Entity-to-Algorithm Allocation: A Multi-Method Algorithm Comparison

Jacomine Grobler[a,*], Andries P. Engelbrecht[b], Graham Kendall[c]

[a] *Department of Industrial and Systems Engineering, University of Pretoria, Pretoria, South Africa*
[b] *Department of Computer Science, University of Pretoria, Pretoria, South Africa*
[c] *School of Computer Science, University of Nottingham, UK and University of Nottingham Malaysia Campus*

Abstract

A multi-method algorithm can 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. An important aspect to consider when designing a multi-method algorithm, is this high level allocation strategy, also referred to as the entity-to-algorithm allocation strategy, which decides which algorithm should be used to optimize which entity at each stage of the optimization process. This paper investigates the entity-to-algorithm allocation problem for a number of recent state-of-the-art multi-method algorithms. These algorithms include the population-based algorithm portfolio, the evolutionary algorithm based on self-adaptive learning population search technique, the fitness based area under the curve bandit operator selection method, the population-based genetic adaptive method for single objective optimization, as well as two heterogeneous meta-hyper-heuristics algorithms, namely the heterogeneous meta-hyper-heuristic algorithm and the exponentially increasing meta-hyper-heuristic algorithm. The aim of the study is to compare the performance of the multi-method algorithms' entity-to-algorithm allocation strategies on a diverse set of floating-point benchmark problems. For this purpose, the multi-method algorithms were standardized to the same pool of low level constituent algorithms. The empirical results show that a heterogeneous meta-hyper-heuristic algorithm's entity-to-algorithm allocation strategy outperforms other multi-method algorithms.

*Corresponding author:
Email address: jacomine.grobler@gmail.com (Jacomine Grobler)

Keywords: Hyper-heuristics, Multi-method algorithms, Ensemble algorithms, Portfolio algorithms

1. Introduction

Over the last five decades a large number of optimization algorithms have 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. 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. Within this context, the simultaneous use of more than one algorithm for solving optimization problems becomes an attractive alternative.

A multi-method algorithm can be described as consisting of one or more entities representing candidate solutions which are evolved 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. This entity-to-algorithm allocation strategy has an important impact on the success of multi-method algorithms since it decides at each time step, which entity should be allocated to which constituent algorithm, i.e. which constituent algorithm should be used to evolve the entity at that time step. This paper focuses on investigating and comparing various entity-to-algorithm allocation strategies which have already been developed in the field of multi-method optimization.

The idea of multiple algorithms collaborating and competing to achieve a better result than each constituent algorithm can achieve in isolation, is not new. In fact, multi-method algorithms have started appearing in various different domains over the last couple of years. Examples include memetic computation [1], algorithm portfolios [2], algorithm ensembles [3], heterogeneous algorithms [4] and hyper-heuristics [5]. Various examples can be found where different operators of the same type of algorithm, e.g. a DE algorithm, are combined to improve algorithm performance and robustness in the field of evolutionary computation and memetic computing. The scope of this article will, however, be confined to ensembles and portfolios consisting of operators from different types of algorithms such as PSO, DE, GA, etc. in a single multi-method framework.

Unfortunately, research within these multi-method fields is often performed in isolation with little interaction between the different research areas. An inspection of the multi-method literature, in general, uncovers similar concepts described by means

of different terminologies. Algorithms are also 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 that 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 the scientific literature, making it difficult to gauge the success of different multi-method algorithms from different research areas. The value that can be obtained from the comparison of different multi-method algorithms under strictly similar conditions, should thus not be underestimated [6].

After a brief analysis of the available multi-method algorithm literature, six multi-method algorithms were selected for evaluation from the subset of multi-method algorithms utilizing different types of algorithms as constituent algorithms. The population-based algorithm portfolio (PAP) of Peng et al. [7], the evolutionary algorithm based on self-adaptive learning population search technique (EEA-SLPS) [8], the population-based genetic adaptive method for single objective optimization (AMAL-GAM-SO) [9], the fitness based area under the curve bandit operator selection method (FAUC-Bandit) [10], as well as two heterogeneous meta-hyper-heuristic algorithms, namely the heterogeneous meta-hyper-heuristic algorithm (HMHH) [11] and the exponentially increasing HMHH algorithm (EIHH1) [12, 13], were selected. These algorithms were standardized with respect to the pool of constituent algorithms, such that the only difference among the algorithms are the way in which algorithms are allocated to entities and the specific control parameters of the allocation mechanisms. The performance of each multi-method algorithm is then compared to that of each of the constituent algorithms. The multi-method algorithms are then compared against each other in order to evaluate the entity-to-algorithm allocation strategies in use by each algorithm. Performance was evaluated on a set of diverse floating-point benchmark problems. The most promising results were obtained by the EIHH1 algorithm which outperformed the other five strategies on a large percentage of the problems. Good performance, when compared to its constituent algorithms, was also demonstrated.

The significance of the paper lies in the fact that, according to the best of the authors' knowledge, this is the first paper in the field of multi-method algorithms to attempt to isolate the entity-to-algorithm allocation strategies of the current state-of-the-art multi-method algorithms. The results will be specifically helpful to new researchers since the entity-to-algorithm allocation strategies are selected from the state-of-the-art in ensemble, portfolio, hyper-heuristic and adaptive operator selec-

tion multi-method algorithms.

The rest of the paper is organized as follows: Section 2 provides a brief overview of the multi-method algorithm literature. Section 3 motivates the selection of the constituent algorithms used in this paper. Section 4 describes the evaluated entity-to-algorithm allocation strategies and their associated multi-method algorithms. The experimental setup and results are documented in Section 5 before the paper is concluded in Section 6.

2. A brief review of related literature

Various algorithms, which adaptively select between different types of operators within the same algorithm, have already been developed. Examples include evolutionary programming algorithms [14], differential evolution (DE) algorithms [15, 16, 17, 18], and particle swarm optimization (PSO) algorithms [19].

The highly related area of memetic computing (MC) also deserves mention and is concerned with algorithms composed of heterogeneous operators (memes) for solving optimization problems [20]. Of specific note is Ong et al's work [21] which investigated the mechanisms of coordination among algorithmic components in MC approaches. They proposed a classification of adaptive coordinating mechanisms consisting of, amongst others, meta-lamarckian learning [22, 23, 24], self-adaptive and co-evolutionary MAs [25, 26, 27, 28], and fitness diversity adaptive algorithms [29, 30, 31, 32, 33].

The scope of this paper is, however, limited to only ensembles and portfolios consisting of operators from different algorithms such as PSOs, DEs, genetic algorithms (GAs), etc. in a multi-method framework. This subset of multi-method algorithms will be discussed and analyzed in more detail throughout the rest of the paper.

Vrugt et al.'s highly successful AMALGAM-SO [9] 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 evolutionary algorithms (EAs). This technique performed well when compared to a number of single method EAs on the 2005 IEEE Congress of Evolutionary Computation benchmark problem set [34].

The heterogenous cooperative algorithm [35] constructs a solution by combining the best solution obtained by different subpopulations. Different EAs are used 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. [7] developed PAP which 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. PAP was later extended to a version based on an estimated performance matrix (EPM-PAP) algorithm [36] which focuses on obtaining good overall performance on a set of problems and the EEA-SLPS algorithm [8] which utilized a different information exchange mechanism.

Recently, Yuen et al. [37] claimed superior performance of their multiple EA (Multi-EA) 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 [38].

From the adaptive operator selection field another successful adaptive strategy selection mechanism was investigated by Fialho et al. [10]. Comparisons of alternative credit assignment methods [39] and strategy selection mechanisms within a differential evolution framework [40] highlighted the superior performance of the FAUC-Bandit technique.

3. Constituent algorithm selection

One of the main ideas of using multiple algorithms in the same optimization run is that the algorithms should complement each other, in other words compensate for the strength and weaknesses of each individual constituent algorithm. Previously Hadka and Reed [41] based the selection of mutation strategies available to their algorithm on the distribution of offspring associated with various operators. Montazeri et al. [42] ensured that their set of low level heuristics contains both exploiter heuristics, designed for intensification, and explorer heuristics, aimed at diversification. Peng et al. [7] proposed a pairwise metric which can be used to determine the risk associated with an algorithm failing to solve the problem in question. Engelbrecht [19] selected complementary swarm behaviours in a heterogeneous PSO by analyzing the exploration-exploitation finger prints of the different PSO updates.

Initially, eight constituent algorithms commonly used in literature as multi-method constituent algorithms, were selected for use in this study. These algorithms included

three DE variations, two PSO variations, one GA and the CMAES algorithm. Two aspects were considered in the selection of the proposed set of constituent algorithms from this larger set of available options. Firstly, the single-method performance of a constituent algorithm on the benchmark problem set used in this paper was considered. Secondly, the constituent algorithms were selected to ensure a diverse set of constituent algorithms based on their diversity profiles [43]. The proposed set of constituent algorithms considered to be best for the 2005 CEC benchmark problem set consists of the following algorithms:

- A GA with a floating-point representation, tournament selection, blend crossover [44, 35], and self-adaptive Gaussian mutation [11];
- The guaranteed convergence PSO algorithm (GCPSO) [45];
- The self-adaptive DE algorithm with neighborhood search (SaNSDE) [46]; and
- The covariance matrix adapting evolution strategy algorithm (CMAES) [47].

As illustrated in Figure 1 for $F11$ from the 2005 IEEE Congress of Evolutionary Computation benchmark problem set in 50 dimensions [34], 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, there is a greater chance of counteracting an inappropriate population diversity management strategy by one algorithm from the set. For example, if three out of the four constituent algorithms 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.

Solution space diversity (SSD) was defined as:

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

by Olorunda and Engelbrecht [48], where n_s is the number of entities in the 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.

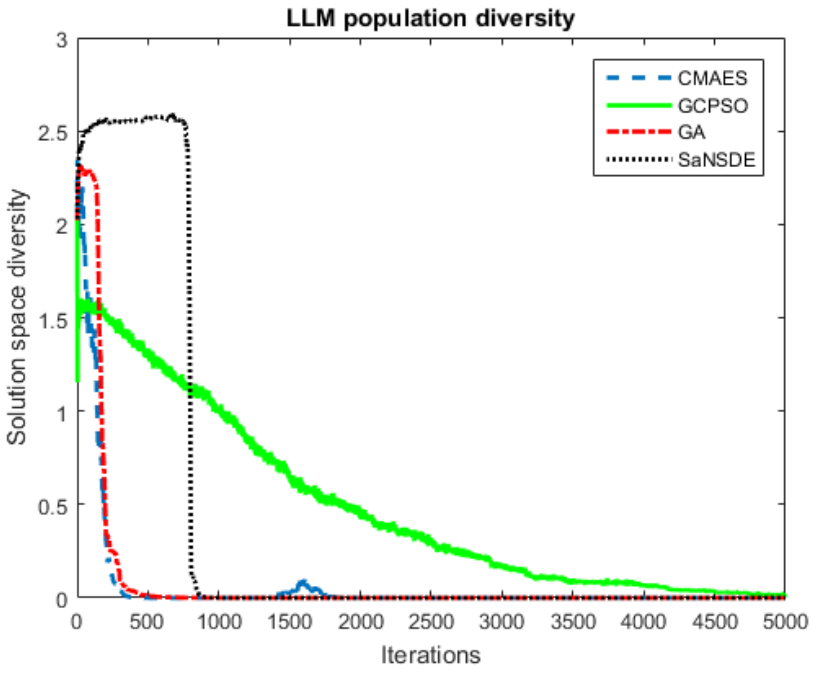

Figure 1: Diversity profiles of different LLMs on $F11$ from the CEC 2005 benchmark problem set in 50 dimensions.

4. Entity-to-algorithm selection strategy comparison

Six entity-to-algorithm allocation selection strategies were selected for comparison from the subset of multi-method algorithms considering different types of established optimization algorithms as constituent algorithms discussed in Section 2. These strategies are taken from the PAP [7], EEA-SLPS [8], AMALGAM-SO [9], FAUC-Bandit [40], HMHH [11], and EIHH1 [12, 13] algorithms. The rest of this section describes these algorithms in more detail.

4.1. Population-based algorithm portfolio

Peng et al. [7] developed PAP which 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 1, entities are divided into subpopulations which are adapted in parallel by an assigned constituent algorithm, where one constituent algorithm 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.

Algorithm 1: The PAP algorithm [7].

Let n_p be the total number of subpopulations
Initialize 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

It should be noted that PAP makes use of a static algorithm selection mechanism. The allocation of entities to constituent algorithms is 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.

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

The only difference between PAP [7] and the EEA-SLPS algorithm [8] is the information exchange mechanism. The worst individual of each subpopulation is

replaced by the current best individual of the entire ensemble at each iteration as indicated in Algorithm 2.

Algorithm 2: The EEA-SLPS algorithm [8].

```
Let x*(t) be the best solution in the entire portfolio at time t.
Initialize n_p subpopulations, P_1, P_2, ..., P_{n_p}
while no stopping condition is satisfied do
    Evaluate all the entities in P_1, P_2, ..., P_{n_p}
    for all algorithms m do
        Adapt P_m using algorithm m
    end
    Activate migration procedure as follows:
    for all subpopulations m do
        if x*(t) ∈ P_m then
            Replace the worst solution in P_m by x*(t)
        end
    end
end
```

4.3. The population-based genetic adaptive method for single objective optimization

The single objective version of the highly successful AMALGAM is the third algorithm that is investigated. Similar to the HMHH algorithm, AMALGAM-SO [9] also uses a common population of entities that are adapted over time. At the start of the optimization run, entities are assigned to constituent algorithms. The entities are then adapted by their allocated constituent algorithms until the population reaches "convergence". "Convergence" is obtained by meeting any one of a complex set of stopping conditions [9] 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 constituent algorithm is reconsidered. Each constituent algorithm's performance, since the previous update, is used to determine the number of entities assigned to it according to,

$$n_m(t) = \left\lfloor n_s \frac{Q_{\delta m}(t)}{\sum_{m=1}^{n_a} Q_{\delta m}(t)} \right\rfloor, \tag{2}$$

where $n_m(t)$ is the number of entities allocated to the m^{th} constituent algorithm at time t, n_s is the population size, and $Q_{\delta m}(t)$ is the total improvement in fitness

function value of all entities assigned to the m^{th} constituent algorithm 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 3.

Algorithm 3: The AMALGAM-SO algorithm [9].

```
Let t_max be the maximum number of iterations remaining in the optimization run
t = 0
Randomly initialize n_m(1), ∀m
while no stopping condition is satisfied do
    CONVERGED = FALSE
    Initialize the parent population X(t) of size n_x by means of a latin hypercube sampling strategy [9].
    while algorithm has not converged (CONVERGED = FALSE) do
        Create offspring population c(t+1) by applying the constituent algorithms to X(t)
        Combine parent and offspring populations, X(t) and c(t+1), and sort in order of decreasing fitness to obtain R(t+1)
        Select X(t+1) from R(t+1) using the species selection mechanism described in [9].
        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), ∀m according to Equation (2)
            Increase population size n_s to 2n_s
            Reset t = 0 and recompute t_max
        end
    end
end
```

It is important to note that the purpose of this paper is to investigate the mechanisms used to allocate entities to constituent algorithms. 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 [9].

A number of aspects need to be addressed with regard to this modified algorithm's performance. Firstly, note that the AMALGAM-SO implementation in [9] is significantly biased towards CMAES. CMAES is known to be one of the best performing algorithms on the CEC 2005 benchmark problem set [47]. CMAES has 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 AMALGAM-SO also ensures 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 of the constituent algorithms at the start of the optimization run and setting the minimum number of entities per algorithm equal for all algorithms.

4.4. Fitness-based area-under-curve multi-armed bandit

The fourth algorithm considered is an online operator selection strategy from the adaptive operator selection field [49]. The FAUC-Bandit algorithm [40] consists of a credit assignment mechanism and a constituent algorithm selection mechanism as shown in Figure 2. The credit assignment mechanism evaluates the performance of each of the available constituent algorithms based on previous successes and assigns a credit to each constituent algorithm. The constituent algorithm selection mechanism then utilizes these constituent algorithm credits in a dynamic bandit-based constituent algorithm selection mechanism to select which constituent algorithm 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 constituent algorithm 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 constituent algorithm under consideration. If an entity has not been generated by the specific constituent algorithm, a

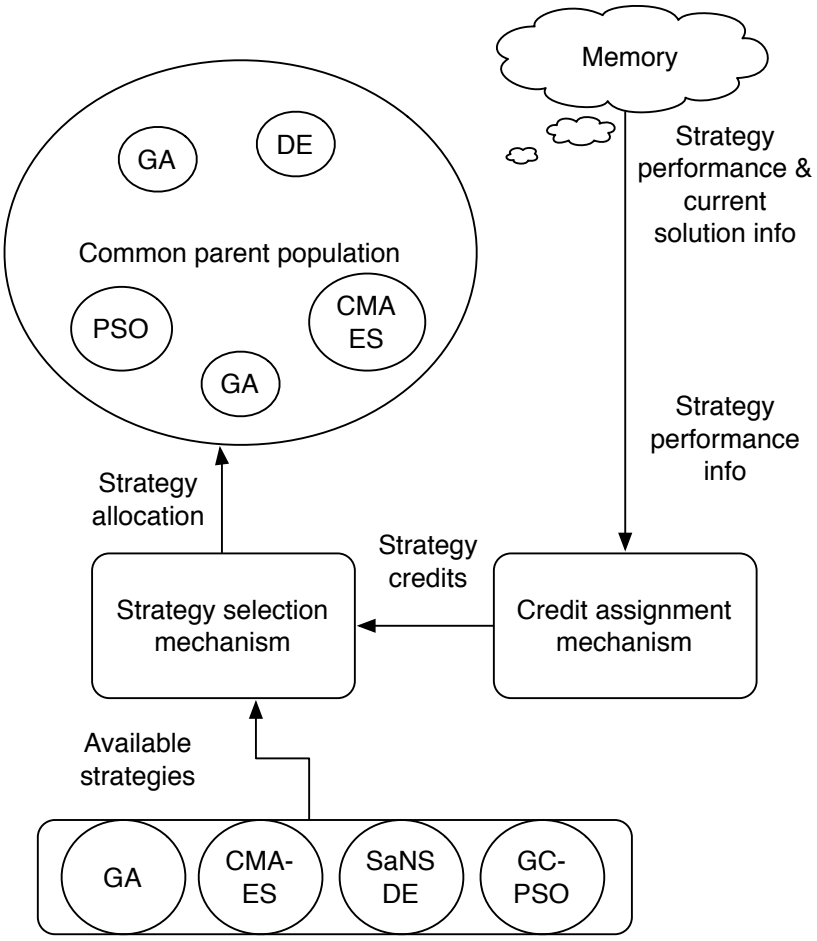

Figure 2: The fitness-based area-under-curve bandit algorithm [40].

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 3 is obtained for each algorithm. The area under the ROC of the m^{th} constituent algorithm at time t, namely $q_m(t)$ is then used as the credit associated with the constituent algorithm under consideration.

The constituent algorithm selection mechanism is based on a maximization op-

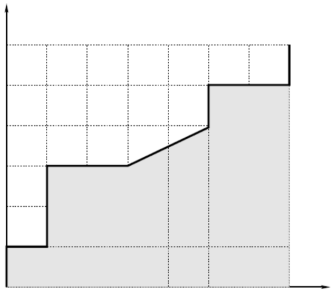

Figure 3: ROC curve of ranked entities (x-axis) versus number of entities generated by the constituent algorithm under consideration (y-axis) [40].

erator,

$$q_m(t) + C\sqrt{\frac{2\log\sum_k \nu_k(t)}{\nu_m(t)}}, \tag{3}$$

where C is a user-defined constant balancing exploration and exploitation and $\nu_m(t)$ refers to the number of times the m^{th} constituent algorithm has been used up until time t. The constituent algorithm 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 4.

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.

4.5. The Heterogeneous Meta-hyper-heuristic

The HMHH algorithm, as illustrated in Figure 4, consists of a common population of entities, each representing 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 (with equal probability) 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 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

Algorithm 4: The FAUC-Bandit algorithm [7].

```
Initialize the parent population X
Randomly select an initial constituent algorithm A_i(1) from the set of
  constituent algorithms to apply to the first entity.
while no stopping condition is satisfied do
    Adapt entity i using constituent algorithm A_i(t)
    Calculate Q_{δA_i(t)}(t), the total improvement in fitness function value of
      entity i after application of algorithm A_i(t) and add entity-to-constituent
      algorithm allocation information to archive A of size W
    Sort all entries in A according to greatest improvement obtained
    for all constituent algorithms m do
        Calculate the area under the ROC curve, q_m(t).
        Calculate the maximization value of algorithm m as defined in
          Equation (3).
    end
    Allocate entity i + 1 to the algorithm m_{i+1} which maximizes Equation (3).
    i = i + 1
end
```

specifically,

$$Q_{\delta m}(t) = \sum_{i=1}^{|\boldsymbol{I}_m(t)|} (f(\boldsymbol{x}_i(t-k)) - f(\boldsymbol{x}_i(t))) \quad \forall i \in \boldsymbol{I}_m(t) \tag{4}$$

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 have already been investigated in the context of the HMHH algorithm.

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. All entities thus have access to the genetic material of all other entities. The HMHH pseudocode is provided in Algorithm 5.

A variation on the standard HMHH algorithm is also considered in this paper.

Algorithm 5: The heterogeneous meta-hyper-heuristic.

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 (4).
    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 rank-based tabu search strategy of Burke et al. [50].
    end
end

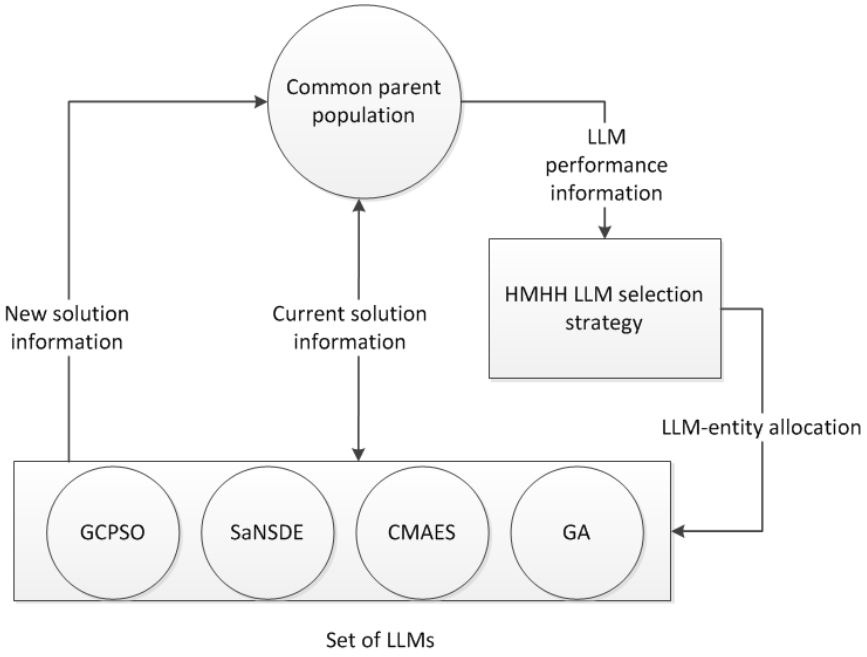

Figure 4: The heterogeneous meta-hyper-heuristic.

The EIHH1 algorithm forces the hyper-heuristic to move from exploitation to exploration. At the start of the optimization run only one constituent algorithm is made available to the hyper-heuristic. As the optimization process progresses, additional constituent algorithms are made available at predetermined exponential time intervals. The idea is to obtain maximum performance gains from the first constituent algorithm. As the performance gains decrease, the rest of the constituent algorithms become available to diversify the heuristic space and improve the overall algorithm performance. Unfortunately, EIHH1 assumes *a priori* knowledge of the LLM performance on the benchmark problem set being solved. The constituent algorithms 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.

5. Empirical evaluation

The six entity-to-algorithm allocation strategies were evaluated on the first 17 problems of the 2005 IEEE Congress of Evolutionary Computation benchmark problem set [34] in dimensions 10, 30, and 50. This benchamark problem set was selected

since it enables algorithm performance evaluation on both unimodal and multimodal functions and includes various expanded and hybridized problems, some with noisy fitness functions.

The constituent algorithm control parameters values listed in Table 1 were found to work well for the algorithms under study in previous research [51]. 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}. The control parameters of the six multi-method algorithms are also specified in Table 1. PAP and AMALGAM-SO's parameters were optimized for the CEC 2005 problems by the original authors, thus these parameters were used as is. In the authors' original work, the FAUC-bandit algorithm's parameters were, however, optimized for the BBOB-2010 noiseless benchmarking problem suite. F-race [52] was used to tune the FAUC-bandit algorithm's control parameters specifically for the 2005 IEEE CEC benchmark problem set. 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. Similar to the analysis of Fialho et al. [10], 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 30 configurations. The final values for C, D, and W obtained by F-race is also provided in Table 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.

Algorithm results 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 to reach the global optimum, $\#FEs$, was recorded. Where the global optimum could not be found within the maximum number of iterations, I_{max}, the difference between the final solution at I_{max} and the global optimum, denoted by FFV, are presented in Tables 2 to Tables 6, where μ and σ denote the mean and standard deviation associated with the corresponding performance measure. The best performing constituent algorithm and best performing multi-method algorithm results for each problem are also highlighted.

Mann-Whitney U tests were used to evaluate the various multi-method algorithms against their constituent algorithms. The results in Table 7 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 strat-

Table 1: Algorithm parameters.

PARAMETER	**VALUE**
CONSTITUENT ALGORITHMS	
PSO parameters	
Acceleration constant (c_1)	$2.0 \longrightarrow 0.7$
Acceleration constant (c_2)	$0.7 \longrightarrow 2.0$
Inertia weight (w)	$0.9 \longrightarrow 0.4$
SaNSDE parameters	As in [46].
GA parameters	
Probability of crossover (p_c)	$0.6 \longrightarrow 0.4$
Probability of mutation (p_m)	0.1
Blend crossover parameter (α)	0.5
GA tournament size (N_t)	13
CMAES parameters	As in [47].
MULTI-METHOD ALGORITHMS	
Common algorithm parameters	
Population size (n_s)	100
Maximum number of iterations (I_{max})	$100n_x$
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
Entities involved in PAP migration (n_{q1})	3
EEA-SLPS migration entities (n_{q2})	1
Modified AMALGAM-SO	As in [9].
FAUC-Bandit	
Size of time window (W)	50
Decay factor (D)	0.5
Exploration-exploitation constant (C)	1
HMHH and EIHH1	
Iterations between re-allocation (k)	5
Size of HMHH tabu list	3
Size of EIHH1 tabu list	2 if $n_a = 4$; 1 if $n_a = 3$; 0 if $n_a \leq 2$

Table 2: 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	$2.0465e+05$	92270
9(50)	7.615	5.9228	$5,00E+05$	0	6.4427	8.5585	$3.4528e+05$	$1.6708e+05$
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 3: 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	$1.0028E+05$	182.09	$3.3135E+05$	$2.3001E+05$	100000	0
3(30)	$3.9749E+05$	$6.7835E+05$	$3.0047E+05$	297.61	$2.1774E+06$	$9.4239E+05$	300000	0
3(50)	$3.5258E+05$	$1.9354E+05$	$5.0059E+05$	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	$3.0046E+05$	279.4	2703.2	1570.9	300000	0
4(50)	84661	17535	$5.0067E+05$	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	$3.0056E+05$	296.09	4501.3	1003.9	$3.00E+05$	0
5(50)	8104.2	3136.2	$5.0057E+05$	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	$1.0035E+05$	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	$1.003E+05$	192.14	20.337	0.076526	$1.00E+05$	0
8(30)	20.701	0.14497	$3.0043E+05$	276.2	20.935	0.036175	$3.00E+05$	0
8(50)	20.924	0.21506	$5.0059E+05$	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	$1.0032E+05$	204.81	12.641	5.2662	$1.00E+05$	0
10(30)	99.106	27.234	$3.0044E+05$	313.97	59.488	20.262	$3.00E+05$	0
10(50)	174.82	41.563	$5.0056E+05$	415	129.28	36.435	$5.00E+05$	0
11(10)	5.9521	1.6896	$1.0033E+05$	193.31	6.1848	1.99	$1.00E+05$	0
11(30)	31.173	2.7352	$3.0053E+05$	285	36.107	3.3108	$3.00E+05$	0
11(50)	59.813	4.4452	$5.007E+05$	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	$3.003E+05$	280.71	4509.1	4094.9	$3.00E+05$	0
12(50)	27771	36755	$5.0048E+05$	429.24	20643	13136	$5.00E+05$	0
13(10)	0.56433	0.42115	$1.0025E+05$	183.96	0.97	0.41621	$1.00E+05$	0
13(30)	1.447	0.45196	$3.004E+05$	392.1	5.27	1.9156	$3.00E+05$	0
13(50)	2.7203	0.75874	$5.0045E+05$	444.33	15.294	5.1243	$5.00E+05$	0
14(10)	3.4857	0.30168	$1.0028E+05$	193.21	3.146	0.30093	$1.00E+05$	0
14(30)	13.11	0.26151	$3.0052E+05$	261.17	13.197	0.28793	$3.00E+05$	0
14(50)	22.768	0.29265	$5.0039E+05$	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	$3.004E+05$	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	$1.0028E+05$	191.28	120.82	14.091	$1.00E+05$	0
16(30)	271.9	147.42	$3.0044E+05$	324.32	202.35	154.3	$3.00E+05$	0
16(50)	223.31	116.25	$5.0064E+05$	423.03	156.62	104.93	$5.00E+05$	0
17(10)	151.56	26.789	$1.0034E+05$	199.97	126.78	21.222	$1.00E+05$	0
17(30)	377.21	142.57	$3.0046E+05$	297.68	216.95	174.71	$3.00E+05$	0
17(50)	381.6	81.206	$5.0075E+05$	367.45	187.63	135.19	$5.00E+05$	0

Table 4: 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	$1.63E+05$	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	$2.72E+05$	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	$2.09E+05$	49555
6(50)	4.8457	13.791	$4.78E+05$	35158	3.8803	12.716	$4.74E+05$	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 5: 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 6: 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

egy 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, (23-10-8) in row 1 column 2 of Table 7, indicates that the HMHH algorithm outperformed SaNSDE 23 times over the benchmark problem set. Ten draws and eight losses were recorded.

Table 7: Hypotheses analysis of PAP, EEA-SLPS, modified AMALGAM-SO, FAUC-Bandit, HMHH, and EIHH1 versus their constituent algorithms.

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

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 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 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 5 and 6 provide examples of this behaviour. Note that SaNSDE significantly outperformed CMAES on this problem.

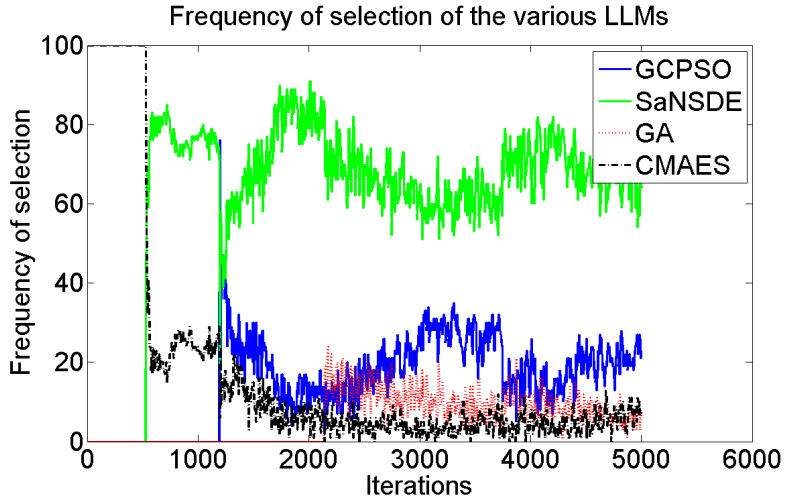

Figure 5: Frequency of use of each of the LLMs in the EIHH1 algorithm on the $F12$ of the CEC 2005 benchmark problem set in 50 dimensions. Frequency of use is determined by the number of entities allocated to the LLM under consideration per iteration. It should be noted that SaNSDE is the best performing algorithm for $F12$ and EIHH1 thus correctly selects SaNSDE over the best overall performing CMAES algorithm.

The various multi-method algorithms were also compared to each other. In Table 8, Mann-Whitney U tests were used to compare each multi-method algorithm to each one of the other multi-method algorithms. The number of times each multi-method algorithm significantly outperformed each one of the other mutli-method algorithms, performed similarly, or was outperformed by another multi-method algorithm, was again recorded and the same "number of wins-draws-losses" format of Table 7 was used. To illustrate, (6-11-34) in row 1 column 2 of Table 8, indicates

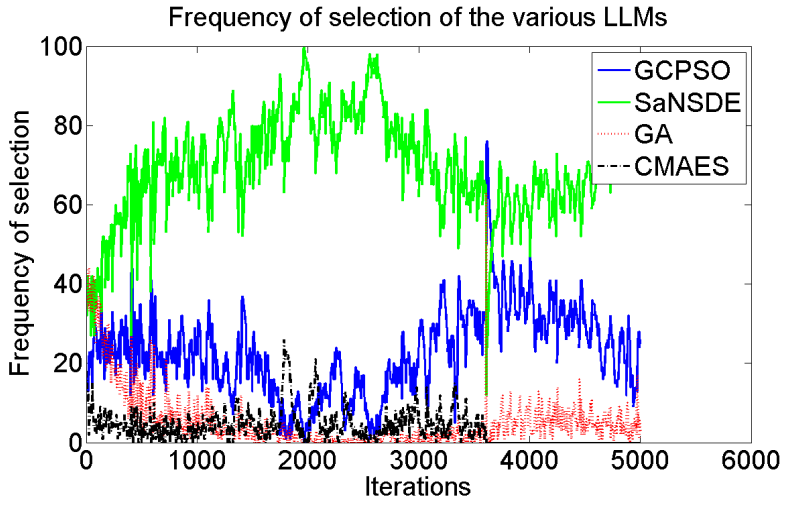

Figure 6: Frequency of use of each of the LLMs in the HMHH algorithm on $F12$ of the CEC 2005 benchmark problem set in 50 dimensions. Frequency of use is determined by the number of entities allocated to the LLM under consideration per iteration. It should be noted that SaNSDE is the best performing algorithm for $F12$ and HMHH thus correctly selects SaNSDE over the best overall performing CMAES algorithm.

that the HMHH algorithm outperformed the EIHH1 algorithm six times over the benchmark problem set. Eleven draws and 34 losses were recorded.

The results indicate that the EIHH1 algorithm is in fact the best performing algorithm out of the six benchmark algorithms evaluated. EIHH1 performed statistically significantly 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.

An analysis of the heuristic space diversity (HSD) [13] of the benchmark algorithms, a metric which indicates the extent of convergence to a single LLM, provides further useful information with regard to algorithm performance. The HSD for $F17$ from the CEC 2005 benchmark problem set in 10, 30 and, 50 dimensions is plotted in Figure 7. Only the graphs of this problem are provided due to space constraints. The graphs of the other problems are similar and are, however, available from the corresponding author on request.

As can be seen in Figure 7, PAP and EEA-SLPS both maintained a constant HSD 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

Table 8: Hypotheses analysis of the multi-method algorithms when compared to each other.

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

same. The advantage of maintaining a high HSD 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 4.3. HSD thus remains constant over time until the population "converges" and then the HSD drops rapidly when algorithms are re-allocated to entities. This rapid decrease in HSD can be seen at around iterations 1100, 1150, 1750, and 2000 for $F17$ in 10 dimensions. 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 $F17$ and some of the other problem graphs.

Even though the modified AMALGAM-SO started off with a significantly high HSD, 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 HSD of the population, resulting in a sporadic HSD over time with no clear strategy with regards to exploration and exploitation of the heuristic space. With regard to the meta-hyper-heuristic algorithms, 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.

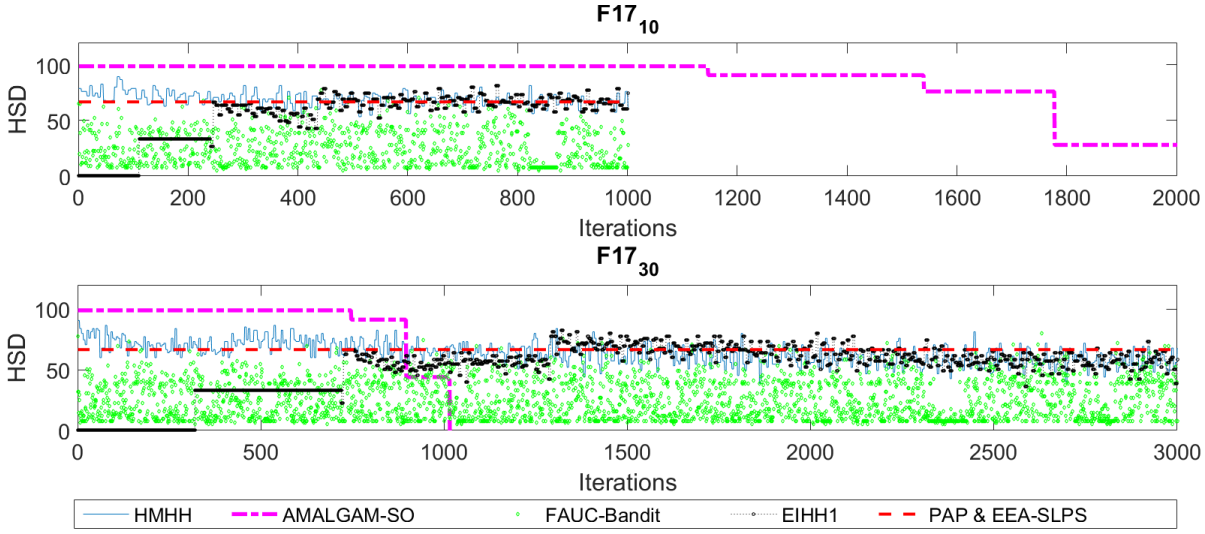

Figure 7: Comparison of HSD (y-axes) versus iterations (x-axes) for PAP, EEA-SLPS, modified AMALGAM-SO, FAUC-Bandit, HMHH, and EIHH1 on $F17$ of the CEC 2005 benchmark problem set in 10 and 30 dimensions.

6. Conclusion

This paper investigated the entity-to-algorithm allocation strategies of six popular multi-method algorithms, namely PAP, EEA-SLPS, a modified AMALGAM-SO, FAUC-Bandit, HMHH and EIHH1. The same set of constituent algorithms were used for each algorithm and a comparison was performed between the multi-method benchmark algorithms to determine the performance contribution of the entity-to-algorithm allocation strategy. The multi-method benchmark algorithms were also compared to each individual constituent algorithm to determine the difference in performance obtained by each of the multi-method algorithms in contrast to simply using a single-method algorithm. EIHH1 was shown to outperform the other five multi-method algorithms and also performed well relative to its constituent algorithms. The second best performing algorithm was PAP, followed, in worsening order, by HMHH, EEA-SLPS, the modified AMALGAM-SO and FAUC-Bandit.

Future work could focus on extending this study to include other memetic computing algorithms and multi-method algorithms which adaptively select between different types of operators within the same algorithm.

Acknowledgments

The authors would like to thank Dr. F Peng and Prof. X Yao for their generous provision of the population based algorithm portfolio source code.

References

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

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

[3] T. G. Dietterich, Ensemble methods in machine learning, in: Multiple classifier systems, Springer, 2000, pp. 1–15.

[4] A. P. Engelbrecht, Heterogeneous particle swarm optimization, in: Swarm Intelligence, Springer, 2010, pp. 191–202.

[5] E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Ozcan, R. Qu, Hyper-heuristics: A survey of the state of the art, Journal of the Operational Research Society 64 (2013) 1695–1724.

[6] J. Grobler, A. P. Engelbrecht, G. Kendall, V. Yadavalli, Multi-method algorithms: Investigating the entity-to-algorithm allocation problem, Proceedings of the 2013 Congress on Evolutionary Computation (2013) 570–577.

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

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

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

[10] Á. Fialho, M. Schoenauer, 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 GECCO 2010 Workshop on Black-Box Optimization Benchmarking (2010) 1535–1542.

[11] J. Grobler, A. P. Engelbrecht, G. Kendall, V. Yadavalli, Investigating the impact of alternative evolutionary selection strategies on multi-method global optimization, Proceedings of the 2011 Congress on Evolutionary Computation (2011) 2337–2344.

[12] J. Grobler, A. P. Engelbrecht, G. Kendall, V. Yadavalli, Heuristic space diversity management in a meta-hyperheuristic framework, Proceedings of the 2014 IEEE Congress on Evolutionary Computation (2014) 1863–1869.

[13] J. Grobler, A. Engelbrecht, G. Kendall, V. Yadavalli, Heuristic space diversity control for improved meta-hyper-heuristic performance, Information Sciences 300 (2015) 49–62.

[14] X. Yao, Y. Liu, L. G., Evolutionary programming made faster, IEEE Transactions on Evolutionary Computation 3 (2) (1999) 82–102.

[15] R. Mallipeddi, P. N. Suganthan, G. K. Pan, M. F. Tasgetiren, Differential evolution algorithm with ensemble of parameters and mutation strategies, Applied soft computing.

[16] A. K. Qin, P. N. Suganthan, Self-adaptive differential evolution algorithm for numerical optimization, Proceedings of the 2005 Congress on Evolutionary Computation 2 (2005) 1785–1791.

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

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

[19] A. Engelbrecht, Scalability of a heterogeneous particle swarm optimizer, Proceedings of the 2011 IEEE Symposium on Swarm Intelligence (SIS) (2011) 1–8.

[20] F. Caraffini, F. Neri, G. Iacca, A. Mol, Parallel memetic structures, Information Sciences 227 (2013) 60–82.

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

[22] P. Korošec, J. Šilc, B. Filipič, The differential ant-stigmergy algorithm, Information Sciences 192 (2012) 82–97.

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

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

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

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

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

[28] E. Yu, P. Suganthan, Ensemble of niching algorithms, Information Sciences 180 (15) (2010) 2815–2833.

[29] A. Caponio, G. L. Cascella, F. Neri, N. Salvatore, 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) (2007) 28–41.

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

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

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

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

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

[35] O. Olorunda, A. P. Engelbrecht, An analysis of heterogeneous cooperative algorithms, Proceedings of the 2009 Congress on Evolutionary Computation (2009) 1562–1569.

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

[37] S. Yuen, C. Chow, 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, ACM, 2013, pp. 567–574.

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

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

[40] Á. Fialho, R. Ros, M. Schoenauer, M. Sebag, Comparison-based adaptive strategy selection with bandits in differential evolution, in: Parallel Problem Solving from Nature, PPSN XI, Springer, 2010, pp. 194–203.

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

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

[43] J. Grobler, The heterogeneous meta-hyper-heuristic: From low level heuristics to low level meta-heuristics, Ph.D. thesis, University of Pretoria (2015).

[44] L. J. Eshelman, Real-coded genetic algorithms and interval-schemata, Foundations of genetic algorithms 2 (1993) 187–202.

[45] F. Van den Bergh, A. Engelbrecht, A new locally convergent particle swarm optimiser, Proceedings of the 2011 International Conference on Systems, Man and Cybernetics 3 (2002) 6–12.

[46] Z. Yang, K. Tang, X. Yao, Self-adaptive differential evolution with neighborhood search, Proceedings of the 2008 Congress on Evolutionary Computation (2008) 1110–1116.

[47] A. Auger, N. Hansen, A restart cma evolution strategy with increasing population size, Proceedings of the 2005 Congress on Evolutionary Computation 2 (2005) 1769–1776.

[48] O. Olorunda, A. Engelbrecht, Measuring exploration/exploitation in particle swarms using swarm diversity, Proceedings of the 2008 IEEE Congress on Evolutionary Computation (2008) 1128–1134.

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

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

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

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