91
7(30)	0.0036667	0.0080872	$1.30E+05$	$1.13E+05$	0	0	**16657**	622.94
7(50)	0.0033333	0.0095893	$2.03E+05$	$1.67E+05$	0	0	**25013**	676.57
8(10)	20.06	0.086681	$1.00E+05$	0	20.054	0.096654	$1.00E+05$	0
8(30)	20.169	0.12041	$3.00E+05$	0	**20.122**	0.10621	$3.00E+05$	0
8(50)	**20.796**	0.37683	$5.00E+05$	0	21.123	0.0312	$5.00E+05$	0
9(10)	0.005	0.0050855	**43320**	19202	0.233	0.41917	58827	27886
9(30)	2.3763	1.3964	$2.98E+05$	10079	3.4547	3.7103	$2.76E+05$	42827
9(50)	14.373	3.3496	$5.00E+05$	0	21.058	6.896	$5.00E+05$	0
10(10)	15.556	9.1746	$1.00E+05$	0	1.8793	1.2046	**88200**	30602
10(30)	55.768	19.838	$3.00E+05$	0	**10.553**	4.5767	$3.00E+05$	0
10(50)	57.474	48.572	$5.00E+05$	0	**26.417**	10.946	$5.00E+05$	0
11(10)	5.0464	2.2525	97283	14880	1.0214	1.149	**66540**	41709
11(30)	24.378	5.1224	$3.00E+05$	0	**10.785**	6.4151	$3.00E+05$	0
11(50)	44.831	9.4654	$5.00E+05$	0	**21.232**	7.9897	$5.00E+05$	0
12(10)	302.86	546.06	69543	39317	317.66	627.11	**63770**	45137
12(30)	2611.5	3622.3	$2.95E+05$	20162	4822.2	4782.6	$3.00E+05$	0
12(50)	21252	14310	$5.00E+05$	0	35970	22535	$5.00E+05$	0
13(10)	**0.43267**	0.16282	99050	5203.4	0.44167	0.14283	$1.00E+05$	0
13(30)	2.0187	0.44262	$3.00E+05$	0	1.8363	0.55461	$3.00E+05$	0
13(50)	4.1393	0.92703	$5.00E+05$	0	4.141	0.82065	$5.00E+05$	0
14(10)	3.6397	0.29122	$1.00E+05$	0	**2.6953**	0.37906	$1.00E+05$	0
14(30)	13.14	0.44011	$3.00E+05$	0	**10.288**	0.96526	$3.00E+05$	0
14(50)	22.527	0.3707	$5.00E+05$	0	**19.92**	0.74443	$5.00E+05$	0
15(10)	258.98	210.57	**80170**	30406	366.66	88.409	$1.00E+05$	0
15(30)	373.72	108.43	$3.00E+05$	0	276.66	104	$3.00E+05$	0
15(50)	259.17	114.32	$5.00E+05$	0	256.66	77.386	$5.00E+05$	0
16(10)	138.39	25.216	$1.00E+05$	0	**100.8**	13.241	$1.00E+05$	0
16(30)	153.27	128.88	$3.00E+05$	0	120.7	142.49	$3.00E+05$	0
16(50)	**74.477**	45.497	$5.00E+05$	0	148.7	163.59	$5.00E+05$	0
17(10)	133.28	22.828	$1.00E+05$	0	**96.525**	7.7402	$1.00E+05$	0
17(30)	200.71	169.72	$3.00E+05$	0	190.59	177.28	$3.00E+05$	0
17(50)	168.33	164.8	$5.00E+05$	0	**78.116**	78.053	$5.00E+05$	0

Table B.14: Results comparison: PAP and EEA-SLPS.

Prob (Dims)	PAP				EEA-SLPS			
	FFV		# FEs		FFV		# FEs	
	μ	σ	μ	σ	μ	σ	μ	σ
1(10)	$1.00E-06$	0	13857	630.64	$1.00E-06$	0	13923	517.74
1(30)	$1.00E-06$	0	39190	5945.1	$1.00E-06$	0	35297	2974.8
1(50)	$1.00E-06$	0	70543	12587	$1.00E-06$	0	61427	11642
2(10)	$1.00E-06$	0	18760	1030.4	$1.00E-06$	0	18553	858.12
2(30)	$1.00E-06$	0	90063	2729.3	$1.00E-06$	0	90507	2064.3
2(50)	$1.00E-06$	0	$2.12E+05$	2708	$1.00E-06$	0	$2.1192E+05$	5535.8
3(10)	$1.00E-06$	0	46067	2040.3	$1.00E-06$	0	45283	2295.7
3(30)	$1.00E-06$	0	$2.88E+05$	5481.5	$1.00E-06$	0	$2.8519E+05$	5400.7
3(50)	1184.3	1011.8	$5.00E+05$	0	900.69	732.05	$5.00E+05$	0
4(10)	$1.00E-06$	0	20483	1424.7	$1.00E-06$	0	19927	1128.9
4(30)	4448.8	5562.2	$2.87E+05$	39164	2161.6	3328.6	$2.7523E+05$	54905
4(50)	23718	14259	$5.00E+05$	0	20340	13581	$5.00E+05$	0
5(10)	$1.00E-06$	0	25010	2790.5	$1.00E-06$	0	32470	7258.9
5(30)	1115.5	1446.5	$3.00E+05$	0	894.25	777.66	$3.00E+05$	0
5(50)	4779.2	1791.2	$5.00E+05$	0	5151.4	1333.9	$5.00E+05$	0
6(10)	0.13267	0.72665	50613	10870	0	0	49337	8334.1
6(30)	0.51867	1.3463	$2.75E+05$	27166	4.6447	13.185	$2.8726E+05$	30467
6(50)	17.502	17.011	$5.00E+05$	0	35.467	38.309	$5.00E+05$	0
7(10)	0.0026667	0.0058329	62867	35481	0.005	0.0073108	74563	36835
7(30)	0	0	82047	41811	0	0	$1.1778E+05$	93406
7(50)	0.0016667	0.0046113	$2.03E+05$	$1.67E+05$	0.001	0.0040258	$1.6185E+05$	$1.3644E+05$
8(10)	20.058	0.086279	$1.00E+05$	0	20.116	0.10344	$1.00E+05$	0
8(30)	20.264	0.16296	$3.00E+05$	0	20.281	0.1898	$3.00E+05$	0
8(50)	20.399	0.13771	$5.00E+05$	0	20.42	0.20265	$5.00E+05$	0
9(10)	0.22033	0.40067	68493	26929	0.069667	0.36084	45953	20620
9(30)	2.6757	1.92	$2.81E+05$	55032	1.255	1.6865	$\mathbf{2.0465E+05}$	92270
9(50)	7.615	5.9228	$5.00E+05$	0	6.4427	8.5585	$\mathbf{3.4528E+05}$	$1.6708E+05$
10(10)	12.857	6.2645	$1.00E+05$	0	8.8837	5.4366	$1.00E+05$	0
10(30)	70.555	19.764	$3.00E+05$	0	52.687	23.623	$3.00E+05$	0
10(50)	131.04	26.332	$5.00E+05$	0	102.69	44.344	$5.00E+05$	0
11(10)	4.085	1.2619	$1.00E+05$	0	4.1765	1.5439	$1.00E+05$	0
11(30)	20.034	2.6451	$3.00E+05$	0	20.661	2.6114	$3.00E+05$	0
11(50)	39.579	5.2391	$5.00E+05$	0	40.78	6.4723	$5.00E+05$	0
12(10)	231.53	539.16	71737	28064	92.191	330.6	69873	28012
12(30)	3573.5	3310.2	$3.00E+05$	0	7803.4	10925	$3.00E+05$	0
12(50)	30185	26195	$5.00E+05$	0	33888	28340	$5.00E+05$	0
13(10)	0.498	0.15873	$1.00E+05$	0	0.35533	0.15822	$1.00E+05$	0
13(30)	1.74	0.42099	$3.00E+05$	0	1.719	0.8862	$3.00E+05$	0
13(50)	3.2063	0.62491	$5.00E+05$	0	3.661	1.7035	$5.00E+05$	0
14(10)	3.4067	0.31705	$1.00E+05$	0	3.4057	0.32298	$1.00E+05$	0
14(30)	13.026	0.31914	$3.00E+05$	0	12.89	0.49124	$3.00E+05$	0
14(50)	22.832	0.36725	$5.00E+05$	0	22.604	0.36941	$5.00E+05$	0
15(10)	155.02	172.03	88220	20623	98.246	131.64	84703	23414
15(30)	261.27	126.22	$3.00E+05$	54.772	**257.13**	96.888	$3.00E+05$	0
15(50)	**247.57**	86.953	$5.00E+05$	0	254.84	81.626	$5.00E+05$	0
16(10)	126.8	21.726	$1.00E+05$	0	116.11	18.226	$1.00E+05$	0
16(30)	134.75	47.695	$3.00E+05$	0	**106.48**	66.794	$3.00E+05$	0
16(50)	155.75	60.738	$5.00E+05$	0	113.34	49.239	$5.00E+05$	0
17(10)	126.88	22.781	$1.00E+05$	0	115.59	9.4575	$1.00E+05$	0
17(30)	177.08	110.98	$3.00E+05$	0	146.88	107.95	$3.00E+05$	0
17(50)	246.31	88.772	$5.00E+05$	0	173.04	72.529	$5.00E+05$	0

Table B.15: Results comparison: Modified AMALGAM-SO and FAUC-Bandit.

Prob (Dims)	AMALGAM-SO				FAUC-Bandit			
	FFV		# FEs		FFV		# FEs	
	μ	σ	μ	σ	μ	σ	μ	σ
1(10)	$1.00E-06$	0	21259	35972	$1.00E-06$	0	38853	18096
1(30)	$1.00E-06$	0	33100	2680.3	$1.00E-06$	0	$2.1628E+05$	50495
1(50)	$1.00E-06$	0	83251	5861.2	$1.00E-06$	0	$3.9614E+05$	69337
2(10)	$1.00E-06$	0	91097	28241	0.062333	0.1303	100000	0
2(30)	0.016001	0.087635	$1.6153E+05$	$1.1561E+05$	68.307	52.263	300000	0
2(50)	10.23	55.595	$4.3093E+05$	$1.0929E+05$	1251.1	602.54	500000	0
3(10)	$3.62E+04$	81188	$1.00E+05$	182.09	$3.3135E+05$	$2.3001E+05$	100000	0
3(30)	$3.9749E+05$	$6.7835E+05$	$3.00E+05$	297.61	$2.1774E+06$	$9.4239E+05$	300000	0
3(50)	$3.5258E+05$	$1.9354E+05$	$5.00E+05$	411.09	$3.6405E+06$	$1.4462E+06$	$5.00E+05$	0
4(10)	0.062334	0.3339	91857	25903	0.20233	0.38718	100000	0
4(30)	28194	10394	$3.00E+05$	279.4	2703.2	1570.9	300000	0
4(50)	84661	17535	$5.00E+05$	348.5	27428	8516.5	$5.00E+05$	0
5(10)	$1.00E-06$	0	36660	42461	$1.00E-06$	0	93027	13710
5(30)	3639.9	1209.3	$3.00E+05$	296.09	4501.3	1003.9	$3.00E+05$	0
5(50)	8104.2	3136.2	$5.00E+05$	361.84	11270	1715.5	$5.00E+05$	0
6(10)	0.76167	1.7768	39415	34229	105.36	215.02	100000	0
6(30)	12.706	18.571	$2.7203E+05$	62809	217.44	291.33	300000	0
6(50)	34.991	30.985	$4.9458E+05$	24860	329.53	371.75	$5.00E+05$	0
7(10)	0.23267	0.16613	$1.00E+05$	201.88	1.1133	0.9626	100000	0
7(30)	0.010333	0.014967	$1.8984E+05$	$1.2875E+05$	366.07	433.63	$2.9825E+05$	9603.4
7(50)	0.006	0.0096847	$2.1356E+05$	$1.9159E+05$	964.66	561.56	500000	0
8(10)	20.263	0.11594	$1.00E+05$	192.14	20.337	0.076526	$1.00E+05$	0
8(30)	20.701	0.14497	$3.00E+05$	276.2	20.935	0.036175	$3.00E+05$	0
8(50)	20.924	0.21506	$5.00E+05$	399.66	21.135	0.03082	$5.00E+05$	0
9(10)	3.0123	2.5993	87887	32250	0.64233	0.95319	84387	19219
9(30)	2.746	2.7737	$2.4722E+05$	$1.0135E+05$	9.2517	4.682	300000	0
9(50)	3.6053	2.5711	$4.6126E+05$	$1.2053E+05$	22.294	4.1053	$5.00E+05$	0
10(10)	17.988	7.3616	$1.00E+05$	204.81	12.641	5.2662	$1.00E+05$	0
10(30)	99.106	27.234	$3.00E+05$	313.97	59.488	20.262	$3.00E+05$	0
10(50)	174.82	41.563	$5.00E+05$	415	129.28	36.435	$5.00E+05$	0
11(10)	5.9521	1.6896	$1.00E+05$	193.31	6.1848	1.99	$1.00E+05$	0
11(30)	31.173	2.7352	$3.00E+05$	285	36.107	3.3108	$3.00E+05$	0
11(50)	59.813	4.4452	$5.00E+05$	310.21	68.436	4.6194	$5.00E+05$	0
12(10)	469.89	814.31	69376	41505	682.42	733.23	98060	10626
12(30)	**2526.4**	3129.4	$3.00E+05$	280.71	4509.1	4094.9	$3.00E+05$	0
12(50)	27771	36755	$5.00E+05$	429.24	**20643**	13136	$5.00E+05$	0
13(10)	0.56433	0.42115	$1.00E+05$	183.96	0.97	0.41621	$1.00E+05$	0
13(30)	**1.447**	0.45196	$3.00E+05$	392.1	5.27	1.9156	$3.00E+05$	0
13(50)	**2.7203**	0.75874	$5.00E+05$	444.33	15.294	5.1243	$5.00E+05$	0
14(10)	3.4857	0.30168	$1.00E+05$	193.21	3.146	0.30093	$1.00E+05$	0
14(30)	13.11	0.26151	$3.00E+05$	261.17	13.197	0.28793	$3.00E+05$	0
14(50)	22.768	0.29265	$5.00E+05$	411.12	22.987	0.23939	$5.00E+05$	0
15(10)	269.09	164.92	96429	15148	255.25	191.1	95267	14253
15(30)	327.58	108.32	$3.00E+05$	259.61	308.85	136.03	300000	0
15(50)	306.85	108.05	$4.8781E+05$	69390	321.29	130.19	$5.00E+05$	0
16(10)	146.39	31.537	$1.00E+05$	191.28	120.82	14.091	$1.00E+05$	0
16(30)	271.9	147.42	$3.00E+05$	324.32	202.35	154.3	$3.00E+05$	0
16(50)	223.31	116.25	$5.00E+05$	423.03	156.62	104.93	$5.00E+05$	0
17(10)	151.56	26.789	$1.00E+05$	199.97	126.78	21.222	$1.00E+05$	0
17(30)	377.21	142.57	$3.00E+05$	297.68	216.95	174.71	$3.00E+05$	0
17(50)	381.6	81.206	$5.00E+05$	367.45	187.63	135.19	$5.00E+05$	0

Table B.16: Results comparison: CMAES and SaNSDE.

Prob (Dims)	CMAES				SaNSDE			
	FFV		# FEs		FFV		# FEs	
	μ	σ	μ	σ	μ	σ	μ	σ
1(10)	$1.00E-06$	0	**8526.7**	302.78	$1.00E-06$	0	20180	424.59
1(30)	$1.00E-06$	0	**19110**	447.48	$1.00E-06$	0	38973	839.51
1(50)	$1.00E-06$	0	26930	726.42	$1.00E-06$	0	54373	1255.6
2(10)	$1.00E-06$	0	9156.7	286.1	$1.00E-06$	0	38377	2088.3
2(30)	$1.00E-06$	0	26783	739.1	$1.00E-06$	0	$2.8926E+05$	19917
2(50)	$1.00E-06$	0	52903	869.2	19.154	31.42	$5.00E+05$	0
3(10)	$1.00E-06$	0	13320	379.11	$1.00E-06$	0	46337	2059.9
3(30)	$1.00E-06$	0	61173	1387.4	$1.7946E+05$	$1.1489E+05$	$3.00E+05$	0
3(50)	$1.00E-06$	0	$1.566E+05$	2244.2	$6.4456E+05$	$2.5505E+05$	$5.00E+05$	0
4(10)	$1.00E-06$	0	9590	283.27	$1.00E-06$	0	42767	2605.2
4(30)	$1.00E-06$	0	29357	570.35	195.19	186.51	$3.00E+05$	0
4(50)	$1.00E-06$	0	59607	998.25	9700.8	4681.3	$5.00E+05$	0
5(10)	$1.00E-06$	0	**17433**	546.04	$1.00E-06$	0	36280	1098.7
5(30)	$\mathbf{1.00E-06}$	0	$1.1465E+05$	3960.1	978.56	425.28	$3.00E+05$	0
5(50)	$\mathbf{1.00E-06}$	0	$3.4146E+05$	29588	4752.9	862.02	$5.00E+05$	0
6(10)	0.00066667	0.0025371	**18950**	744.52	0.26533	1.0098	52987	13213
6(30)	0.13267	0.72665	$\mathbf{1.2018E+05}$	37870	0.859	1.7187	$2.7619E+05$	33794
6(50)	**0.13267**	0.72665	$2.8902E+05$	51830	32.511	25.913	$5.00E+05$	0
7(10)	1267	$4.6252E-13$	$1.00E+05$	0	0.041667	0.025608	99920	438.18
7(30)	4696.3	$2.7751E-12$	$3.00E+05$	0	0.012	0.017889	$1.8329E+05$	$1.188E+05$
7(50)	6195.3	0	$5.00E+05$	0	0.0033333	0.0060648	$2.929E+05$	$1.9719E+05$
8(10)	20.312	0.11271	$1.00E+05$	0	20.345	0.077626	$1.00E+05$	0
8(30)	20.892	0.17459	$3.00E+05$	0	20.886	0.041728	$3.00E+05$	0
8(50)	21.127	0.030189	$5.00E+05$	0	21.066	0.03368	$5.00E+05$	0
9(10)	1.9457	1.5105	88203	30601	0	0	43690	2247.8
9(30)	39.564	6.4543	$3.00E+05$	0	0	0	$1.0608E+05$	4250.5
9(50)	62.344	7.3313	$5.00E+05$	0	**0**	0	$1.6323E+05$	11859
10(10)	**1.647**	1.1767	90947	27625	5.646	1.367	$1.00E+05$	0
10(30)	**9.391**	3.2817	$3.00E+05$	0	31.503	5.4713	$3.00E+05$	0
10(50)	**24.551**	7.5998	$5.00E+05$	0	67.351	11.503	$5.00E+05$	0
11(10)	1.2989	1.3647	30310	10496	5.3248	1.0486	$1.00E+05$	0
11(30)	**9.0255**	3.0546	$3.00E+05$	0	27.5	1.6478	$3.00E+05$	0
11(50)	**19.875**	5.2923	$5.00E+05$	0	54.258	2.6223	$5.00E+05$	0
12(10)	1546.1	2735.5	70053	43090	24.701	30.914	81150	23456
12(30)	20324	19261	$3.00E+05$	0	10594	2868.8	$3.00E+05$	0
12(50)	65826	69500	$5.00E+05$	0	**26251**	14685	$5.00E+05$	0
13(10)	0.897	0.25323	$1.00E+05$	0	**0.34233**	0.056366	$1.00E+05$	0
13(30)	3.179	0.56064	$3.00E+05$	0	**1.2977**	0.1177	$3.00E+05$	0
13(50)	5.3587	0.83373	$5.00E+05$	0	**2.3463**	0.17738	$5.00E+05$	0
14(10)	**2.5847**	0.53024	$1.00E+05$	0	3.2743	0.25292	$1.00E+05$	0
14(30)	10.394	0.8103	$3.00E+05$	0	12.707	0.23694	$3.00E+05$	0
14(50)	**19.45**	1.0963	$5.00E+05$	0	22.324	0.25815	$5.00E+05$	0
15(10)	343.32	97.143	$1.00E+05$	0	67.475	120.66	86313	17729
15(30)	**266.27**	64.911	$3.00E+05$	0	298.02	118.04	$3.00E+05$	0
15(50)	**245**	63.66	$5.00E+05$	0	256.6	80.671	$5.00E+05$	0
16(10)	**96.626**	9.3047	$1.00E+05$	0	102.06	5.2458	$1.00E+05$	0
16(30)	130.49	150.98	$3.00E+05$	0	**64.807**	29.134	$3.00E+05$	0
16(50)	98.632	126.2	$5.00E+05$	0	**78.543**	67.16	$5.00E+05$	0
17(10)	99.584	10.511	$1.00E+05$	0	117.61	10.118	$1.00E+05$	0
17(30)	175.66	179.47	$3.00E+05$	0	**104.17**	41.022	$3.00E+05$	0
17(50)	182.28	166.26	$5.00E+05$	0	125.65	68.354	$5.00E+05$	0

Table B.17: Results comparison: GCPSO and GA.

Prob (Dims)	GCPSO				GA			
	FFV		# FEs		FFV		# FEs	
	μ	σ	μ	σ	μ	σ	μ	σ
1(10)	138.4	6.8073	$1.00E+05$	0	$1.00E-06$	0	18557	1582.4
1(30)	231.81	29.491	$3.00E+05$	0	$1.00E-06$	0	99297	4860.4
1(50)	356.19	50.373	$5.00E+05$	0	$1.00E-06$	0	$4.1101E+05$	20554
2(10)	544.1	1.5915	$1.00E+05$	0	1.0873	1.54	$1.00E+05$	0
2(30)	567	3.0169	$3.00E+05$	0	446.67	228.69	$3.00E+05$	0
2(50)	595.87	4.663	$5.00E+05$	0	6142	2003.1	$5.00E+05$	0
3(10)	970.96	1411.8	99177	4509.6	$9.7613E+05$	$1.0468E+06$	$1.00E+05$	0
3(30)	10543	8705.2	$3.00E+05$	0	$5.9255E+06$	$2.6545E+06$	$3.00E+05$	0
3(50)	92056	80840	$5.00E+05$	0	$1.4672E+07$	$4.648E+06$	$5.00E+05$	0
4(10)	320.69	0.20813	$1.00E+05$	0	380.19	510.25	$1.00E+05$	0
4(30)	325.45	1.5416	$3.00E+05$	0	15946	7051.1	$3.00E+05$	0
4(50)	333.64	3.3105	$5.00E+05$	0	49819	11900	$5.00E+05$	0
5(10)	12.858	0.28301	$1.00E+05$	0	869.56	1241.9	$1.00E+05$	0
5(30)	22.356	0.52122	$3.00E+05$	0	12887	3070	$3.00E+05$	0
5(50)	31.806	0.50936	$5.00E+05$	0	23042	3062.5	$5.00E+05$	0
6(10)	175.39	54.32	$1.00E+05$	0	346.91	1274	$1.00E+05$	0
6(30)	126.4	67.495	$3.00E+05$	0	1281.6	2540.4	$3.00E+05$	0
6(50)	135.12	47.409	$5.00E+05$	0	2356.8	4657.4	$5.00E+05$	0
7(10)	433.94	23.69	$1.00E+05$	0	1267	$4.6252E-13$	$1.00E+05$	0
7(30)	551.63	137.44	$3.00E+05$	0	4696.3	$2.7751E-12$	$3.00E+05$	0
7(50)	570.07	105.33	$5.00E+05$	0	6195.3	0	$5.00E+05$	0
8(10)	393.28	23.459	$1.00E+05$	0	20.197	0.12287	$1.00E+05$	0
8(30)	577.37	212.99	$3.00E+05$	0	20.368	0.094336	$3.00E+05$	0
8(50)	490.93	130.51	$5.00E+05$	0	**20.458**	0.085059	$5.00E+05$	0
9(10)	120.01	$7.2269E-14$	26480	799.31	0.0033333	0.0047946	**18983**	6020.3
9(30)	120.01	$7.2269E-14$	**69423**	1540	0.0043333	0.0050401	$1.568E+05$	39913
9(50)	120.01	$7.2269E-14$	$1.1299E+05$	3772	2.79	1.4605	$4.9838E+05$	8873.1
10(10)	120.01	$7.2269E-14$	38517	1131.4	31.174	12.988	$1.00E+05$	0
10(30)	120.01	$7.2269E-14$	$1.9977E+05$	11139	122.94	32.782	$3.00E+05$	0
10(50)	120.01	$7.2269E-14$	$4.9344E+05$	9271	210.01	48.102	$5.00E+05$	0
11(10)	64449	60668	$1.00E+05$	0	7.5876	1.1929	$1.00E+05$	0
11(30)	$6.6762E+05$	$3.1267E+05$	$3.00E+05$	0	30.659	3.4532	$3.00E+05$	0
11(50)	$1.1557E+06$	$5.8345E+05$	$5.00E+05$	0	54.808	6.0167	$5.00E+05$	0
12(10)	9.99	0	**43280**	3701.2	873.7	1566.8	$1.00E+05$	0
12(30)	**2358.2**	1403.9	$3.00E+05$	0	15214	11587	$3.00E+05$	0
12(50)	28915	8503	$5.00E+05$	0	96421	45347	$5.00E+05$	0
13(10)	180.01	$1.4454E-13$	41123	2053	0.43967	0.16951	$1.00E+05$	0
13(30)	4499.3	974.88	$3.00E+05$	0	1.657	0.47256	$3.00E+05$	0
13(50)	10650	2614.5	$5.00E+05$	0	4.7423	0.93288	$5.00E+05$	0
14(10)	696.49	21.172	99730	1478.9	3.6913	0.31421	$1.00E+05$	0
14(30)	715.66	46.216	$3.00E+05$	0	13.092	0.31847	$3.00E+05$	0
14(50)	730.94	25.802	$5.00E+05$	0	22.696	0.34193	$5.00E+05$	0
15(10)	967.01	0.061026	$1.00E+05$	0	230.47	215.32	**66487**	39189
15(30)	4398.1	10.078	$3.00E+05$	0	330.52	176.73	$2.8213E+05$	54544
15(50)	5895.3	$9.2504E-13$	$5.00E+05$	0	263.56	76.502	$5.00E+05$	0
16(10)	239.76	0.097519	$1.00E+05$	0	162.3	23.794	$1.00E+05$	0
16(30)	239.2	0.11511	$3.00E+05$	0	210.75	100.3	$3.00E+05$	0
16(50)	239.16	0.17519	$5.00E+05$	0	191.18	84.78	$5.00E+05$	0
17(10)	447.56	1.4475	96700	12585	160.47	28.047	$1.00E+05$	0
17(30)	419.07	11.331	$3.00E+05$	0	256.62	162.06	$3.00E+05$	0
17(50)	366.43	18.114	$5.00E+05$	0	237.14	111.19	$5.00E+05$	0

Appendix C

Graphs

This appendix provides addition plots of the HSD of various algorithms used throughout the thesis. The HSD of the median run of HMHH, LDHH, EDHH, EIHH2, and LIHH2 for problems 1 to 17 from the CEC 2005 problem set in dimensions 10, 30, and 50 are plotted in Figures C.1 to C.9. The HSD of the median run of HMHH, EIHH1, PAP, EEA-SLPS, modified AMALGAM-SO, and FAUC-Bandit for the same set of problems are plotted in Figures C.10 to C.18.

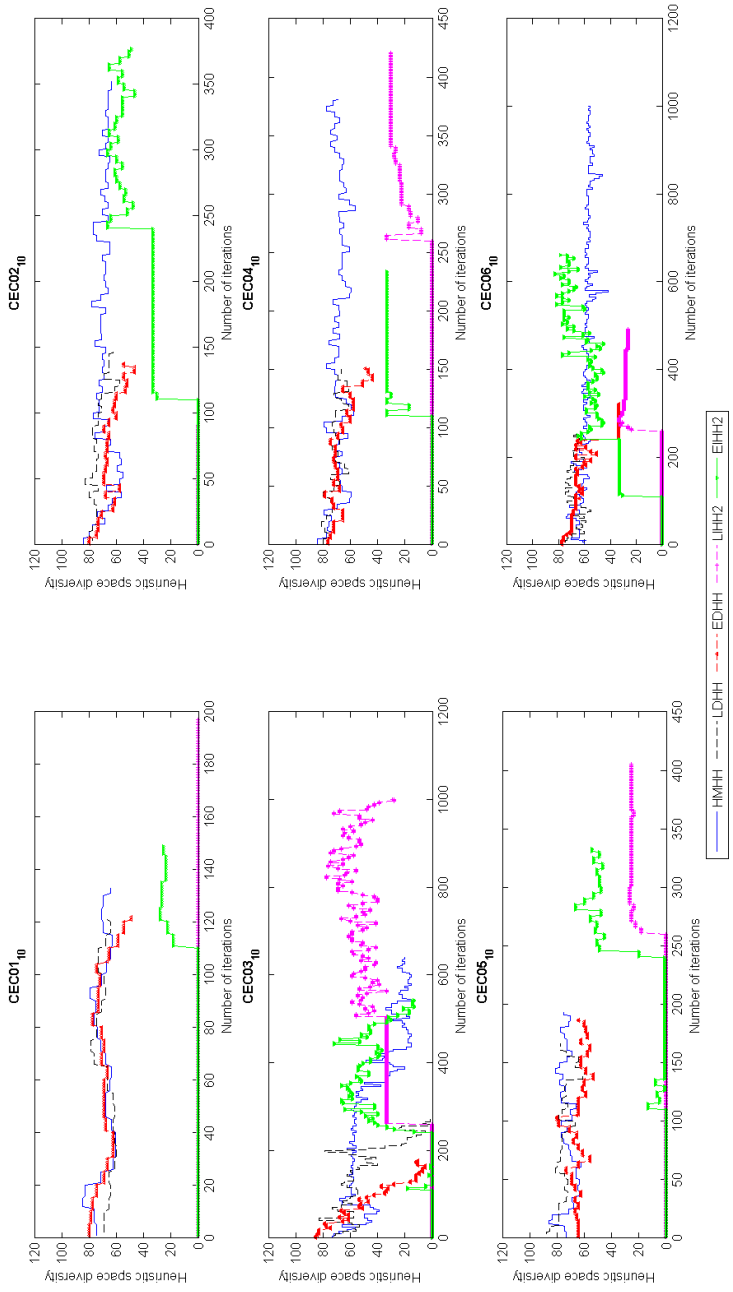

Figure C.1: Graphs of heuristic space diversity (y-axes) versus iterations (x-axes) for the first six CEC 2005 benchmark problems in 10 dimensions.

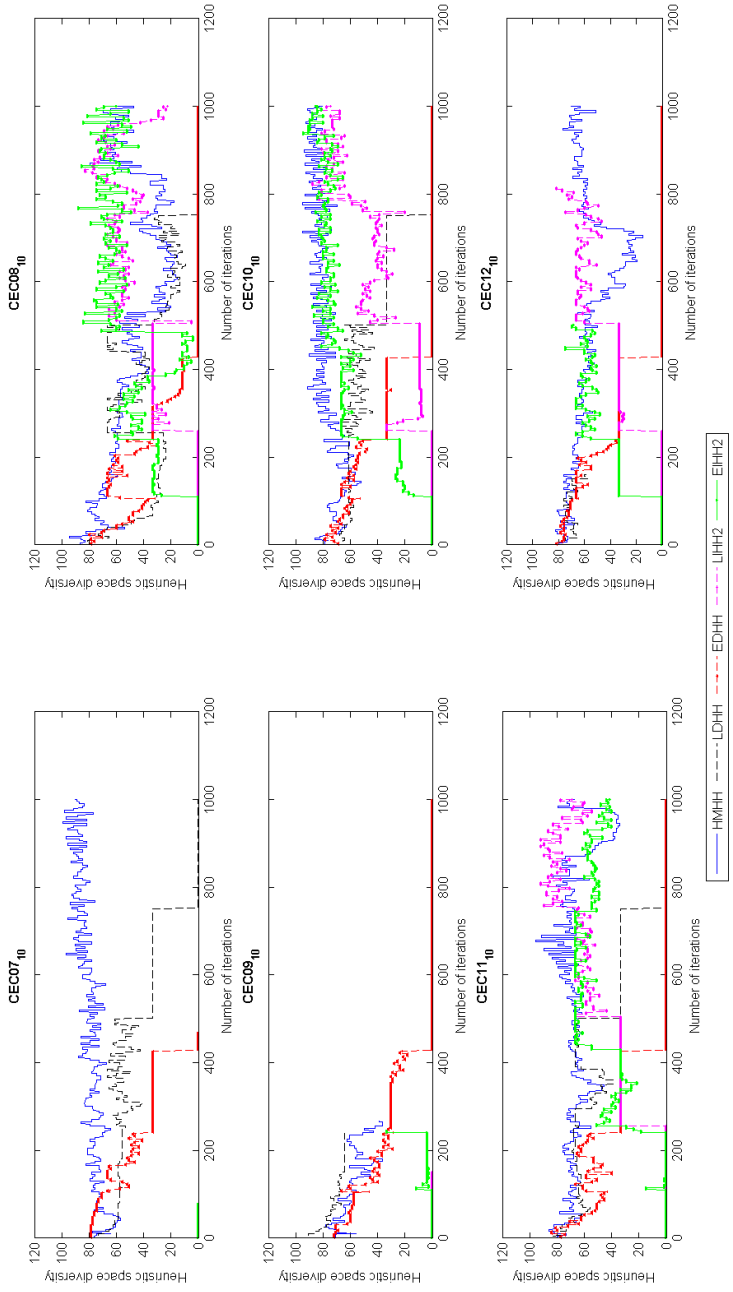

Figure C.2: Graphs of heuristic space diversity (y-axes) versus iterations (x-axes) for problems 7 to 12 of the CEC 2005 benchmark problems in 10 dimensions.

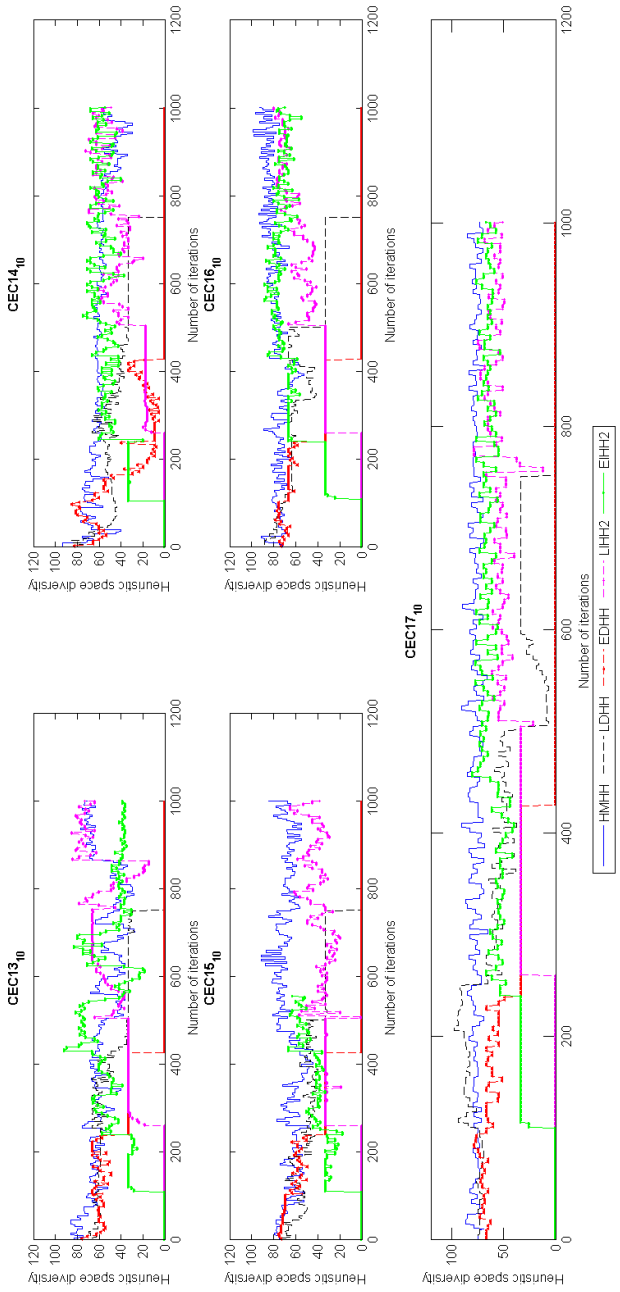

Figure C.3: Graphs of heuristic space diversity (y-axes) versus iterations (x-axes) for problems 13 to 17 of the CEC 2005 benchmark problems in 10 dimensions.

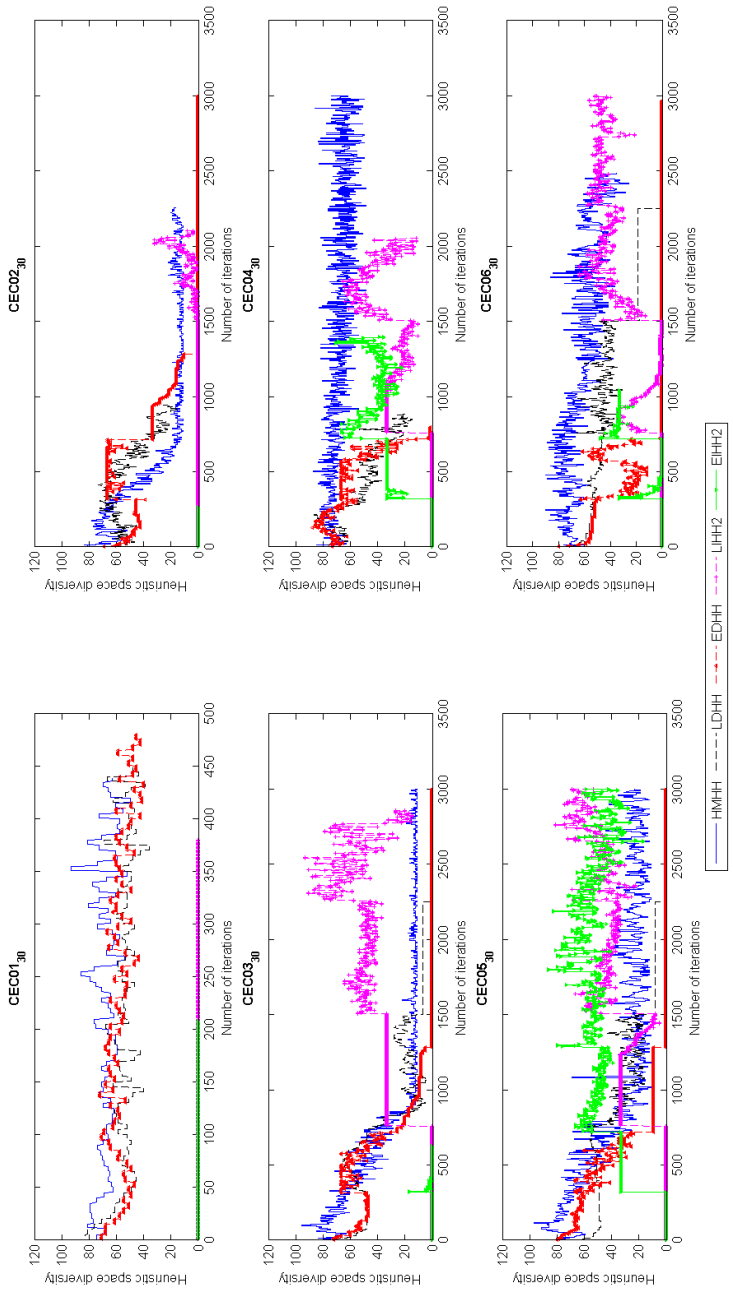

Figure C.4: Graphs of heuristic space diversity (y-axes) versus iterations (x-axes) for the first six CEC 2005 benchmark problems in 30 dimensions.

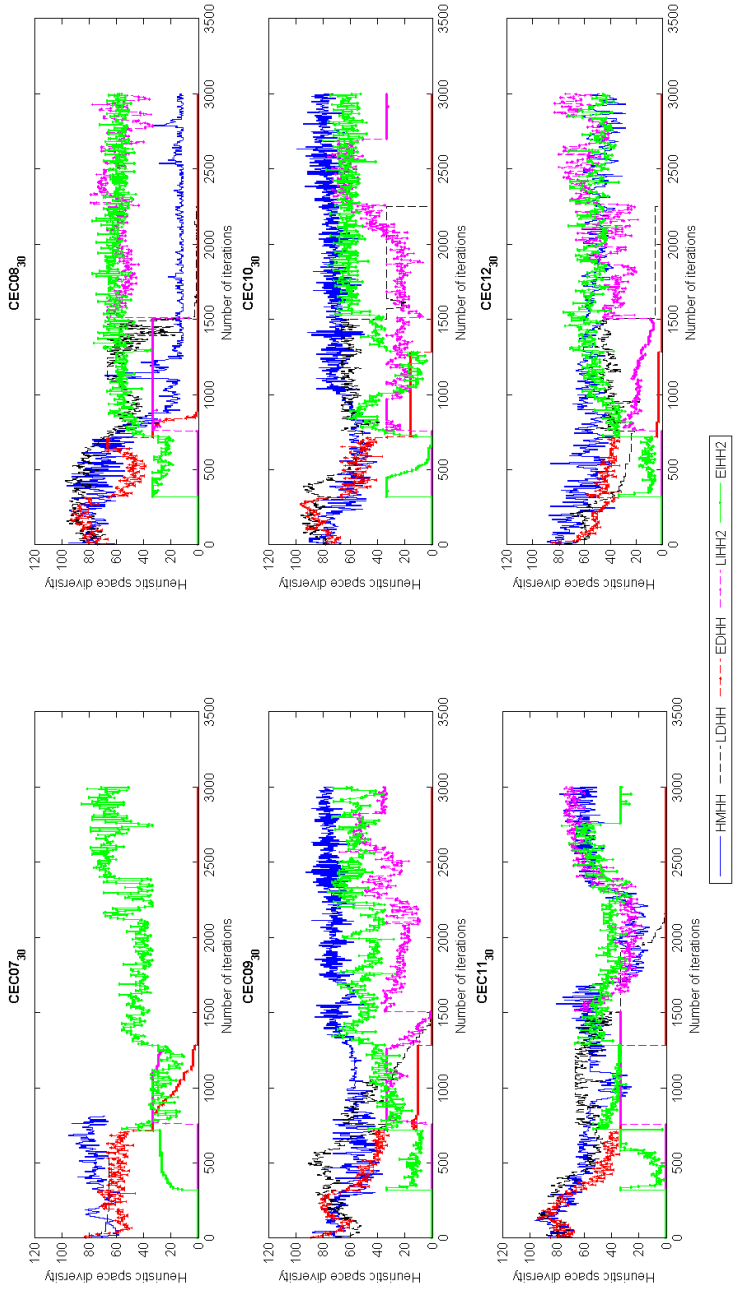

Figure C.5: Graphs of heuristic space diversity (y-axes) versus iterations (x-axes) for problems 7 to 12 of the CEC 2005 benchmark problems in 30 dimensions.

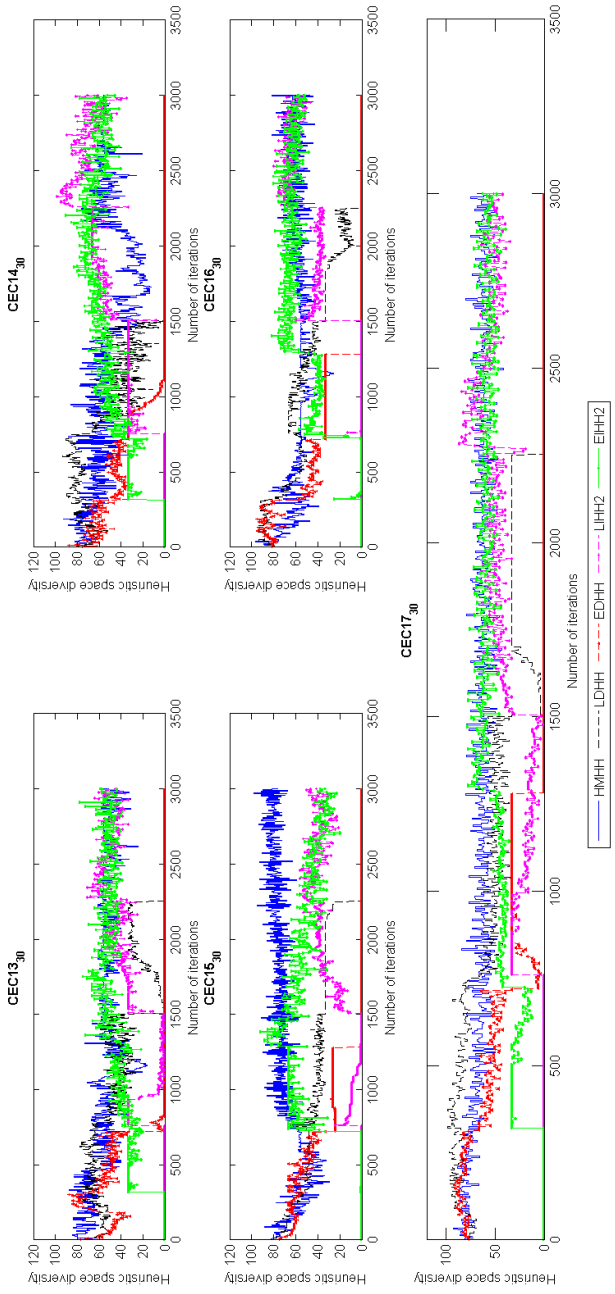

Figure C.6: Graphs of heuristic space diversity (y-axes) versus iterations (x-axes) for problems 13 to 17 of the CEC 2005 benchmark problems in 30 dimensions.

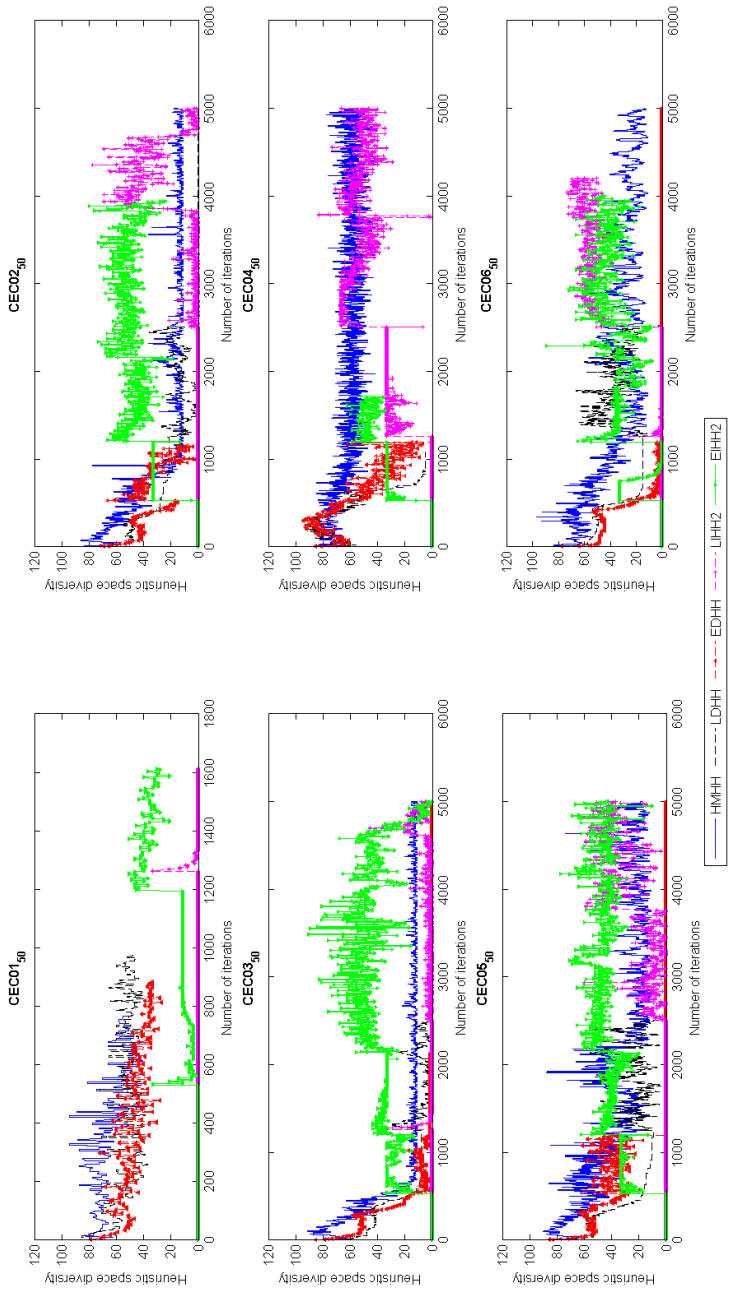

Figure C.7: Graphs of heuristic space diversity (y-axes) versus iterations (x-axes) for the first six CEC 2005 benchmark problems in 50 dimensions.

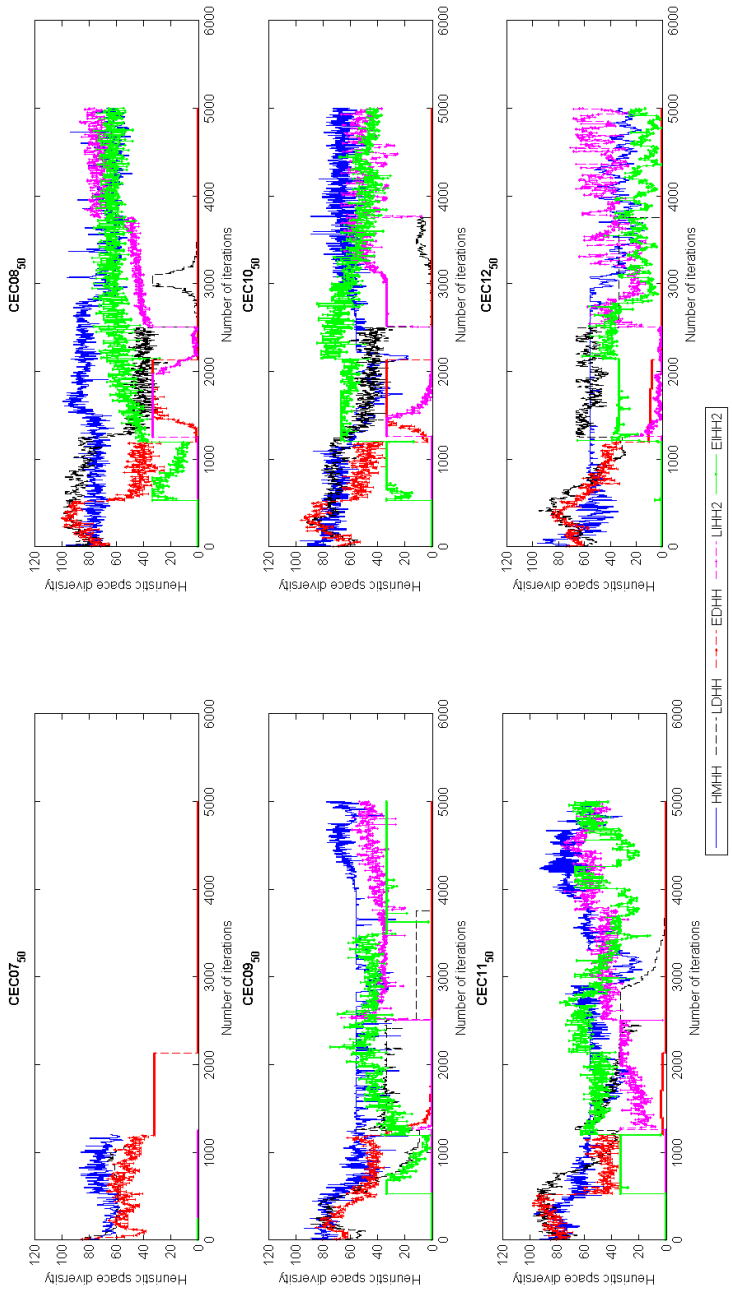

Figure C.8: Graphs of heuristic space diversity (y-axes) versus iterations (x-axes) for problems 7 to 12 of the CEC 2005 benchmark problems in 50 dimensions.

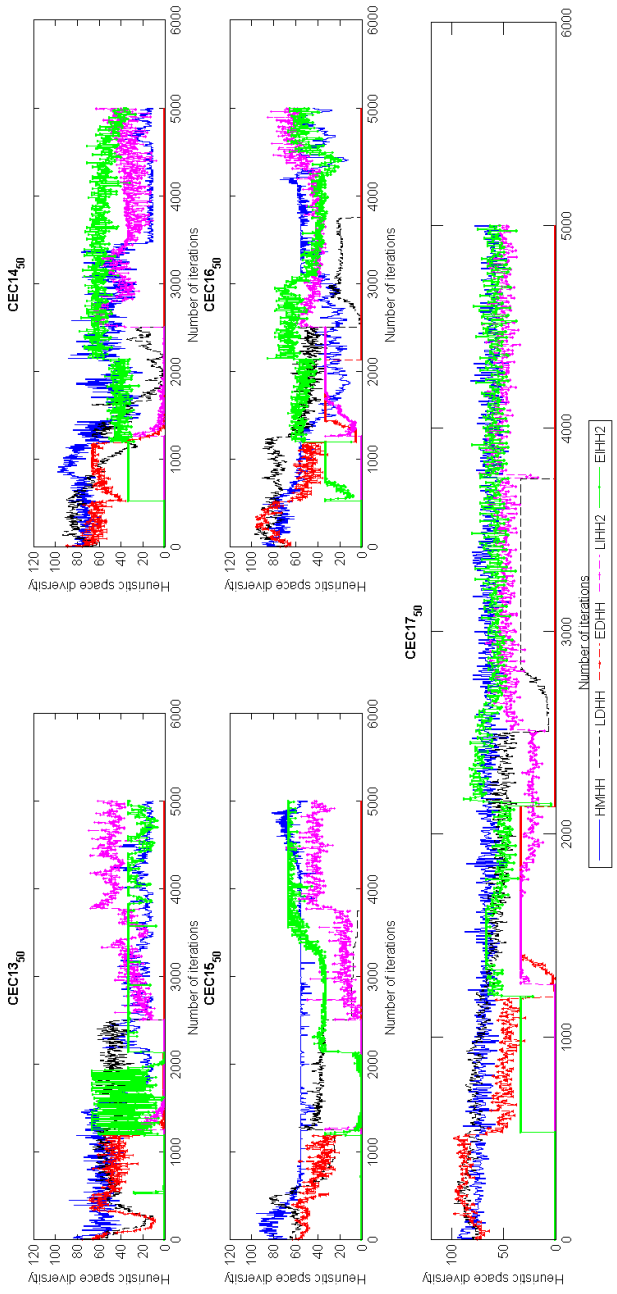

Figure C.9: Graphs of heuristic space diversity (y-axes) versus iterations (x-axes) for problems 13 to 17 of the CEC 2005 benchmark problems in 50 dimensions.

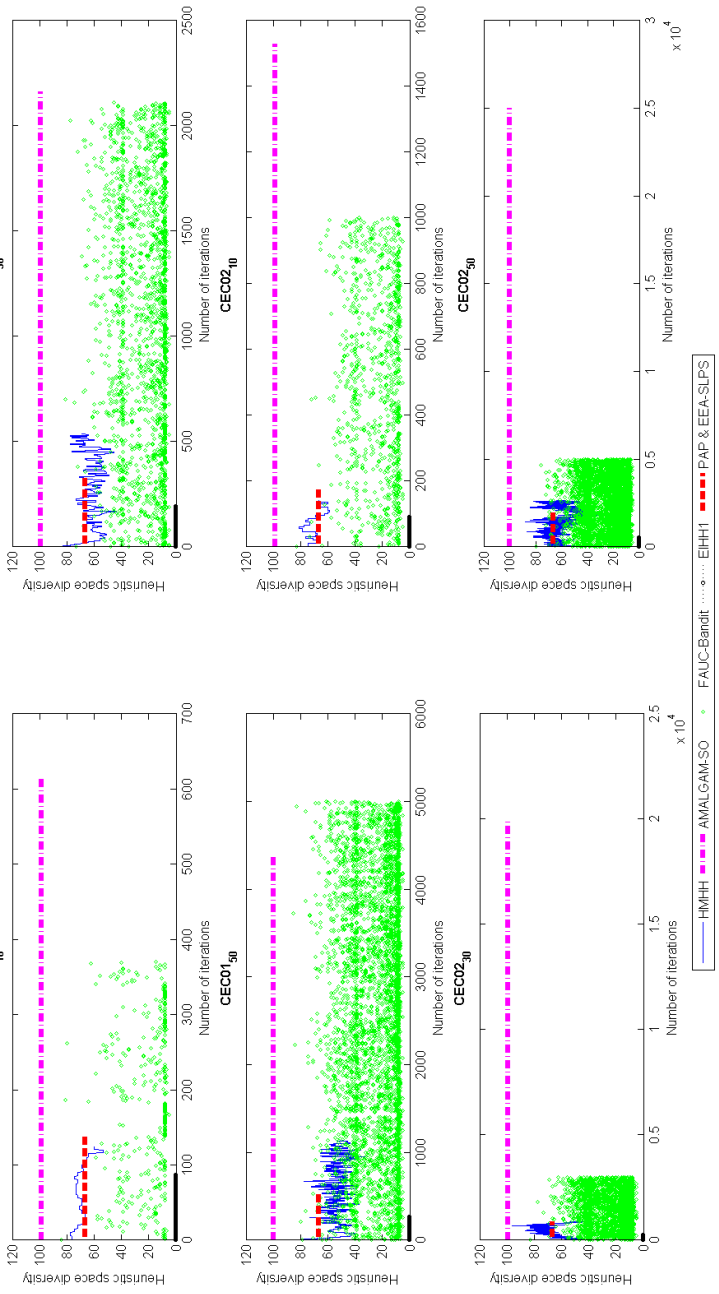

Figure C.10: Comparison of heuristic space diversity (y-axes) versus iterations (x-axes) for HMHH, EIHH1, PAP, EEA-SLPS, modified AMALGAM-SO, and FAUC-Bandit on the first two CEC 2005 benchmark problems in 10, 30, and 50 dimensions.

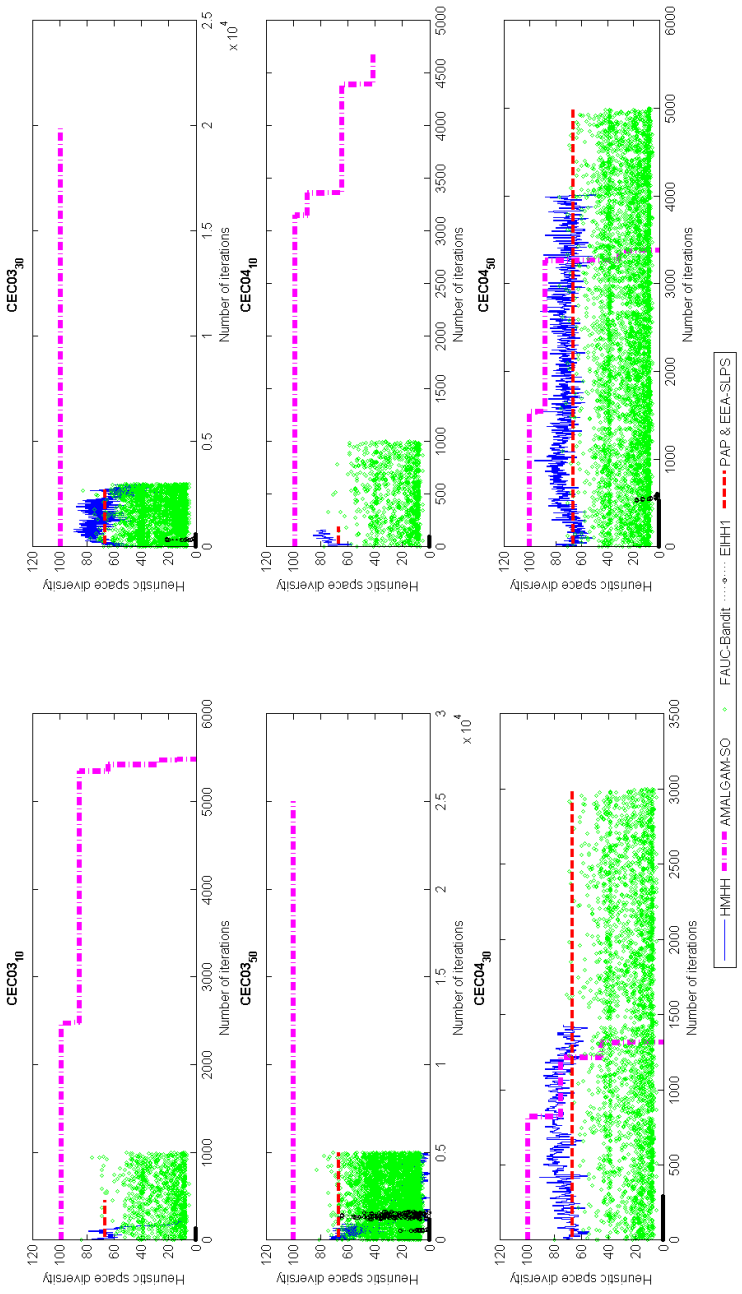

Figure C.11: Comparison of heuristic space diversity (y-axes) versus iterations (x-axes) for HMHH, EIHH1, PAP, EEA-SLPS, modified AMALGAM-SO, and FAUC-Bandit on problems three and four of the CEC 2005 benchmark problems in 10, 30, and 50 dimensions.

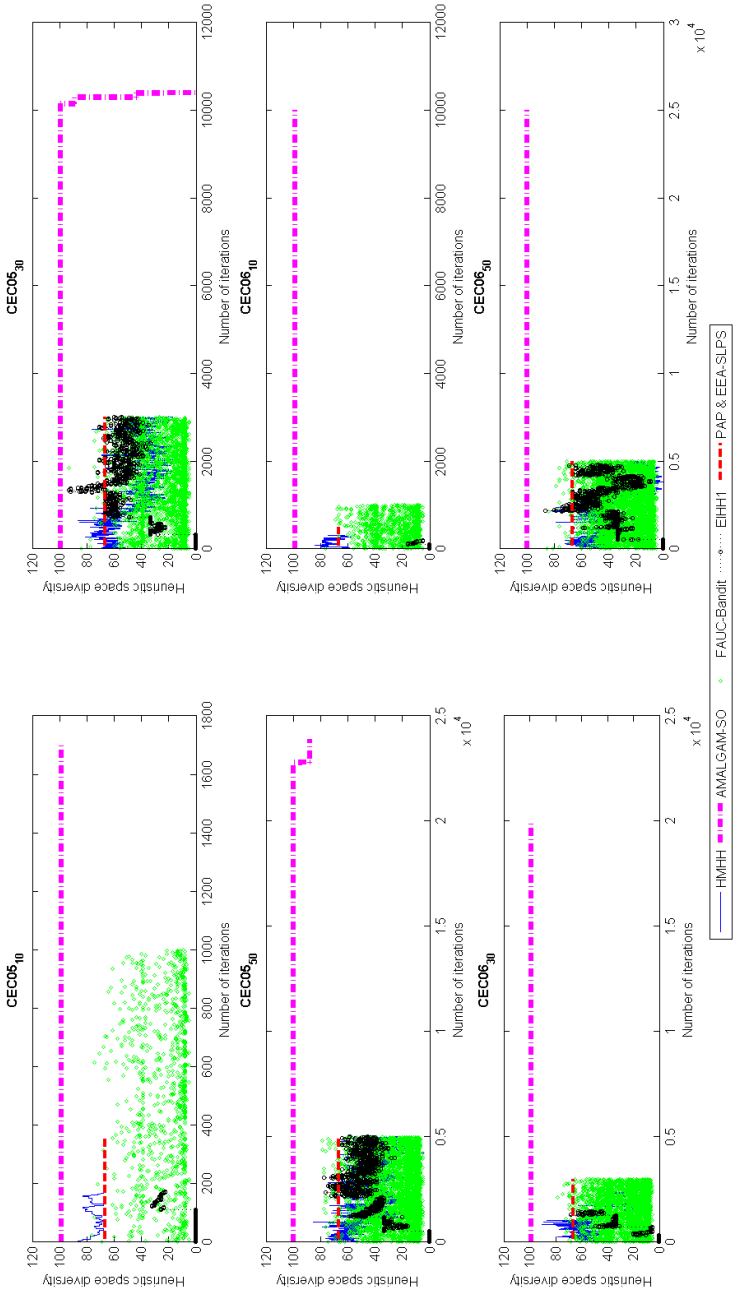

Figure C.12: Comparison of heuristic space diversity (y-axes) versus iterations (x-axes) for HMHH, EIHH1, PAP, EEA-SLPS, modified AMALGAM-SO, and FAUC-Bandit on problems five and six of the CEC 2005 benchmark problems in 10, 30, and 50 dimensions.

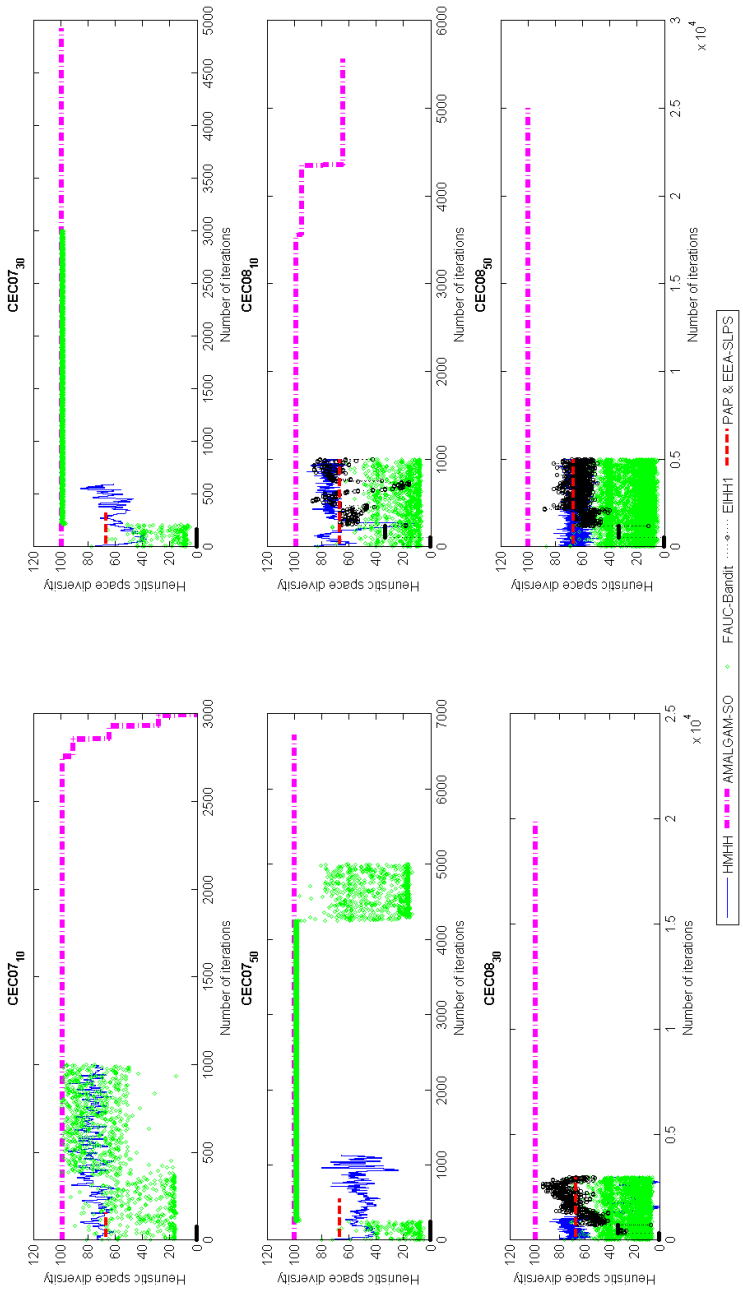

Figure C.13: Comparison of heuristic space diversity (y-axes) versus iterations (x-axes) for HMHH, EIHH1, PAP, EEA-SLPS, modified AMALGAM-SO, and FAUC-Bandit on problems seven and eight of the CEC 2005 benchmark problems in 10, 30, and 50 dimensions.

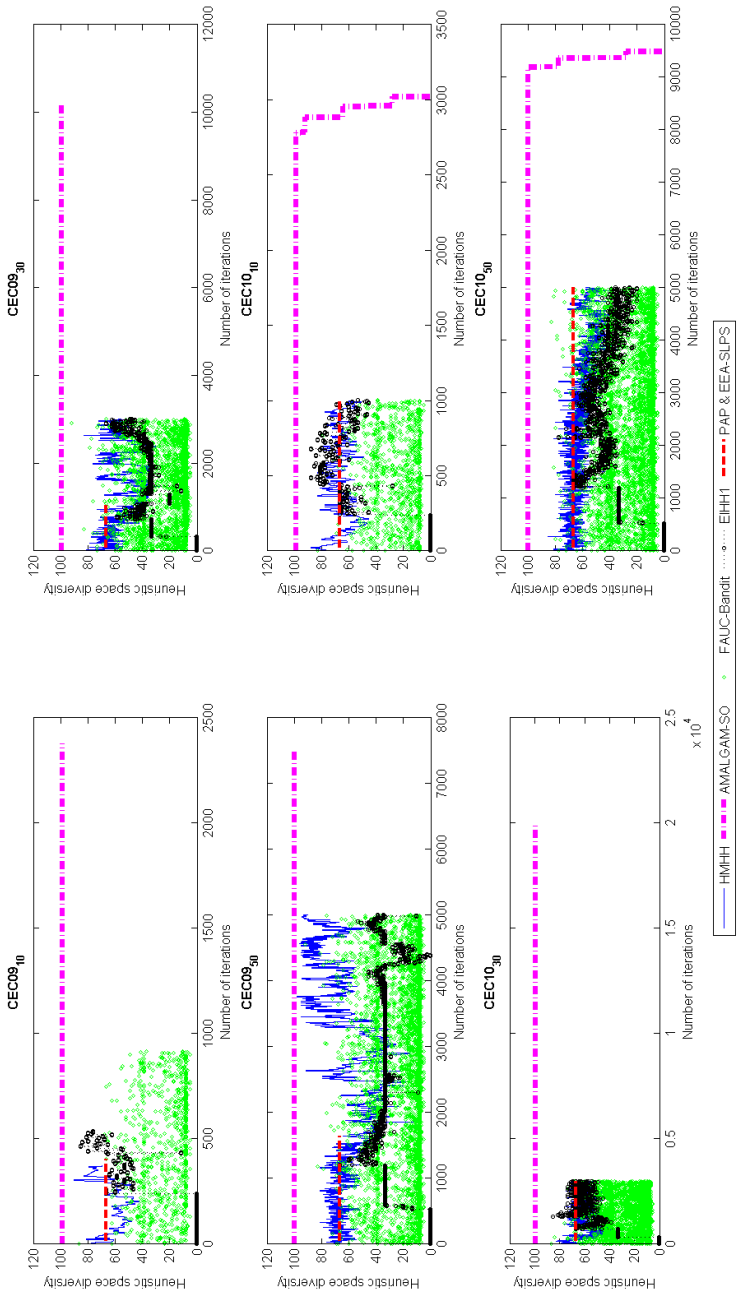

Figure C.14: Comparison of heuristic space diversity (y-axes) versus iterations (x-axes) for HMHH, EIHH1, PAP, EEA-SLPS, modified AMALGAM-SO, and FAUC-Bandit on problems nine and ten of the CEC 2005 benchmark problems in 10, 30, and 50 dimensions.

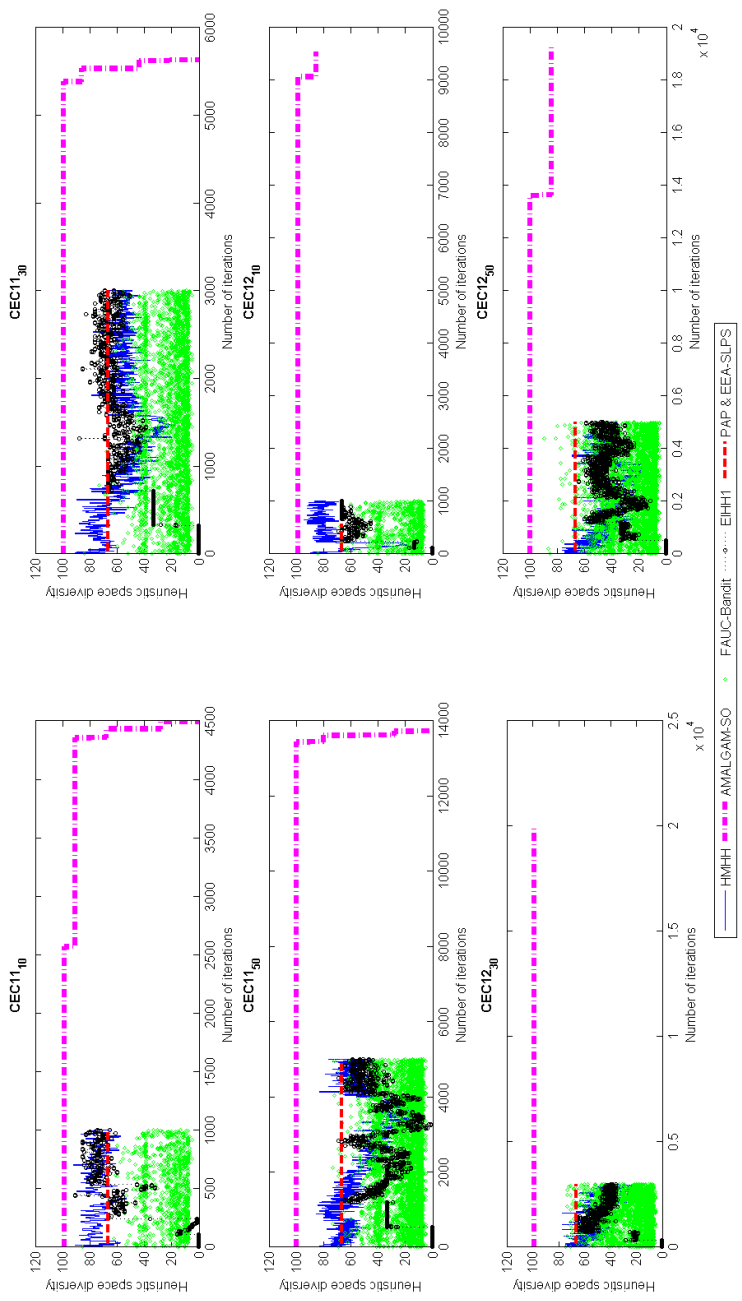

Figure C.15: Comparison of heuristic space diversity (y-axes) versus iterations (x-axes) for HMHH, EIHH1, PAP, EEA-SLPS, modified AMALGAM-SO, and FAUC-Bandit on problems 11 and 12 of the CEC 2005 benchmark problems in 10, 30, and 50 dimensions.

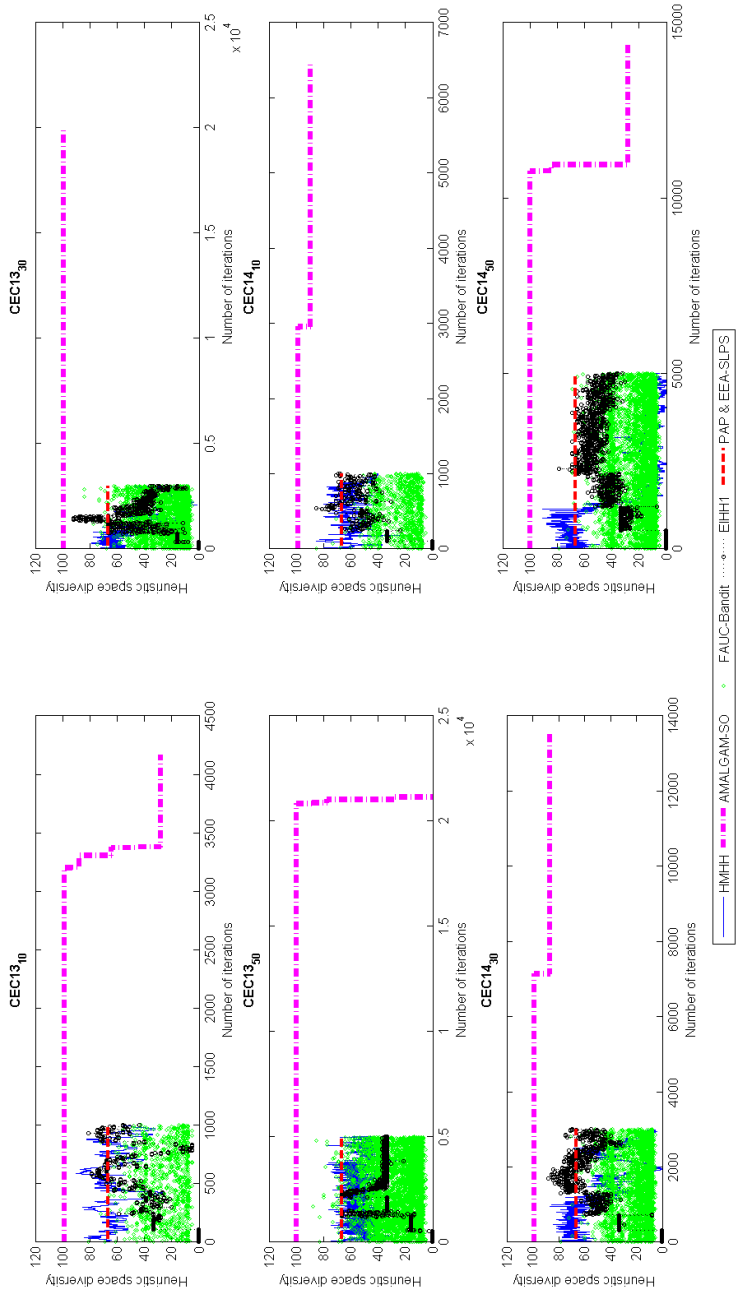

Figure C.16: Comparison of heuristic space diversity (y-axes) versus iterations (x-axes) for HMHH, EIHH1, PAP, EEA-SLPS, modified AMALGAM-SO, and FAUC-Bandit on problems 13 and 14 of the CEC 2005 benchmark problems in 10, 30, and 50 dimensions.

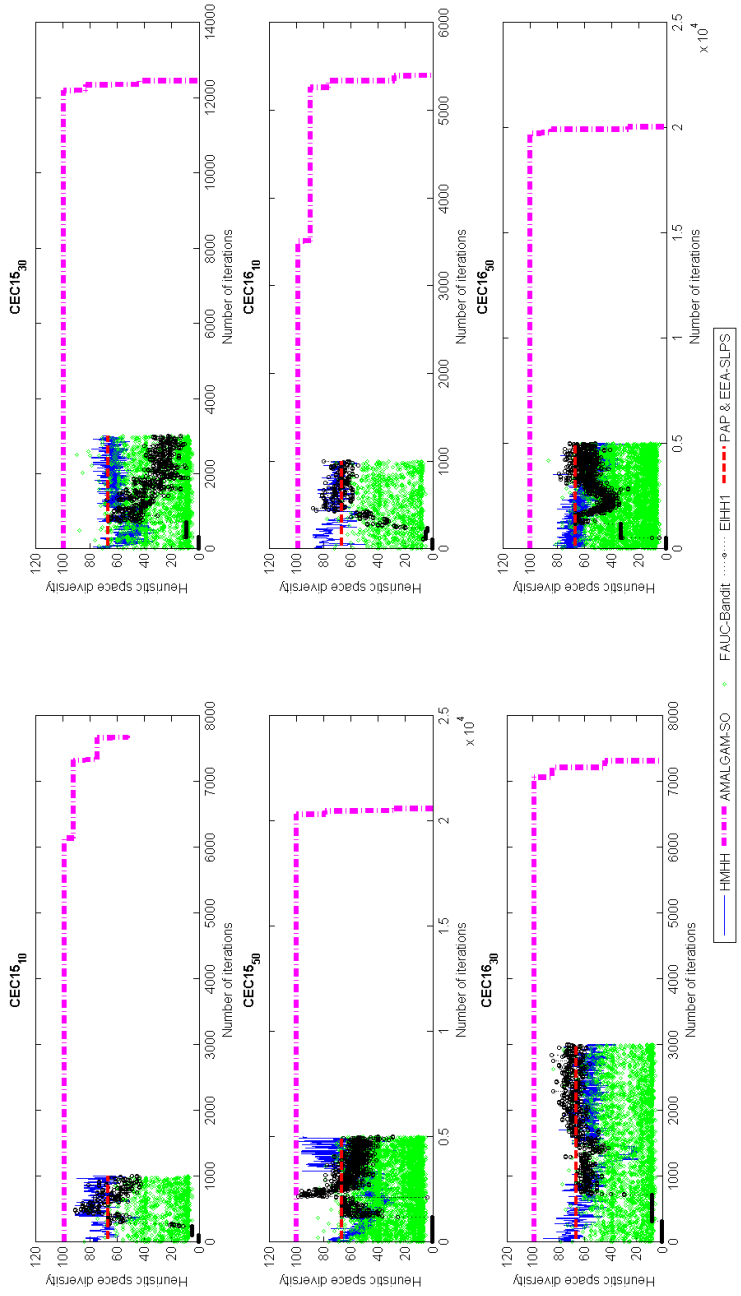

Figure C.17: Comparison of heuristic space diversity (y-axes) versus iterations (x-axes) for HMHH, EIHH1, PAP, EEA-SLPS, modified AMALGAM-SO, and FAUC-Bandit on problems 15 and 16 of the CEC 2005 benchmark problems in 10, 30, and 50 dimensions.

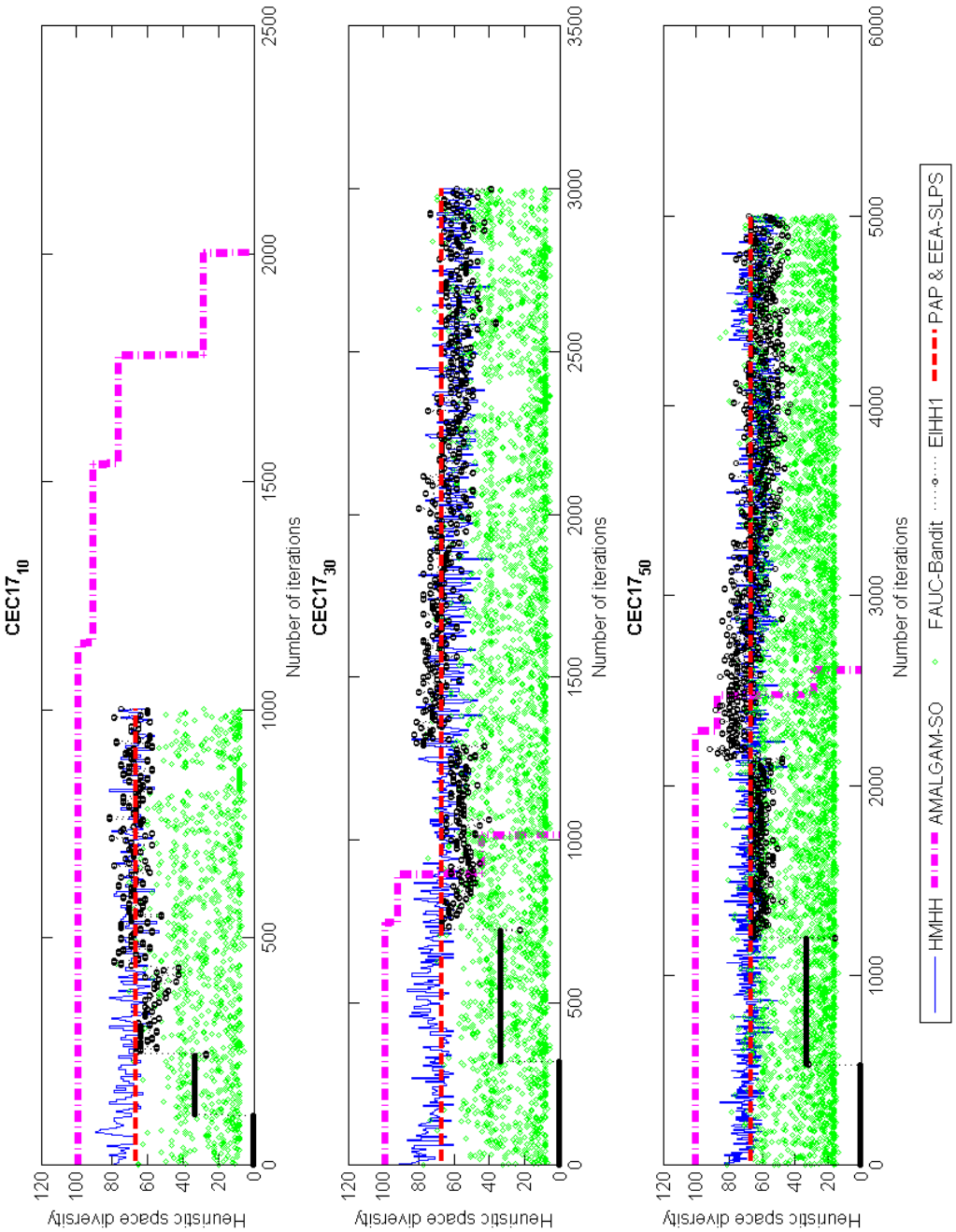

Figure C.18: Comparison of heuristic space diversity (y-axes) versus iterations (x-axes) for HMHH, EIHH1, PAP, EEA-SLPS, modified AMALGAM-SO, and FAUC-Bandit on problem 17 of the CEC 2005 benchmark problems in 10, 30, and 50 dimensions.

Appendix D

Acronyms

All acronyms used in this thesis are listed below in alphabetic order with the meaning of the acronym alongside.

#FEs	Number of function evaluations
ACO	Ant colony optimization
ADIVHH	HMHH with adaptive species selection approach
ALSHH	HMHH with adaptive local search
AMALGAM-SO	Population-based genetic adaptive method for single objective optimization
AOS	Adaptive operator selection
BBOB problem set	Black-box optimization benchmarking problem set
BBPSO	Barebones PSO
BFGS	BroydenFletcherGoldfarbShanno algorithm
BOLT	HMHH with Boltzman selection
CEC	Congress on Evolutionary Computation
CLPSO	Convergent linear particle swarm optimization algorithm
CMAES	Covariance matrix adapting evolutionary strategy algorithm
DE	Differential evolution
DE/R2B/y/z	DE rand-to-best algorithm
DIVHH	HMHH with constant species selection approach

EA	Evolutionary algorithms
EDHH	HMHH with exponentially decreasing HSD
EEA-SLPS	Evolutionary algorithm based on self-adaptive learning population search techniques
EIHH1	HMHH with exponentially increasing HSD with *a priori* information
EIHH2	HMHH with exponentially decreasing HSD without *a priori* information
EPM-PAP	PAP based on an estimated performance matrix
ES	Evolutionary strategy
FAUC-Bandit	Fitness-based area-under-curve bandit algorithm
FFV	Fitness function value
GA	Genetic algorithm
gbest	Global best position in the PSO algorithm
GCPSO	Guaranteed convergence particle swarm optimization
GD	Great deluge algorithm
HH	Hyper-heuristic
HMHH	Heterogeneous meta-hyper-heuristic
HSD	Heuristic space diversity
IEEE	Institute of electrical and electronics engineers
LDHH	HMHH with linearly decreasing HSD
LIHH1	HMHH with linearly increasing HSD with *a priori* information
LIHH2	HMHH with linearly decreasing HSD without *a priori* information
LLM	Low level meta-heuristic
LS	Local search
LS1HH	HMHH with local search applied to the best entity at each iteration
LS2HH	HMHH with local search applied to a randomly selected entity at each iteration
LS3HH	HMHH with local search applied to an entity selected by means of roulette-wheel selection at each iteration
LS4HH	HMHH with local search defined as one of the constituent algorithms
MA	Memetic algorithm

Multi-EA	The multiple EA algorithm
NSDE	The differential evolution with neighbourhood search algorithm
PAP	Population-based algorithm portfolio
PMS	Parallel memetic structure
pbest	Personal best position of a particle in the PSO algorithm
PSO	Particle swarm optimization
RAND	HMHH with random selection strategy
RANK	HMHH with rank-based selection strategy
ROC	Receiver operating curve
ROUL	HMHH with roulette-wheel-based selection strategy
SA	Simulated annealing
SaDE	Self adaptive differential evolution algorithm
SaNSDE	Self-adaptive differential evolution with neighbourhood search
SSD	Solution space diversity
TOUR	HMHH with tournament-based selection
TS	Tabu search
TSHH	HMHH with rank-based selection and tabu search

Appendix E

Symbols

This appendix provides a list of all symbols used in this thesis as well as their definitions.

$\#FEs \triangleq$	The number of function evaluations which were needed to reach the global optimum within a specified accuracy
$a \triangleq$	The first line search parameter
$\boldsymbol{A} \triangleq$	The archive of entity-to-LLM allocation information in the FAUC-Bandit algorithm
$A_i(t) \triangleq$	The index of the LLM applied to entity i at time t
$\alpha \triangleq$	Blend crossover parameter used in the GA
$b \triangleq$	The second line search parameter
$c_{ij}(t) \triangleq$	The j^{th} dimension of the i^{th} offspring at time t
$c_1, c_2 \triangleq$	Acceleration constants used in the PSO algorithm
$c_{1\delta}, c_{2\delta} \triangleq$	The per iteration change in the PSO acceleration constants
$c_{cov} \triangleq$	The learning rate for the covariance matrix update in the CMAES algorithm
$C_{FAUC} \triangleq$	The scaling factor which balances exploration and exploitation in the FAUC-Bandit algorithm
$\boldsymbol{c}(t) \triangleq$	Candidate offspring population of potential candidate solutions for inclusion in $\boldsymbol{X}(t)$ at time t
$\boldsymbol{C}(t) \triangleq$	The covariance matrix of the CMAES algorithm at time t

$c_{c_{CMA}} \triangleq$ The inverse of the backward time horizon of the evolution path $p_{c_{CMA}}$

$c_\mu \triangleq$ The learning rate for the rank-μ update of the covariance matrix in the CMAES algorithm

$c_\sigma \triangleq$ The inverse of the backward time horizon of the evolution path p_σ in the CMAES algorithm

$D \triangleq$ The decay factor of the FAUC-Bandit Algorithm

$D(p, r) \triangleq$ The euclidian distance between entity p and r

$D_h(t) \triangleq$ The heuristic space diversity metric at time t

$\delta \triangleq$ The difference between x_{max} and x_{min}

$\Delta \triangleq$ The speed of the rising water in the great deluge algorithm

$e_c \triangleq$ The threshold parameter which limits the number of consecutive moves where no improvement occurs in the fitness of the *gbest* particle of the GCPSO algorithm

$e_s \triangleq$ The threshold parameter which limits the number of consecutive improvements in the fitness of the *gbest* particle of the GCPSO algorithm

$\epsilon \triangleq$ The user-defined distance allowed between two entities in the species selection mechanism

$\eta \triangleq$ The number of consecutive failures of the *gbest* particle of the GCPSO algorithm

$f(\boldsymbol{x}_i(t)) \triangleq$ The fitness function value of entity $\boldsymbol{x}_i(t)$ at time t

$f_{\delta k} \triangleq$ The improvement in fitness function value due to the k^{th} entity in $\boldsymbol{p}_{r_{succ}}$

$\boldsymbol{f}(p) \triangleq$ The set of features associated with problem p in Rice's algorithm selection problem

$F \triangleq$ The scaling factor used in the DE algorithm

$FFV \triangleq$ The difference between the global optimum and the final fitness function value obtained

$F_i \triangleq$ The scaling factor of individual i used in the self adaptive DE algorithm

$F_\delta \triangleq$ The per iteration change in the scaling factor, F

$\gamma \triangleq$ A constant between 0 and 1

$\boldsymbol{H} \triangleq$ The algorithm space in Rice's algorithm selection problem

$I_{max} \triangleq$ The maximum allowable number of iterations of an algorithm

$\boldsymbol{I}_m(t) \triangleq$ The set of entities allocated to the m^{th} LLM at time t

$k \triangleq$ The number of iterations between re-allocation of entities to LLMs

$\kappa_1 \triangleq$ The migration interval measured in number of iterations of the PAP algorithm

$\kappa_2 \triangleq$ The migration interval measured in number of iterations of the EEA-SLPS algorithm

$l_1 \triangleq$ The learning rate for the rank-one update of the covariance matrix in the CMAES algorithm

$\hat{\lambda} \triangleq$ Number of offspring of best entity in the rank-based selection strategy

$\tilde{\lambda} \triangleq$ Parameter used in the rank-based selection strategy

$LB_{div}(t) \triangleq$ The lower solution space diversity bound at time t

$\boldsymbol{\Lambda}_m \triangleq$ The set of best individuals from all other subpopulations other than $\boldsymbol{P}_m$

$m_i^* \triangleq$ The highest ranking non-tabu LLM w.r.t. entity i

$\boldsymbol{m}(t) \triangleq$ The distribution mean and current most likely best solution in the CMAES algorithm at time t

$MaxTabu \triangleq$ The maximum size of the tabu list in the TS algorithm

$\mu \triangleq$ The mean over 30 simulations of the associated performance measurement

$\mu_{cov} \triangleq$ The parameter for weighting between the rank-one and rank-μ update in the CMAES algorithm

$\mu_{eff} \triangleq$ The variance effective selection mass of the CMAES algorithm

$n_a \triangleq$ The number of LLMs available for selection

$n_m(t) \triangleq$ The number of entities allocated to the m^{th} LLM at time t

$n_p \triangleq$ The total number of subpopulations

$n_q \triangleq$ The number of entities migrated between subpopulations in the PAP algorithm

$n_s \triangleq$ The number of entities in a population

$ns_1 \triangleq$ The number of offspring successfully entering the next generation generated by Eq. (2.22)

$ns_2 \triangleq$ The number of offspring successfully entering the next generation generated by Eq. (2.24)

$n_t \triangleq$ Tournament size of the tournament-based selection strategy

$N_t \triangleq$ Tournament size used in the GA

$n_\varphi \triangleq$ The number of equality constraints in an optimization problem

$n_\varsigma \triangleq$ The number of inequality constraints in an optimization problem

$n_x \triangleq$ The number of dimensions of a particle or individual

$\nu_m(t) \triangleq$ The number of times the m^{th} LLM has been used up until time t in the FAUC-Bandit algorithm

$N(\mu, \sigma^2) \triangleq$ A random number sampled from a normal distribution with mean μ and standard deviation σ

$n_{q1} \triangleq$ The number of entities involved in PAP migration

$n_{q2} \triangleq$ The number of entities involved in EEA-SLPS migration

$p \triangleq$ A problem instance

$p_c \triangleq$ The crossover probability used in a GA

$p_{c\delta} \triangleq$ The per iteration change in the crossover probability, p_c

$p_{c_{CMA}}(t) \triangleq$ The anisotropic evolution path in the CMAES algorithm at time t

$p_m \triangleq$ The probability of selection of the m^{th} LLM

$p_{mut} \triangleq$ The mutation probability used in a GA

$p_r \triangleq$ The reproduction probability used in a DE algorithm

$p_{r_\mu} \triangleq$ The mean of the normal distribution from which p_r is sampled in the SaNSDE algorithm

$\boldsymbol{p}_{r_{succ}} \triangleq$ A set storing the p_r values associated with all successful entities as well as their resulting improvement in fitness value, $f_{\delta k}$, in the SaNSDE algorithm

$p_\sigma \triangleq$ The evolution path or search path in the CMAES algorithm

$\boldsymbol{P} \triangleq$ A set of problem instances

$p_T \triangleq$ The probability of using the DE/rand/x/y base vector selection strategy to generate the trial vector in the SaNSDE algorithm

$\Psi \triangleq$ The water level in the great deluge algorithm

$\boldsymbol{P}_m \triangleq$ The m^{th} subpopulation of entities in the PAP algorithm

$q_l \triangleq$ The l^{th} successful p_{ri} value entered into set $\boldsymbol{p}_{r_{succ}}$ in the SaNSDE algorithm

$q_{mt} \triangleq$ The area under the ROC of the m^{th} LLM at time t in the FAUC-Bandit algorithm

$Q_{\delta m}(t) \triangleq$ The total improvement in fitness function value of all entities assigned to the m^{th} LLM from iteration $t-k$ to iteration t

$r_{1j}(t), r_{2j}(t) \triangleq$ Random numbers in the range [0,1] sampled from a uniform random distribution, $U(0, 1)$

$r_{im}(t) \triangleq$ The rank of the m^{th} LLM with respect to entity i at time t used in the TS-based selection strategy

$r_m(t) \triangleq$ The rank of the m^{th} LLM at iteration t in the rank-based selection strategy

$\rho(t) \triangleq$ The time-dependent GCPSO scaling factor

$\boldsymbol{R}(t+1) \triangleq$ Combined population consisting of $\boldsymbol{C}(t+1)$ and $\boldsymbol{X}(t)$ at time $t+1$ sorted in order of decreasing fitness

$\sigma \triangleq$ The standard deviation over 30 simulations of the associated performance measurement

$\sigma_{BBPSO} \triangleq$ The standard deviation of the population from which $v_{ij}(t+1)$ is sampled in the barebones PSO algorithm

$\sigma_{CMA}(t) \triangleq$ The step size at time t of the CMAES algorithm

$\boldsymbol{S}(\boldsymbol{f}(p)) \triangleq$ The selection mapping of problem p with features $\boldsymbol{f}$ into algorithm space $\boldsymbol{H}$.

$t \triangleq$ Time step during an algorithm's progression

$t_{max} \triangleq$ The maximum number of iterations remaining in the optimization run in the AMALGAM algorithm

$\mathsf{T} \triangleq$ The target number of entities per algorithm in the HSD metric

$\tau \triangleq$ The index of the best solution in a swarm or population

$T_{ij}(t) \triangleq$ The j^{th} component of the i^{th} target vector at time t used in the DE algorithm

$T_{Bolt}(t) \triangleq$ Temperature parameter at time t of the Boltzman selection strategy

$\boldsymbol{\tau} \triangleq$ The tabu list

$U(a, b) \triangleq$ A uniform random distribution with lower bound, a and upper bound, b

$UB_{D_h(t)} \triangleq$ The upper bound of the HSD measure $D_h(t)$

$UB_{div}(t) \triangleq$ The upper solution space diversity bound at time t

$\upsilon \triangleq$ The acceptance rate used in the AMALGAM-SO algorithm

Υ The temperature of the system used in the SA algorithm

$v_{ij}(t) \triangleq$ The j^{th} dimension of the velocity vector of the i^{th} particle time t used in the PSO algorithm

$V_{max} \triangleq$ The maximum velocity of a particle used in the PSO algorithm

$\varphi_q \triangleq$ The q^{th} equality constraint of an optimization problem

$\varrho_i \triangleq$ The rank position of entity i in the FAUC-Bandit algorithm

$\varsigma_q \triangleq$ The q^{th} inequality constraint of an optimization problem

$\vartheta \triangleq$ A Cauchy random variable with scale parameter equal to one as used in the SaNSDE algorithm

$w \triangleq$ The inertia weight used in the PSO algorithm

$W \triangleq$ The window size of the FAUC-Bandit algorithm

$w_\delta \triangleq$ The per iteration change in the inertia weight, w

$w_k \triangleq$ The k^{th} recombination weight in the CMAES algorithm

$x_{ij}(t) \triangleq$ The j^{th} component of the i^{th} entity or candidate solution at time t

$x_j^*(t) \triangleq$ The j^{th} dimension of the best individual at time t

$\hat{x}_{ij}(t) \triangleq$ The j^{th} dimension of the best previous position of individual i at time t used in the PSO algorithm

$\boldsymbol{x}_{i_n} \triangleq$ The n^{th} individual randomly selected from the population in the DE algorithm

$\boldsymbol{x}_{min} \triangleq$ The lower bound of the decision variables

$\boldsymbol{x}_{max} \triangleq$ The upper bound of the decision variables

$\boldsymbol{X}(t) \triangleq$ Population of candidate solutions or entities at time t

$y(m(p)) \triangleq$ The performance mapping of algorithm m on problem p in Rice's algorithm selection problem

$\boldsymbol{Y} \triangleq$ Objective space as defined in Rice's algorithm selection problem

$z \triangleq$ An arbitrary non-negative number

$\zeta \triangleq$ The number of consecutive successes of the *gbest* particle of the GCPSO algorithm

Appendix F

Publications

This appendix lists all the papers that have been published, or are currently under review, that were derived from work done in this thesis:

- J. Grobler, A.P. Engelbrecht, G. Kendall, and V.S.S. Yadavalli. Heuristic space diversity control for improved meta-hyper-heuristic performance. *Information Sciences*, 300: 49–62, 2015.
- J. Grobler, A.P. Engelbrecht, G. Kendall, and V.S.S. Yadavalli. Entity-to-algorithm allocation: a multi-method algorithm comparison *IEEE Transactions on Evolutionary Computation*, submitted for review.
- J. Grobler, A.P. Engelbrecht, G. Kendall, and V.S.S. Yadavalli. The entity-to-algorithm allocation problem: extending the analysis. *Proceedings of the IEEE Symposium on Computational Intelligence and Ensemble Learning*, pages 1–8, 2014.
- J. Grobler, A.P. Engelbrecht, G. Kendall, and V.S.S. Yadavalli. Heuristic space diversity management in a meta-hyper-heuristic framework. *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1863–1869, 2014.
- J. Grobler, A.P. Engelbrecht, G. Kendall, and V.S.S. Yadavalli. Multi-method algorithms: Investigating the entity-to-algorithm allocation problem. *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 570–577, 2013.

- J. Grobler and A.P. Engelbrecht. Solution space diversity management in a meta-hyperheuristic framework. *Proceedings of the BRICS Congress on Computational Intelligence*, pages 270–276, 2013.
- J. Grobler, A.P. Engelbrecht, G. Kendall, and V.S.S. Yadavalli. Investigating the use of local search for improving meta-hyperheuristic performance. *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1–8, 2012.
- J. Grobler, A.P. Engelbrecht, G. Kendall, and V.S.S. Yadavalli. Investigating the impact of alternative evolutionary selection strategies on multi-method global optimization. *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 2337–2344, 2011.
- J. Grobler, A.P. Engelbrecht, G. Kendall, and V.S.S. Yadavalli. Alternative hyper-heuristic strategies for multi-method global optimization. *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1–8, 2010.