The Entity-to-Algorithm Allocation Problem: Extending the Analysis

Jacomine Grobler
Department of Industrial and Systems Engineering
University of Pretoria and Council for Scientific and Industrial Research
Pretoria, South Africa
Email: jacomine.grobler@gmail.com

Andries P. Engelbrecht
Department of Computer Science
University of Pretoria
Pretoria, South Africa

Graham Kendall[1] and V.S.S. Yadavalli[2]
School of Computer Science
University of Nottingham, UK and Malaysia[1]
Department of Industrial and Systems Engineering
University of Pretoria
Pretoria, South Africa[2]

***Abstract*—This paper extends the investigation into the algorithm selection problem in hyper-heuristics, otherwise referred to as the entity-to-algorithm allocation problem, introduced by Grobler *et al.*. Two newly developed population-based portfolio algorithms (the evolutionary algorithm based on self-adaptive learning population search techniques (EEA-SLPS) and the Multi-EA algorithm) are compared to two meta-hyper-heuristic algorithms. The algorithms are evaluated under similar conditions and the same set of constituent algorithms on a diverse set of floating-point benchmark problems. One of the meta-hyper-heuristics are shown to outperform the other algorithms, with EEA-SLPS coming in a close second.**

I. Introduction

Since the first multi-method algorithms, memetic algorithms, were developed the allocation of solutions or entities to the various algorithms making up the memetic algorithm, has become an important consideration. Therefore, investigating various entity-to-algorithm allocation strategies is important to determine best practice guidelines for multi-method algorithms. Grobler *et al.* [1] initiated such a study, where three multi-method algorithms were compared with regards to their entity-to-algorithm allocation strategy. Over the last year new multi-method algorithms have, however, been proposed and it thus became necessary to extend this work to include additional algorithms.

An analysis of the multi-method literature identified a number of promising multi-method algorithms developed over the last couple of years. The population-based algorithm portfolio [4] and the fitness-based area under the curve-bandit strategy [5] were included in the previous investigation. This paper thus focuses on investigating the entity-to-algorithm allocation strategies of the multiple evolutionary algorithm of Yuen *et al.* [3], the evolutionary algorithm based on self-adaptive learning population search techniques [2], and two versions of the heterogeneous meta-hyper-heuristic (HMHH) [6] algorithm.

One of the main issues in multi-method algorithm literature is the fact that each newly introduced multi-method algorithm utilizes its own set of low level algorithms as selected by the authors. However, since the selection of the set of low level algorithms has an enormous impact on the performance of the multi-method algorithm, it is imperative that different entity-to-algorithm allocation mechanisms should be compared while both use the same set of multi-method algorithms. This practice is currently not adopted in literature, and it is thus impossible to know which high-level strategy is truly better. Herein lies the contribution of this paper, namely providing an unbiased comparison of three different multi-method algorithms to independently evaluate the entity-to-algorithm allocation mechanism.

Algorithm performance was evaluated on a set of varied floating-point benchmark problems. The most promising results were obtained by the exponentially increasing heuristic space diversity (EIHH) algorithm, but EEA-SLPS and the standard HMHH demonstrated similar performance. Good performance, when compared to the multi-method algorithms' constituent algorithms, was also demonstrated.

The rest of the paper is organized as follows: Section II provides a brief overview of the multi-method algorithm literature. Section III describes the multi-method algorithms which were evaluated. The results are documented in Section IV before the paper is concluded in Section V.

II. A brief review of related literature

Multi-method algorithms have appeared in various different domains over the last couple of years. Examples include memetic computation [7], algorithm portfolios [8], algorithm ensembles [9], and hyper-heuristics [10]. This section provides a closer inspection of multi-method literature.

Memetic algorithms (MAs), the algorithmic pairing of a population-based search method with one or more refinement methods [7], can be considered the first multi-method techniques applied in the field of computational intelligence [11]. 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.

Various more complex self-adaptive multi-method algorithms have since been developed. In the DE domain, the intelligent selection of mutation operators and control parameters during optimization is considered in [18], [19],

and [20]. The self-adaptive DE algorithm of Qin and Suganthan [12] 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 [21].

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 of Olorunda and Engelbrecht [13] 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.

Vrugt *et al.*'s highly successful population-based genetic adaptive method for single objective optimization (AMALGAM-SO) [14] 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. This technique performed well when compared to a number of single method EAs on the 2005 IEEE Congress of Evolutionary Computation benchmark problem set [30]. Closer inspection of the algorithm, however, uncovers a large bias towards CMAES. Since it is well-known that CMAES is the best choice among the available algorithms for solving the CEC2005 problems, the question of how well the algorithm will perform without this assistance, remains to be answered.

Peng *et al.* [4] 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. What influence this has on algorithm performance, remains to be determined.

Tang *et al.* [15] recently developed an extended version of PAP, namely PAP based on Estimated Performance Matrix (EPM-PAP) which contains a novel constituent algorithms selection module. EMP-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, HMHH and AMALGAM which attempt to obtain the best possible solution on a specific problem instance.

Another successful adaptive strategy selection mechanism was investigated by Fialho *et al.* [16]. Comparisons of alternative credit assignment methods [17] and strategy selection mechanisms within a differential evolution framework [5] highlighted the superior performance of the fitness-based area under curve bandit (FAUC-Bandit) technique.

Hyper-heuristics [22] 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. Most recent hyper-heuristic algorithms consist of a high level methodology which controls 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. The tabu-search hyper-heuristic of Burke *et al.* [23] and the simulated annealing based hyper-heuristic of Dowsland *et al.* [24] are notable examples.

In summary, a number of state of the art algorithms can be identified from the fields of algorithm portfolios, algorithm ensembles, adaptive operator selection strategies and hyper-heuristics. The FAUC-Bandit method [5] was previously identified as the most promising adaptive operator selection mechanism [1]. Until recently, PAP [4] was the best performing portfolio algorithm and AMALGAM [14] was the best ensemble algorithm. The HMHH algorithm [6] was the only hyper-heuristic selecting from a set of low level meta-heuristics (LLMs) during the optimization run. However, the recent development of two algorithms, namely the multiple evolutionary algorithm (Multi-EA) [3] and the evolutionary algorithm based on self-adaptive learning population search techniques (EEA-SPLS) [2], has led to confusion with regards to the relative performance of the most promising multi-method algorithms. This confusion is largely caused by the use of different low level heuristics in each of the two algorithms. Multi-EA claims superior performance to PAP and AMALGAM and EEA-SLPS claims superior performance to PAP. Grobler *et al.* previously compared the performance of FAUC-Bandit, PAP and HMHH, but it has now become necessary to extend this evaluation to include Multi-EA and EEA-SLPS. The focus of this paper is thus to compare the performance of Multi-EA and EEA-SLPS to two variations of the HMHH algorithm. The next section will provide more detail with regards to the experimental setup.

III. A SELECTION OF MULTI-METHOD ALGORITHMS

To ensure that Multi-EA, EEA-SLPS and the two HMHH algorithms are evaluated under a common baseline, it is important that the same set of constituent algorithms are used by each multi-method algorithm. The constituent algorithms used in this paper were selected based on their diversity profiles as described in [1]:

- A genetic algorithm (GA) with a floating-point representation, tournament selection, blend crossover [25] [13], and self-adaptive gaussian mutation [6].
- The guaranteed convergence particle swarm optimization algorithm (GCPSO) [26].
- The self-adaptive (SaNSDE) algorithm [27].
- The covariance matrix adapting evolutionary strategy algorithm (CMAES) [28].

The algorithm control parameters values listed in Table I were found to work well for the algorithms under study during initial parameter optimization. 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 I. LOW-LEVEL META-HEURISTIC ALGORITHM PARAMETERS.

Parameter	Value used
GCPSO 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 specified in [27].
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 [28].

The four multi-method algorithm frameworks which were selected for comparison are described in the rest of this section.

A. The heterogeneous meta-hyper-heuristic algorithm

The HMHH algorithm consists of a number of algorithmic elements. As indicated in Figure 1, these elements 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, the population of entities is divided equally into a number of subpopulations. These subpopulations are adapted in parallel by the set of LLM algorithms. Each entity is able to access the genetic material of other subpopulations, as if part of a common population of entities. The allocation of entities to LLMs is updated on a dynamic basis throughout the optimization run. The idea is that an intelligent algorithm can be adapted 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. The LLM allocation is maintained for k iterations, while the common parent population is continuously updated with new information and 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.

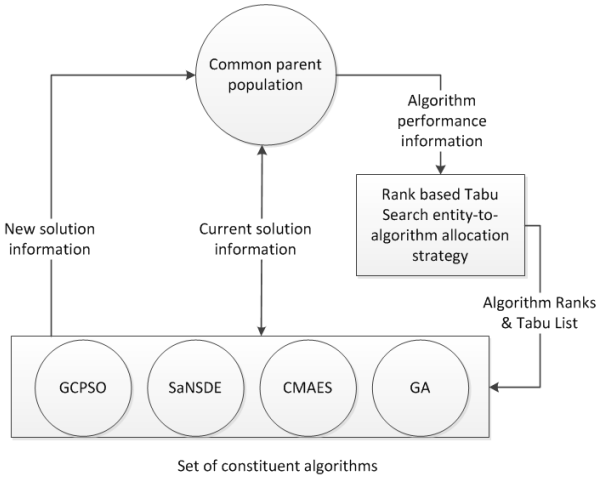

Fig. 1. The heterogeneous meta-hyper-heuristic.

More specifically,

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

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.

The HMHH algorithm was considered promising since each entity can 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, and the role the LLM is playing in the larger algorithm.

With reference to Burke *et al.*'s classification [23], the heterogeneous meta-hyper-heuristic algorithm can be considered a heuristic selection methodology and an online learning perturbation hyper-heuristic. For the sake of completeness, the HMHH pseudocode is provided in Algorithm 1.

A derivative of the HMHH algorithm, the exponentially increasing heuristic space diversity hyper-heuristic algorithm with no *a priori* knowledge (EIHH) [31], was also included in the investigation. The EIHH has a similar structure as the HMHH algorithm, but uses the set of available LLMs to influence the heuristic space diversity of the algorithm. At the start of the optimization run only one LLM is made available to the hyper-heuristic. As the optimization process progresses, additional randomly selected LLMs are made available at predetermined exponential time intervals. Here the hyper-heuristic is forced to move from exploitation to exploration. 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. The EIHH was shown to provide the best results when compared to a number of other heuristic space diversity management

Algorithm 1: The heterogeneous meta-hyper-heuristic.

$t = 0$
Initialize the parent population $\boldsymbol{X}(t+1)$
$A_i(t)$ denotes the index of the LLM applied to entity i at iteration t
for *all entities* $i \in \boldsymbol{X}(t+1)$ **do**
    Randomly select an initial LLM, $A_i(1)$, from the set of LLMs to apply to entity i
end
while *stopping conditions are not satisfied* **do**
    for *all entities* i **do**
        Apply LLM $A_i(t)$ to entity i for k iterations
        $t = t + k$
        Calculate $Q_{\delta m}(t)$ using Equation (1).
    end
    for *all entities* i **do**
        Use $Q_{\delta m}(t)$ as input to select LLM $A_i(t)$ according to the rank-based tabu search entity-to-LLM allocation strategy.
    end
end

strategies and also outperformed the HMHH algorithm so will also be included in the multi-method algorithm evaluation.

B. The multiple evolutionary algorithm

The basis of the multiple evolutionary algorithm (Multi-EA) [3] is a portfolio of promising evolutionary algorithms which are run independently from each other. Throughout the optimization run, the partial convergence curves of the constituent evolutionary algorithms are extrapolated to a common future point in time. These extrapolations are used at each iteration to predict the future performance of each algorithm and determines which algorithm will be used to execute the next iteration of the portfolio. During the development of the algorithm Yuen *et al.* [3] experimented with various extrapolation methods including exponential curves, polynomials and Taylor series. However, a least squares error linear regression method was determined to be the most effective solution. The robustness of the algorithm is further increased by subdividing each partial convergence curve into a number of subcurves, each with their own prediction. All the predictions for a specific algorithm are then considered to be sample points of an unknown distribution and a bootstrap probability distribution is fitted to the points. The final prediction for each algorithm is then sampled from the distribution associated with its predictive measure. The main idea is to have multiple linear regression models each with an equal probability of selection for each algorithm at each iteration of the portfolio. For the sake of completeness, the pseudocode of the Multi-EA algorithm is provided in Algorithm 2.

One of the main differences between Multi-EA and some of the other popular portfolio methods is that each constituent algorithm is executed independently from all other algorithms and no information exchange mechanism is used between the different subpopulations. The reason provided for this decision is that offspring from different

Algorithm 2: The multiple evolutionary algorithm [3].

Let n_a be the number of constituent algorithms available for selection.
Let M_n be the number of entities assigned to algorithm n.
for *all algorithms* $m = 1$ *to* n_a **do**
    Run algorithm m until there is a change in fitness. Let α_m be the number of generations that algorithm m has run.
end
while *stopping conditions are not satisfied* **do**
    Compute the nearest common future point t_{min}, where $t_{min} = \max((\alpha_1 + 1)M_1, \ldots, (\alpha_{n_a} + 1)M_{n_a})$.
    for *all algorithms* $m = 1$ *to* n_a **do**
        Construct convergence curve $C_m = \{(j, f_m(j)) | j = 1, \ldots, \alpha_m\}$.
        Construct sub-curves $\{C_m(1), \ldots, C_m(\alpha_i - 1)\}$.
        for *each sub-curve* $C_m(l)$ **do**
            Perform a least square line fit to obtain line parameters a and b.
            Predict the fitness at the smallest future point, $pf_m(t_{min}, l)$.
        end
        Use the $\alpha_m - 1$ sample points of $pf_m(t_{min}, l)$ where $l = 1, \ldots, \alpha_m - 1$, to construct a bootstrap probability distribution $bpd_m(t_{min})$.
        Sample $bpd_m(t_{min})$ to obtain $pf_m(t_{min})$.
    end
    Choose the algorithm with index which has the best predicted performance according to $\boldsymbol{pf}(t_{min})$.
    Run it for one generation so that $\alpha_m = \alpha_m + 1$.
    Record the best solution found thus far by the portfolio.
end

algorithms is not able to mislead each other. However, this also results in the different algorithms being unable to learn from each other which influences the benefit of using multiple algorithms simultaneously. Since the computational budget is divided between different algorithms without the algorithms benefiting from each other's learning, it is debatable whether the portfolio will be greater than the sum of its parts.

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

Similar to PAP [4], the evolutionary algorithm based on self-adaptive learning population search techniques (EEA-SLPS) [2] 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 [2] 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 3.

Algorithm 3: The evolutionary algorithm based on self-adaptive learning population search techniques [2].

```
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 stopping conditions are not 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
```

IV. Empirical evaluation

The four multi-method algorithms were evaluated on the first 14 problems of the 2005 IEEE Congress of Evolutionary Computation benchmark problem set, which enables algorithm performance evaluation on both unimodal and multimodal functions and includes various expanded and hybridized problems, some with noisy fitness functions [30]. The control parameters of the three multi-method algorithms are used "as-is" as defined by the original authors and are specified in Table II.

TABLE II. HMHH, Multi-EA, and EEA-SLPS parameters.

Parameter	Value used
Common algorithm parameters	
Population size (n_s)	100
Maximum number of iterations (I_{max}), where n_x denote the number of dimensions.	$100n_x$
Multi-EA 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
HMHH	
Number of iterations between re-allocation (k)	5
Size of tabu list	3

The results of the multi-method algorithm framework comparison is presented in Table V, where the results for each algorithm were recorded over 30 independent simulation runs. 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. Where the global optimum could not be found within the maximum number of iterations, the final solution at I_{max}, denoted by FFV, was recorded.

Statistical tests were also used to evaluate the significance of the results. The results in Table IV were obtained by comparing each dimension-problem-combination of the algorithm under evaluation, to all of the dimension-problem-combinations of the other algorithms. For every comparison, a Mann-Whitney U test at 95% significance was performed (using the two sets of 30 data points of the two algorithms under comparison) and if the first algorithm statistically significantly outperformed the second algorithm, a win was recorded. If no statistical difference could be observed a draw was recorded. If the second algorithm outperformed the first strategy, a loss was recorded for the first algorithm. The total number of wins, draws and losses were then recorded for all combinations of the algorithm under evaluation. As an example, (5-19-4) in row 2 column 1, indicates that the EIHH algorithm significantly outperformed the HMHH algorithm five times over the benchmark problem set. Furthermore, 19 draws and four losses were recorded.

TABLE III. Hypotheses analysis of alternative multi-method algorithms.

	HMHH	EIHH	EEA-SLPS
HMHH	NA	$4-19-5$	$11-8-9$
EIHH	$5-19-4$	NA	$6-16-6$
EEA-SLPS	$9-8-11$	$6-16-6$	NA
Multi-EA	$3-4-21$	$3-1-24$	$2-3-23$
	Multi-EA	**TOTAL**	
HMHH	$21-4-3$	$36-3-17$	
EIHH	$24-1-3$	$35-36-13$	
EEA-SLPS	$23-3-2$	$38-27-19$	
Multi-EA	NA	$8-8-68$	

From the results it can be seen that the HMHH, EIHH, and EEA-SLPS algorithms all scored a high number of wins. Overall the EIHH algorithm is the best performing algorithm due to the largest difference between its number of wins and losses. The performance of the Multi-EA algorithm on the benchmark set is, however, disappointing. This worse than expected performance should be investigated further, however, the lack of an information exchange mechanism is a strong suspect for the poor performance.

In an attempt to further verify the performance of the multi-method algorithms, each multi-method algorithm was also compared under similar conditions to its constituent algorithms. The results are recorded in Table VI. In Table IV Mann-Whitney U tests were used to compare the performance of each constituent algorithm to the different multi-method algorithm. The same number of wins-draws-losses format of Table IV was used.

TABLE IV. Hypotheses analysis of the various algorithms versus their constituent algorithms.

Algorithm	HMHH	EIHH	EEA-SLPS	Multi-EA
CMAES	0-3-25	4-2-22	4-2-22	2-2-24
SaNSDE	17-2-9	16-8-4	12-12-4	5-0-23
GA	22-3-3	23-2-3	23-4-1	4-5-19
GCPSO	20-1-7	20-3-5	19-3-6	8-3-17
TOTAL	55-18-39	63-15-34	58-21-33	19-10-83

TABLE V. RESULTS OF THE EVALUATION OF ALTERNATIVE MULTI-METHOD FRAMEWORKS ON THE 2005 IEEE CEC BENCHMARK PROBLEM SET.

Prob(Dims)	HMHH				EEA-SLPS				Multi-EA			
	FFV		# FEs		FFV		# FEs		FFV		# FEs	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
1(10)	$1.00E-06$	0	11870	538.93	$1.00E-06$	0	13923	517.74	2014.3	918.43	$1.00E+05$	0
1(30)	$1.00E-06$	0	52983	2723.3	$1.00E-06$	0	35297	2974.8	13116	4690.2	$3.00E+05$	0
2(10)	$1.00E-06$	0	14013	1246.7	$1.00E-06$	0	18553	858.12	3813.7	1256.8	$1.00E+05$	0
2(30)	$1.00E-06$	0	90727	15876	$1.00E-06$	0	90507	2064.3	55613	12540	$3.00E+05$	0
3(10)	$1.00E-06$	0	22750	3003	$1.00E-06$	0	45283	2295.7	$1.29E+07$	$4.09E+06$	$1.00E+05$	0
3(30)	295.35	998.74	$2.79E+05$	27782	$1.00E-06$	0	$2.85E+05$	5400.7	$2.56E+08$	$8.58E+07$	$3.00E+05$	0
4(10)	$1.00E-06$	0	15853	1341.8	$1.00E-06$	0	19927	1128.9	3820.4	1945.1	$1.00E+05$	0
4(30)	$1.00E-06$	0	$1.50E+05$	27962	2161.6	3328.6	$2.75E+05$	54905	64308	11193	$3.00E+05$	0
5(10)	$1.00E-06$	0	17110	883.31	$1.00E-06$	0	32470	7258.9	967.18	371.63	$1.00E+05$	0
5(30)	174.5	278.17	$3.00E+05$	0	894.25	777.66	$3.00E+05$	0	12144	2627.4	$3.00E+05$	0
6(10)	0	0	33417	7388.8	0	0	49337	8334.1	$1.50E+08$	$7.09E+07$	$1.00E+05$	0
6(30)	0.13267	0.72665	$2.43E+05$	33935	4.6447	13.185	$2.87E+05$	30467	$5.35E+09$	$3.12E+09$	$3.00E+05$	0
7(10)	0.162	0.13632	$1.00E+05$	0	0.005	0.0073108	74563	36835	0.0023333	0.0062606	35988	42732
7(30)	0.0036667	0.0080872	$1.30E+05$	$1.13E+05$	0	0	$1.18E+05$	93406	0.00033333	0.0018257	55202	97805
8(10)	20.06	0.086681	$1.00E+05$	0	20.116	0.10344	$1.00E+05$	0	20.312	0.074652	$1.00E+05$	0
8(30)	20.169	0.12041	$3.00E+05$	0	20.281	0.1898	$3.00E+05$	0	20.942	0.050561	$3.00E+05$	0
9(10)	0.005	0.0050855	43320	19202	0.069667	0.36084	45953	20620	26.159	11.6	$1.00E+05$	0
9(30)	2.3763	1.3964	$2.98E+05$	10079	1.255	1.6865	$2.05E+05$	92270	112.17	28.185	$3.00E+05$	0
10(10)	15.556	9.1746	$1.00E+05$	0	8.8837	5.4366	$1.00E+05$	0	47.404	13.574	$1.00E+05$	0
10(30)	55.768	19.838	$3.00E+05$	0	52.687	23.623	$3.00E+05$	0	247.25	64.791	$3.00E+05$	0
11(10)	5.0464	2.2525	97283	14880	4.1765	1.5439	$1.00E+05$	0	4.1868	1.6632	$1.00E+05$	0
11(30)	24.378	5.1224	$3.00E+05$	0	20.661	2.6114	$3.00E+05$	0	20.147	3.3931	$3.00E+05$	0
12(10)	302.86	546.06	69543	39317	92.191	330.6	69873	28012	18616	10679	$1.00E+05$	0
12(30)	2611.5	3622.3	$2.95E+05$	20162	7803.4	10925	$3.00E+05$	0	$3.08E+05$	$1.07E+05$	$3.00E+05$	0
13(10)	0.43267	0.16282	99050	5203.4	0.35533	0.15822	$1.00E+05$	0	170.15	164.66	$1.00E+05$	15.833
13(30)	2.0187	0.44262	$3.00E+05$	0	1.719	0.8862	$3.00E+05$	0	18462	11116	$3.00E+05$	0
14(10)	3.6397	0.29122	$1.00E+05$	0	3.4057	0.32298	$1.00E+05$	0	3.6133	0.32996	$1.00E+05$	0
14(30)	13.14	0.44011	$3.00E+05$	0	12.89	0.49124	$3.00E+05$	0	13.391	0.18989	$3.00E+05$	0

TABLE VI. RESULTS OF THE LOW-LEVEL META-HEURISTIC ALGORITHMS ON THE 2005 IEEE CEC BENCHMARK PROBLEM SET.

Prob (Dims)	CMAES				SaNSDE				GA				GCPSO			
	FFV		# FEs		FFV		# FEs		FFV		# FEs		FFV		# FEs	
	μ	σ	μ	σ	μ	σ	μ	σ	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	$1.00E-06$	0	18557	1582.4	138.4	6.8073	$1.00E+05$	0
1(30)	$1.00E-06$	0	19110	447.48	$1.00E-06$	0	38973	839.51	$1.00E-06$	0	99297	4860.4	231.81	29.491	$3.00E+05$	0
2(10)	$1.00E-06$	0	9156.7	286.1	$1.00E-06$	0	38377	2088.3	1.0873	1.54	$1.00E+05$	0	544.1	1.5915	$1.00E+05$	0
2(30)	$1.00E-06$	0	26783	739.1	$1.00E-06$	0	$2.8926E+05$	19917	446.67	228.69	$3.00E+05$	0	567	3.0169	$3.00E+05$	0
3(10)	$1.00E-06$	0	13320	379.11	$1.00E-06$	0	46337	2059.9	$9.7613E+05$	$1.0468E+06$	$1.00E+05$	0	970.96	1411.8	99177	4509.6
3(30)	$1.00E-06$	0	61173	1387.4	$1.7946E+05$	$1.1489E+05$	$3.00E+05$	0	$5.9255E+06$	$2.6545E+06$	$3.00E+05$	0	10543	8705.2	$3.00E+05$	0
4(10)	$1.00E-06$	0	9590	283.27	$1.00E-06$	0	42767	2605.2	380.19	510.25	$1.00E+05$	0	320.69	0.20813	$1.00E+05$	0
4(30)	$1.00E-06$	0	29357	570.35	195.19	186.51	$3.00E+05$	0	15946	7051.1	$3.00E+05$	0	325.45	1.5416	$3.00E+05$	0
5(10)	$1.00E-06$	0	17433	546.04	$1.00E-06$	0	36280	1098.7	869.56	1241.9	$1.00E+05$	0	12.858	0.28301	$1.00E+05$	0
5(30)	$1.00E-06$	0	$1.1465E+05$	3960.1	978.56	425.28	$3.00E+05$	0	12887	3070	$3.00E+05$	0	22.356	0.52122	$3.00E+05$	0
6(10)	$6.67E-04$	$2.54E-03$	18950	744.52	0.26533	1.0098	52987	13213	346.91	1274	$1.00E+05$	0	840.01	$4.6252E-13$	26480	799.31
6(30)	0.13267	0.72665	$1.2018E+05$	37870	0.859	1.7187	$2.7619E+05$	33794	1281.6	2540.4	$3.00E+05$	0	840.01	$4.6252E-13$	69423	1540
7(10)	1267	$4.6252E-13$	$1.00E+05$	0	0.041667	0.025608	99920	438.18	1267	$4.6252E-13$	$1.00E+05$	0	270.01	$1.7345E-13$	38517	1131.4
7(30)	4696.3	$2.7751E-12$	$3.00E+05$	0	0.012	0.017889	$1.8329E+05$	$1.188E+05$	4696.3	$2.7751E-12$	$3.00E+05$	0	270.01	$1.7345E-13$	$1.998E+05$	11139
8(10)	20.312	0.11271	$1.00E+05$	0	20.345	0.077626	$1.00E+05$	0	20.197	0.12287	$1.00E+05$	0	64679	60668	$1.00E+05$	0
8(30)	20.892	0.17459	$3.00E+05$	0	20.886	0.041728	$3.00E+05$	0	20.368	0.094336	$3.00E+05$	0	$6.6785E+05$	$3.1267E+05$	$3.00E+05$	0
9(10)	1.9457	1.5105	88203	30601	0	0	43690	2247.8	0.0033333	0.0047946	18983	6020.3	120.01	$7.2269E-14$	43280	3701.2
9(30)	39.564	6.4543	$3.00E+05$	0	0	0	$1.0608E+05$	4250.5	0.0043333	0.0050401	$1.568E+05$	39913	2228.2	1403.9	$3.00E+05$	0
10(10)	1.647	1.1767	90947	27625	5.646	1.367	$1.00E+05$	0	31.174	12.988	$1.00E+05$	0	19.99	0	41123	2053
10(30)	9.391	3.2817	$3.00E+05$	0	31.503	5.4713	$3.00E+05$	0	122.94	32.782	$3.00E+05$	0	4699.3	974.88	$3.00E+05$	0
11(10)	1.2989	1.3647	30310	10496	5.3248	1.0486	$1.00E+05$	0	7.5876	1.1929	$1.00E+05$	0	306.49	21.172	99730	1478.9
11(30)	9.0255	3.0546	$3.00E+05$	0	27.5	1.6478	$3.00E+05$	0	30.659	3.4532	$3.00E+05$	0	325.66	46.216	$3.00E+05$	0
12(10)	1546.1	2735.5	70053	43090	24.701	30.914	81150	23456	873.7	1566.8	$1.00E+05$	0	1547	0.061026	$1.00E+05$	0
12(30)	20324	19261	$3.00E+05$	0	10594	2868.8	$3.00E+05$	0	15214	11587	$3.00E+05$	0	4978.1	10.078	$3.00E+05$	0
13(10)	0.897	0.25323	$1.00E+05$	0	0.34233	0.056366	$1.00E+05$	0	0.43967	0.16951	$1.00E+05$	0	10.237	0.097519	$1.00E+05$	0
13(30)	3.179	0.56064	$3.00E+05$	0	1.2977	0.1177	$3.00E+05$	0	1.657	0.47256	$3.00E+05$	0	10.803	0.11511	$3.00E+05$	0
14(10)	2.5847	0.53024	$1.00E+05$	0	3.2743	0.25292	$1.00E+05$	0	3.6913	0.31421	$1.00E+05$	0	27.558	1.4475	96700	12585
14(30)	10.394	0.8103	$3.00E+05$	0	12.707	0.23694	$3.00E+05$	0	13.092	0.31847	$3.00E+05$	0	8.6763	7.1708	$3.00E+05$	0

Similar to the results obtained in [6], it can be seen that the HMHH algorithm again performed statistically significantly better a large number of times when compared to three of its four constituent algorithms. CMAES performed better than the HMHH when solving unimodal problems, but the HMHH performance improved in comparison with CMAES as problem size and complexity increased. An inspection of the algorithm ranks does, however, indicate that the HMHH algorithm was able to identify CMAES as the best performing algorithm and bias the search towards CMAES. The inefficiency of the HMHH algorithm is then understandable since computational resources are required to first "learn" which algorithm is the best algorithm for the problem at hand.

V. Conclusion

This paper has investigated four alternative multi-method algorithms using the same set of constituent algorithms. The heterogenous meta-hyper-heuristic algorithm was shown to outperform the Multi-EA algorithm and EEA-SLPS approach and also compared favourably to its constituent algorithms.

Future work could focus on extending this study to a larger set of benchmark problems of different sizes. A more in-depth analysis of the impact of various entity-to-algorithm allocation features on multi-method algorithm performance could also be useful as well as a more detailed study into the mechanisms driving the poor performance of Multi-EA.

References

[1] 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 Congress on Evolutionary Computation*, pp. 570–577, 2013.

[2] 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*, vol. 231, pp. 329–346, 2014.

[3] S. Y. Yuen, C.K. Chow, and X. Zhang. "Which algorithm should I choose at any point of the search: an evolutionary portfolio approach," *Proceedings of the 2013 Conference on Genetic and Evolutionary Computation*, pp. 567–574, 2013.

[4] F. Peng, K. Tang, G. Chen, and X. Yao,"Population-Based Algorithm Portfolios for Numerical Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 5, pp. 782–800, 2010.

[5] A. Fialho, R. Ros, M. Schoenauer, and M. Sebag, "Comparison-based Adaptive Strategy Selection in Differential Evolution," *Proceedings of the 2011 Genetic and Evolutionary Computation Conference*, 2011.

[6] 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 2011 IEEE Congress on Evolutionary Computation*, pp. 2337-2344, 2011.

[7] X. Chen, Y. Ong, M. Lim, and K. Tan, "A multi-facet survey on memetic computation," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 5, pp. 591–607, 2011.

[8] C. P. Gomes and B. Selman, "Algorithm portfolios," *Artificial Intelligence*, vol. 126, no. 1–2, pp. 43–62, 2001.

[9] T. G. Dietterich, "Ensemble methods in machine learning," *Lecture notes in computer science*, pp. 1–15, 2000.

[10] E. K. Burke, G. Kendall, and E. Soubeiga, "A Tabu-Search Hyperheuristic for Timetabling and Rostering," *Journal of Heuristics*, vol. 9, no. 6, pp. 451–470, 2003.

[11] W. E. Hart, N. Krasnogor, and J. E. Smith, "Recent Advances in Memetic Algorithms," *Springer-Verlag*, 2005.

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

[13] O. Olorunda and A. P. Engelbrecht, "An Analysis of Heterogeneous Cooperative Algorithms," *Proceedings of the 2009 IEEE Congress on Evolutionary Computation*, pp. 1562–1569, 2009.

[14] 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*, vol. 13, no. 2, pp. 243–259, 2009.

[15] K. Tang, F. Peng, G. Chen, and X. Yao. "Population-based Algorithm Portfolios with automated constituent algorithms selection," *Information Sciences*, vol. 279, pp. 94–104, 2014.

[16] 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 GECCO 2010 Workshop on Black-Box Optimization Benchmarking*, 2010.

[17] W. Gong, A. Fialho, and Z. Cai, "Adaptive strategy selection in differential evolution," *Proceedings of the 2010 Genetic and Evolutionary Computation Conference*, 2010.

[18] R. Mallipeddi, P. N. Suganthan, Q. K. Pan, and M. F. Tasgetiren "Differential evolution algorithm with ensemble of parameters and mutation strategies," *Applied Soft Computing*, vol. 11, pp. 1679–1696, 2011.

[19] 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*, In press.

[20] J. Tvrdik "Modifications of Differential Evolution with Composite Trial Vector Generation Strategies," *Soft Computing Models in Industrial and Environmental Applications Advances in Intelligent Systems and Computing*, vol. 188, pp. 113-122, 2013.

[21] A.P. Engelbrecht "Heterogeneous particle swarm optimization," *Swarm Intelligence*, pp. 191-202, 2010.

[22] E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and R. Qu, "Hyper-heuristics: A survey of the state of the art," tech. rep., University of Nottingham, 2010.

[23] E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and J. R. Woodward, "A Classification of Hyper-heuristic Approaches," *International Series in Operations Research and Management Science*, In M. Gendreau and J-Y Potvin (Eds.), Springer (in press).

[24] K. A. Dowsland, E. Soubeiga, and E. K. Burke, "A simulated annealing based hyperheuristic for determining shipper sizes for storage and transportation," *European Journal of Operational Research*, vol. 179, pp. 759–774, 2007.

[25] L. J. Eshelman and J. D. Schaffer, "Real-coded genetic algorithms and interval schemata," In D. Whitley, editor, *Foundations of Genetic Algorithms*, vol. 2, pp. 187–202, 1993.

[26] 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*, vol. 3, pp. 6–12, 2002.

[27] Z. Yang, K. Tang, and X. Yao, "Self-adaptive Differential Evolution with Neighbourhood Search," *Proceedings of the 2008 IEEE Congress on Evolutionary Computation*, pp. 1110-1116, 2008.

[28] A. Auger, and N. Hansen, "A Restart CMA evolution strategy With increasing population size," *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, pp. 1769-1776, 2005.

[29] D. Hadka and P. Reed "Borg: An auto-adaptive Many-Objective Evolutionary Computing Framework," *Evolutionary Computation*, 2012.

[30] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y. P. Chen, A. Auger, and S. Tiwari. "Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization," *Nanyang Technological University and Kanpur Genetic Algorithms Laboratory*, 2005, KanGAL Report Number 2005005.

[31] 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 Congress on Evolutionary Computation*, 2014.