Alternative hyper-heuristic strategies for multi-method global optimization

Jacomine Grobler, *Student Member, IEEE,* Andries P. Engelbrecht, *Senior Member, IEEE,* Graham Kendall, *Member, IEEE,* V.S.S. Yadavalli

***Abstract*— The purpose of this paper is to investigate the use of meta-heuristics as low-level heuristics in a hyper-heuristic framework. A novel multi-method hyper-heuristic algorithm which makes use of a number of common meta-heuristics is presented. Algorithm performance is evaluated on a diverse set of real parameter benchmark problems and meaningful conclusions are drawn with respect to the selection of alternative low-level heuristics and the acceptance of the obtained solutions within the proposed multi-method meta-heuristic approach.**

I. Introduction

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

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

This paper introduces a novel heterogeneous meta-hyper-heuristic algorithm consisting of a high-level hyper-heuristic framework which is used to select the most appropriate solution strategy to apply to a specific candidate solution at a specific time during an optimization run. The impact of alternative strategies for selecting the low-level heuristics and accepting the obtained solutions within the context of the proposed framework, is also investigated. Algorithm performance is evaluated against a set of varied real parameter benchmarking problems and promising results are obtained with respect to solution quality and algorithm robustness.

The significance of the paper lies in the fact that, to the best of the authors' knowledge, this paper describes the first investigation into the use of alternative meta-heuristics as low-level heuristics in a hyper-heuristic framework. Furthermore, the investigation into suitable selection and acceptance strategies also provided useful information regarding future algorithm refinement.

The rest of the paper is organized as follows: Section II provides an overview of existing literature. Section III introduces the heterogeneous meta-hyper-heuristic (HMHH) while Section IV describes the selection and acceptance strategies which were evaluated. The results are documented in Section V before the paper is concluded in Section VI.

II. A brief review of related literature

Both hyper-heuristics and multi-method algorithms were considered relevant to this research and are briefly introduced in this section.

A. Hyper-heuristics

Burke *et al.* [3] define a hyper-heuristic as "*a search method or learning mechanism for selecting or generating heuristics to solve computational search problems*". The basic idea is not only to obtain an appropriate solution for a specific problem, but rather to focus on automating the development of the method used to obtain an appropriate solution. To achieve this aim, a search space of low-level heuristics is defined, upon which a high level search strategy operates. This approach is in contrast to the "traditional" meta-heuristic strategy of searching directly through a search space of candidate solutions. From this comparison, it is clear that hyper-heuristic implementations benefit from increased generality – a valuable attribute considering the specialist resources required for the development of advanced AI-based

Jacomine Grobler is with Denel Dynamics Pty (Ltd). as well as the Department of Industrial and Systems Engineering at the University of Pretoria, South Africa (corresponding author to provide e-mail: jacomine.grobler@gmail.com).

Andries P. Engelbrecht is with the Department of Computer Science at the University of Pretoria, South Africa.

Graham Kendall is with the School of Computer Science at the University of Nottingham, UK.

V.S.S. Yadavalli is with the Department of Industrial and Systems Engineering at the University of Pretoria, South Africa.

978-1-4244-8126-2/10/$26.00 ©2010 IEEE

algorithms as well as the problem-dependent nature of most meta-heuristic algorithm implementations.

Burke *et al.* [3] also provide a unified classification of recent hyper-heuristic research (Figure 1). Hyper-heuristics are classified according to the nature of the heuristic search space as well as the source of feedback during learning. With respect to the nature of the heuristic search space, two popular approaches, namely heuristic selection and heuristic generation, can be identified. Heuristic selection focuses on combining pre-existing heuristics in one higher-level search strategy, whereas heuristic generation is concerned with generating completely new heuristics consisting of basic components or building blocks of existing heuristics. On a more detailed level, construction heuristics attempt to construct a single good candidate solution through the intelligent application of different low-level heuristics. Perturbation heuristics use one low-level heuristic to construct a complete solution, but repetitively apply low-level heuristics in a local search approach to obtain better and better candidate solutions.

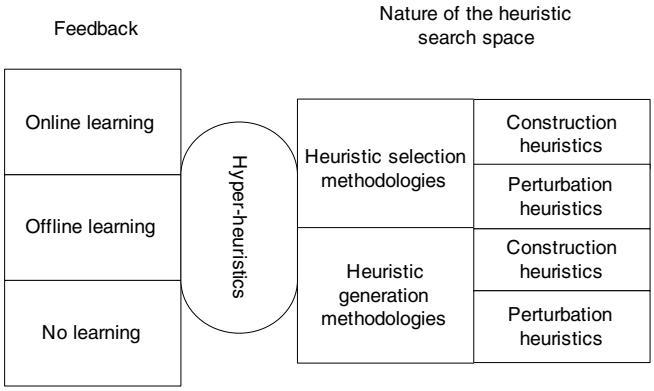

Fig. 1. The heterogeneous meta-hyper-heuristic [3].

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

From a brief analysis of recent hyper-heuristic research based on the above classification, two interesting observations were made. Firstly, very few hyper-heuristic implementations make use of population-based algorithms, i.e. where a population of incumbent solutions are stored over time. Secondly, even though the use of meta-heuristic algorithms have been used many times as a higher-level search strategy in a hyper-heuristic framework, far fewer implementations focus on the use of meta-heuristics as low-level search strategies in a hyper-heuristic framework. The idea of combining different meta-heuristic algorithms into a single algorithm, capitalizing on the strengths of each individual technique is, however, not new. The next section introduces a number of algorithms from existing literature which fit the above description.

B. Heterogeneous algorithms

Memetic algorithms can be considered to be the first multi-method techniques applied in the field of computational intelligence [8]. Here 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.

Recently, more complex self-adaptive multi-method algorithms have been developed. The self-adaptive differential evolution algorithm of Qin and Suganthan [14] makes use of different differential evolution (DE) learning strategies which are weighted based on previous algorithm success. Barkat *et al.* [1] designed an agent-based memetic algorithm for constrained optimization and the patented ISSOP software package uses several optimization strategies in parallel within a neural learning and adaption system.

Vrugt *et al.*'s population-based genetic adaptive method for single objective optimization (AMALGAM-SO) [20] was developed after the success of the multi-objective AMALGAM algorithm. 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. 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 evolutionary algorithms on the 2005 IEEE Congress of Evolutionary Computation benchmark problem set [17].

Another algorithm which inspired the development of the heterogeneous-meta-hyper-heuristic (HMHH) algorithm is the heterogenous cooperative algorithm of Olorunda and Engelbrecht [13]. Traditionally, cooperative algorithms are multi-population techniques where problem variables are distributed over a number of sub-populations to be optimized separately. A solution is constructed by combining the best solution obtained by each subpopulation. The heterogeneous cooperative algorithm makes use of different evolutionary algorithms to update each of the sub-populations, 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 evolutionary algorithms.

III. The Heterogeneous Meta-Hyper-Heuristic Algorithm

With reference to Burke et al.'s classification, the heterogeneous meta-hyper-heuristic (HMHH) algorithm can be considered a heuristic selection methodology and an online learning perturbation hyper-heuristic. The basic idea is to use a high-level search strategy to intelligently control the use of a number of meta-heuristic sub-algorithms over a single optimization run of a real parameter optimization problem. It is hoped that an intelligent algorithm can be evolved which

selects the appropriate meta-heuristic at each K^{th} iteration to be applied to each individual entity in the population of candidate solutions.

The rest of this section will be used to introduce the generic algorithm framework in more detail before the individual sub-algorithms are described.

A. The HMHH algorithm framework

The HMHH algorithm can be described most effectively by means of a diagram. With reference to Figure 2, the HMHH algorithm consists of a common population of entities which each represent a candidate solution, a set of low-level meta-heuristic sub-algorithms and an acceptance and selection strategy. Seven commonly used meta-heuristic sub-algorithms were used as the set of low-level heuristics. The idea of the selection strategy is to attempt to select the most appropriate meta-heuristic sub-algorithm at each K iterations on an individual basis for each entity. The selected meta-heuristic sub-algorithm is then applied to each entity within the context of the common parent population before the acceptance strategy determines whether the new candidate solution obtained should replace the corresponding parent solution within the parent population. This strategy was considered promising since each entity can use unique meta-heuristic operators, helpful for dealing with the specific search space characteristics it is encountering and the role it is playing in the optimization process. In addition, the use of global information stored in the common parent population enables the propagation of information between different entities.

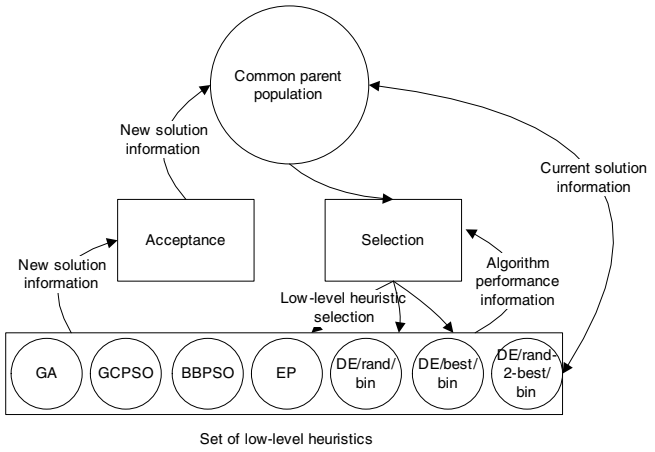

Fig. 2. The heterogeneous meta-hyper-heuristic.

In a multi-method optimization algorithm, the flow of information between different elements of the algorithm becomes important. The main information flows in the HMHH algorithm is also illustrated in Figure 2. Information regarding current local and global best solutions as well as current entity parameter values is transferred from the parent population to the meta-heuristic sub-algorithm at each low-level heuristic selection phase. Information regarding meta-heuristic sub-algorithm performance is fed back to the selection strategy, whereas solution quality information and current parameter values are provided as input to the acceptance strategy. Where the new solution is to be accepted, the current solution information is transferred back to the parent population to complete the loop.

A high-level overview of the HMHH algorithm is provided in Algorithm 1.

Algorithm 1: The heterogeneous meta-hyper-heuristic.

```
Initialize the parent population X
for All entities i ∈ X do
    Randomly select an initial algorithm j_i(1) from the
    set of meta-heuristic sub-algorithms to apply to
    entity i
end
t = 1
K = 25
while The stopping conditions are not met do
    for All entities i do
        Apply sub-algorithm j_i(t) to entity i for K
        iterations
        Record performance statistics for sub-algorithm
        j_i(t) on entity i
    end
    for All entities i do
        Use appropriate acceptance strategy to
        determine whether X should be updated i.e.
        x(t) = x(t − K)
    end
    for All entities i do
        Use appropriate selection strategy to select
        sub-algorithm j_i(t + K)
    end
    t = t + K
end
```

B. Subalgorithms of the MMHH algorithm

The meta-heuristic sub-algorithms employed as low-level heuristics in this paper consists of different variations of genetic algorithms (GAs), differential evolution (DE) and particle swarm optimization (PSO) algorithms. The rest of this section provides more detail with respect to the actual implementation of each sub-algorithm.

1) Genetic algorithms: Olorunda and Engelbrecht [13] used a GA with floating-point representation, tournament selection, blend crossover [7] and gaussian mutation in their heterogeneous cooperative algorithm implementation. The component-wise nature of the blend crossover and Gaussian mutation is thought to be well suited for fine-tuning candidate solutions, a desirable attribute for a sub-algorithm in the context of the HMHH algorithm. Based on this observation, a similar approach has been followed in this paper, with the addition of a self-adaptive mutation operator. For all dimensions, j, if $r \sim U(0,1) \leq p_m$, the probability of mutation is given as

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

with $\varsigma_{ij}(t+1) = \varsigma_{ij}(t+1)e^{\tau_1 N(0,1)+\tau_2 N(0,1)}$, $\tau_1 = \frac{1}{\sqrt{2\sqrt{D}}}$ and $\tau_2 = \frac{1}{\sqrt{2D}}$, where D denotes the number of dimensions [6].

A stand-alone evolutionary program was also implemented as an additional sub-algorithm based on the mutation operator defined in Equation (1), no crossover operator, and a $(\mu + \lambda)ES$ selection strategy.

2) Particle swarm optimization: Two variations on the standard PSO algorithm were implemented, namely the guaranteed convergence PSO algorithm of Van den Bergh and Engelbrecht [19] and the bare-bones PSO algorithm.

The GCPSO algorithm requires that different velocity and displacement updates, respectively indicated by Equation (2) and Equation (3), are applied to the global best particle:

$$\begin{aligned} v_{\tau j}(t+1) =& - x_{\tau j}(t) + x_j^*(t) + w v_{\tau j}(t) + \\ & \rho(t)(1 - 2r_j(t)) \end{aligned} \quad (2)$$

$$x_{\tau j}(t+1) = x_j^*(t) + w v_{\tau j}(t) + \rho(t)(1 - 2r_j(t)) \quad (3)$$

This in fact forces the *gbest* particle into a random search around the global best position. The size of the search space is adjusted on the basis of the number of consecutive successes or failures of the particle, where success is defined as an improvement in the objective function value.

The bare-bones PSO algorithm attempts to guide all particles to a weighted average position between the personal best and neighbourhood best positions. The velocity of dimension j of particle i at time $t+1$ is sampled from a normal distribution:

$$v_{ij}(t+1) = N(\frac{c_1 \hat{x}_{ij}(t) + c_2 x^* ij(t)}{c_1 + c_2}, \sigma), \quad (4)$$

where

$$\sigma = |\hat{x}_{ij}(t) - x^* ij(t)|. \quad (5)$$

3) Differential evolution: Three variations on the classical DE algorithm [16], namely *DE/rand/bin*, *DE/best/bin* and, *DE/rand-to-best/bin* were implemented as sub-algorithms in this paper. *DE/rand/bin* selects $\boldsymbol{x}_{r_1}(t)$ randomly from the previous population and thereby maintains diversity for longer, whereas *DE/best/bin* selects $\boldsymbol{x}_{r_1}(t)$ as the population member with the lowest fitness function and subsequently obtains faster convergence [16]. *DE/rand-to-best/bin* aims to address the limitations of *DE/best/bin* in terms of potential premature convergence. A linearly decreasing value ($\gamma \in (0, 1)$) is incorporated into the equation used to calculate the base vector. This ensures that more emphasis is placed on random base vector selection at the start of the optimization run when population diversity is important. Towards the end, the *DE/best/bin* strategy is emphasized when convergence to the best solution is desirable. The adjusted target vector equation becomes

$$\begin{aligned} T_{ij}(t) =& \gamma x_{\tau j}(t) + (1 - \gamma) x_{i_1 j}(t) + \\ & F(x_{i_2 j}(t) - x_{i_3 j}(t)), \end{aligned} \quad (6)$$

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

IV. Investigation of alternative selection and acceptance strategies

Bilgin *et al.* [2] states that a perturbation hyper-heuristic consists of two critical elements: the low-level heuristic selection strategy and the move acceptance strategy. This section describes the alternative selection and acceptance strategies investigated in this paper.

A. Alternative selection strategies

- **Random:** The random selection strategy randomly assigns a meta-heuristic sub-algorithm to an entity during heuristic selection. No memory of previous good performance is retained and no learning is attempted.
- **Performance based:** This strategy was inspired by Vrugt *et al's* AMALGAM-SO algorithm [20]. The selection strategy selects a new meta-heuristic sub-algorithm based on previous algorithm success with reference to each individual entity. This is achieved by using a roulette wheel-based selection operator which ensures that sub-algorithms which have performed well previously have a higher probability of being selected again. The probability $p_{ij}(t)$ of selecting sub-algorithm l for updating entity i at time t can be calculated as:

$$\begin{aligned} p_{il}(t) =& \sum_{k=1}^{t} \sum_{j=1}^{D} y_{ilk} \\ & |f(x_{ij}(k)) - f(x_{ij}(k+1))|, \end{aligned} \quad (7)$$

  where $y_{ilk} = 1$ if sub-algorithm l was applied to entity i during iteration k and 0 otherwise.
- **Tabu-search based:** The tabu-search based heuristic selection strategy makes use of the heuristic selection mechanism developed by Burke *et al.* for timetabling problems [4]. Here sub-algorithms compete with each other based on the principles of reinforcement learning and a tabu list is maintained for each entity to ensure that poor performing sub-algorithms are not repeatedly applied to the same entity. The selection strategy ranks the sub-algorithms based on their performance with regards to a specific entity and always selects the highest ranking non-tabu sub-algorithm (refer to Algorithm 2).

B. Alternative acceptance strategies

- **Deterministic:** All solutions, whether improving or non-improving, are accepted.
- **Stochastic:** This strategy makes use of simulated annealing concepts to accept non-improving solutions with decreasing probability as the optimization process progresses [5]. All improving moves are accepted, whereas non-improving moves are accepted with probability:

$$p = e^{\frac{-(f(x_{ij}(t+1)) - f(x_{ij}(t)))}{T}}, \quad (8)$$

  where T is the simulated annealing temperature parameter.

Algorithm 2: Burke *et al's* tabu-search hyper-heuristic [4].

Let r_{il} be the rank of sub-algorithm l with respect to entity i.
for *All entities* i **do**
  Apply sub-algorithm l^* to entity i, where l^* is the highest ranking non-tabu sub-algorithm w.r.t. entity i
  if $\sum_{j=1}^{D} y_{ilk}|f(x_{ij}(k)) - f(x_{ij}(k+1))| > 0$ **then**
    $r_{il} = r_{il} + 1$
  else
    if $\sum_{j=1}^{D} y_{ilk}|f(x_{ij}(k)) - f(x_{ij}(k+1))| = 0$ **then**
      $r_{il} = r_{il} - 1$
      Insert sub-algorithm l into $\boldsymbol{T}_i$
    else
      $r_{il} = r_{il} - 1$
      $\boldsymbol{T}_i = \boldsymbol{\Phi}$
    end
  end
end

V. Empirical results

To investigate the impact of alternative selection and acceptance strategies on multi-population meta-hyper-heuristic performance, an empirical comparison between the various selection and acceptance strategies defined in the previous section were conducted. The best performing HMHH algorithms were then compared to four of the HMHH meta-heuristic sub-algorithms which were implemented on a stand-alone basis. This section describes the benchmark problem set, algorithm control parameters and other experimental conditions before the results of the algorithm comparison are presented and discussed.

For performance evaluation purposes, five benchmark problems were selected from literature so that algorithm performance could be evaluated on both uni-modal and multi-modal functions. The general difficulty associated with optimization as well as the inclusion of noise in one of the objective functions also played a role in the selection of appropriate benchmark problems. The following benchmark functions were used:

- The shifted sphere function (CEC 2005 benchmark problem 1) [17]

$$F_1(\boldsymbol{x}) = \sum_{i=1}^{D_1} z_i^2 + f\_bias_1, \boldsymbol{z} = \boldsymbol{x} - \boldsymbol{o} \tag{9}$$

  where $\boldsymbol{o}$ is the shifted global optimum, $F_1(\boldsymbol{x}^*) = f\_bias_1 = -450$, and $-100 \leq x_i \leq 100$. Throughout the rest of this section D_n refers to the dimensionality of problem n.
- The rastrigin function

$$F_2(\boldsymbol{x}) = \sum_{i=1}^{D_2} (x_i^2 - 10\cos(2\pi x_i) + 10D_2 \tag{10}$$

  with $-100 \leq x_i \leq 100$ and the global optimum, $F_2(\boldsymbol{x}^*)$, located at $F_2(0, \ldots, 0) = 0$.
- The salomon function

$$F_3(\boldsymbol{x}) = -\cos(2\pi\sqrt{\sum_{i=1}^{D_3} x_i^2}) + 0.1\sqrt{\sum_{i=1}^{D_3} x_i^2} + 1 \tag{11}$$

  with $-100 \leq x_i \leq 100$ and the global optimum, $F_3(\boldsymbol{x}^*)$, located at $F_3(0, \ldots, 0) = 0$.
- The shifted rotated Weierstrass function (CEC 2005 benchmark problem 11)

$$\begin{aligned} F_4(\boldsymbol{x}) =& \sum_{i=1}^{D_4}(\sum_{k=0}^{k_{max}}[a^k \cos(2\pi b^k(z_i + 0.5))]) \\ & - D_4 \sum_{k=0}^{k_{max}}[a^k \cos(2\pi b^k .0.5)] + f\_bias_4, \\ \boldsymbol{z} =& (\boldsymbol{x} - \boldsymbol{o})\boldsymbol{M} \end{aligned} \tag{12}$$

  where $\boldsymbol{o}$ is the shifted global optimum, $\mathbf{M}$ is a linear transformation matrix (condition number $= 5$), $F_4(\boldsymbol{x}^*) = f\_bias_4 = 90$, and $-0.5 \leq x_i \leq 0.5$.
- The rotated hybrid composition function with noise in the fitness function (CEC 2005 benchmark problem 17: a hybridization of the Rastrigin, Weierstrass, Griewank, Ackley, and sphere functions).

$$\begin{aligned} F_5(\boldsymbol{x}) =& (\sum_{i=1}^{n_f}\{w_i[f_i'(\frac{(\boldsymbol{x} - \boldsymbol{o}_i)}{\alpha_i}\boldsymbol{M}_i) + bias_i]\} + \\ & f_{bias_{i+1}})(1 + 0.2|N(0,1)|) + f_{bias_5} \end{aligned} \tag{13}$$

  where $\boldsymbol{o}_i$ is the shifted global optimum, $\boldsymbol{M}_i$ is the linear transformation matrix, α_i is the stretch or compression constant, w_i is the weight, and $bias_i$ is the bias associated with function i. n_f denotes the number of functions which are hybridized and $F_5(\boldsymbol{x}^*) = f\_bias_5 = 120$, and $-5 \leq x_i \leq 5$.

The algorithm control parameters listed in Table I were used for all simulations in this section. These parameter values were found to work well for the algorithms under study during previous research by the authors. The number of individuals in the population is denoted by n_s and $m \longrightarrow n$ indicates that the associated parameter is decreased linearly from m to n over 95% of the maximum number of iterations, I_{max}.

The actual results of the hyper-heuristic comparison are recorded in Tables II and III, where the results for each strategy were recorded over 30 independent simulation runs. Throughout the rest of this section, μ and σ respectively denote the mean and standard deviation associated with the corresponding performance measure, $CI_{0.05}$ is a 95%

TABLE I
HMHH ALGORITHM PARAMETERS.

Parameter	Value used
General parameters	
n_s	100
PSO-specific parameters	
c_1	$2.0 \longrightarrow 0.7$
c_2	$0.7 \longrightarrow 2.0$
w	$0.9 \longrightarrow 0.4$
DE-specific parameters	
p_r	$0.75 \longrightarrow 0.25$
F	$0.75 \longrightarrow 0.125$
GA-specific parameters	
p_c	$0.6 \longrightarrow 0.4$
p_m	0.1
α	0.5
N_t	13
SA-specific parameters	
T	$100 \longrightarrow 0$
TS-specific parameters	
$Tabu_{max}$	3

confidence interval on μ, and FE denotes the number of function evaluations which was needed to reach the global optimum. Where the global optimum could not be found within the maximum number of iterations, the final solution at I_{max} was recorded. Alg(n,m) denotes the HMHH algorithm implemented with selection strategy n and acceptance strategy m, where $n = 1, 2$ and 3, respectively denotes the random, performance based and Tabu search based selection strategies. $m = 1$ refers to the deterministic acceptance strategy and $m = 2$ refers to the stochastic acceptance strategy.

With respect to the performance of the two acceptance strategies, it would be expected that the SA-based selection strategy would outperform the deterministic acceptance strategy for complex, multi-modal problems. This expectation results mainly from the algorithms ability to explore more at the start of the optimization run and exploit more towards the end of the run. The use of a stochastic acceptance strategy significantly increases the exploration ability of the algorithm allowing for slower convergence and a greater chance of escaping from local optima. The deterministic acceptance strategy may, however, lead to faster convergence on simpler problems.

The performance-based selection strategy has the greatest selective pressure, followed by the Tabu-search based strategy and the random strategy. It would make sense that faster convergence would be obtained by the performance-based selection strategy on simpler problems whereas the random and tabu-search based strategies would be better for multi-modal problems with difficult to search objective and parameter spaces.

Pairwise t-tests were used to analyze the results obtained. In Tables II and III statistically significant performance improvements are highlighted. For example, for the ten dimensional shifted spherical function, Alg(3,1) and Alg(3,2) outperforms Alg(1,1), Alg(1,2), Alg(2,1), and Alg(2,2), by a statistically significant margin.

From this analysis it became clear that Alg(3,2) seemed to be the best performing algorithm for the smaller-sized problems, whereas Alg(3,1) performed the best for larger sized problems. It was also encouraging to note that as problem difficulty increased, the different variations of the HMHH algorithm seemed to perform equally well.

In addition to the selection and acceptance strategy comparisons which were performed, results of four meta-heuristic sub-algorithms implemented on a stand-alone basis are also provided in Tables IV and V. Pairwise t-tests comparing the HMHH algorithms to these four composing sub-algorithms indicated that for all problems, except the shifted spherical function, the HMHH algorithms were able to either outperform or obtain at least as good an answer as the best performing sub-algorithm.

TABLE IV
INVESTIGATING SUB-ALGORITHM PERFORMANCE ON SMALLER SIZED PROBLEMS ($D_1, D_4, D_5 = 10$ AND $D_2, D_3 = 30$).

Problem	Measure	GCPSO	BBPSO	DE	GA
F_1	$\mu(f(\boldsymbol{x}^*))$	-449.10	-449.24	-449.15	-449.22
	σ	0.12	0.16	0.12	0.19
	$CI_{0.05}$	± 0.04	± 0.06	± 0.05	± 0.07
	$\mu(FE)$	345.23	57.17	198.73	30.97
	σ	135.50	4.49	8.06	12.23
	$CI_{0.05}$	± 50.56	± 1.67	± 3.01	± 4.56
F_2	μ	45.14	29.62	23.00	6.19
	σ	11.30	7.97	4.55	2.78
	$CI_{0.05}$	± 4.22	± 2.97	± 1.70	± 1.04
F_3	μ	0.36	0.24	0.18	0.44
	σ	0.07	0.05	0.04	0.10
	$CI_{0.05}$	± 0.03	± 0.02	± 0.01	± 0.04
F_4	μ	96.85	95.59	97.72	95.55
	σ	1.28	1.62	0.49	1.41
	$CI_{0.05}$	± 0.48	± 0.61	± 0.18	± 0.53

TABLE V
INVESTIGATING SUB-ALGORITHM PERFORMANCE ON LARGE SIZED PROBLEMS ($D_1, D_4, D_5 = 50$ AND $D_2, D_3 = 100$).

Problem	Measure	GCPSO	BBPSO	DE	GA
F_1	$\mu(f(\boldsymbol{x}^*))$	-449.04	-449.06	-449.07	-449.05
	σ	0.04	0.04	0.06	0.04
	$CI_{0.05}$	± 0.01	± 0.02	± 0.02	± 0.02
	$\mu(FE)$	1012.27	960.20	1307.87	212.70
	σ	25.20	72.14	6.79	25.16
	$CI_{0.05}$	± 9.40	± 26.92	± 2.53	± 9.39
F_2	μ	189.88	461.14	263.86	712.24
	σ	28.57	155.84	20.76	71.03
	$CI_{0.05}$	± 10.66	± 58.15	± 7.75	± 26.51
F_3	μ	3.12	2.71	1.08	2.97
	σ	0.63	2.48	0.10	0.50
	$CI_{0.05}$	± 0.24	± 0.92	± 0.04	± 0.19
F_4	μ	161.78	162.17	162.80	146.48
	σ	1.86	2.43	1.42	12.24
	$CI_{0.05}$	± 0.69	± 0.91	± 0.53	± 4.57

VI. CONCLUSION

This paper investigated the use of meta-heuristics as low-level heuristics in a hyper-heuristic framework. A novel het-

erogeneous meta-hyper-heuristic algorithm was introduced and the impact of alternative acceptance and selection strategies within the context of the proposed framework was investigated. Initial experimental results indicated that the HMHH algorithm was able to obtain promising results in terms of solution quality and algorithm robustness.

Significant future research opportunities exist in improving the performance of the HMHH algorithm through more detailed analyses of the effect of the alternative strategies on exploration of the search space and exploitation of promising solutions. Including additional low-level heuristics, for example a local search algorithm, into the HMHH algorithm may also prove useful.

REFERENCES

[1] A. S. S. M. U. Barkat, R. Sarker, D. Comfort, and C. Lokan, "An agent based memetic algorithm (AMA) for solving constrained optimization problems," *Proceedings of the 2007 IEEE Congress on Evolutionary Computation*, pp. 999-1006, 2007.

[2] B. Bilgin, E. Ozcan, and E. E. Korkmaz, "An experimental study on hyper-heuristics and final exam scheduling," *Proceedings of the 2006 International Conference on the Practice and Theory of Automated Timetabling*, pp. 123-140, 2006.

[3] 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).

[4] 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.

[5] 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.

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

[7] 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.

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

[9] J. Holland, "Outline for a logical theory of adaptive systems," *Journal of the ACM*, vol. 9, no. 3, pp. 297-314, 1962.

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

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

[12] J. Kennedy and R. Mendes, "Population structure and particle performance," *Proceedings of the IEEE Congress on Evolutionary Computation*, vol. 2, pp. 1671–1676, 2002.

[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] 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.

[15] R. L. Rardin, *Optimization in Operations Research*. Springer, 2004.

[16] R. Storn and K. Price, "Differential evolution — a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, pp. 341–359, 1997.

[17] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y. P. Chen, A. Auger, S. Tiwari, "Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization," *Technical Report, Nanyang Technological University, Singapore and KanGAL Report Number 2005005 (Kanpur Genetic Algorithms Laboratory, IIT Kanpur)*, 2005.

[18] F. van den Bergh, *An analysis of particle swarm optimizers*, PhD Thesis, Department of Computer Science, University of Pretoria, Pretoria, South Africa, 2002.

[19] 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.

[20] 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.

TABLE II

INVESTIGATING ALTERNATIVE SELECTION AND ACCEPTANCE STRATEGIES ON SMALLER SIZED PROBLEMS ($D_1, D_4, D_5 = 10$ AND $D_2, D_3 = 30$).

Problem	**Measure**		**Alg(1,1)**	**Alg(1,2)**	**Alg(2,1)**	**Alg(2,2)**	**Alg(3,1)**	**Alg(3,2)**
Shifted spherical	$f(\boldsymbol{x}^*)$	μ	-449.2	-449.17	-449.19	-449.18	-449.21	-449.2
		σ	0.18	0.14	0.14	0.12	0.16	0.18
		$CI_{0.05}$	$\pm$0.067	$\pm$0.052	$\pm$0.052	$\pm$0.045	$\pm$0.060	$\pm$0.067
	FE	μ	6557	6160	6783	5953	**5843**	**5627**
	(Tolerance $= f(\boldsymbol{x}*) \pm 1 \times 10^{-6}$)	σ	781	739	864	619	461	649
		$CI_{0.05}$	$\pm$291.64	$\pm$275.95	$\pm$322.63	$\pm$231.14	$\pm$172.14	$\pm$242.35
Rastrigin	$f(\boldsymbol{x}^*)$	μ	0.07	1.09	4.84	5.74	0.17	**0**
		σ	0.25	1.29	3.12	4.45	0.64	0
		$CI_{0.05}$	$\pm$0.093	$\pm$0.482	$\pm$1.165	$\pm$2.143	$\pm$0.239	$\pm$0
	FE	μ	249903	374963	492537	500000	209800	208923
	(Tolerance $= f(\boldsymbol{x}*) \pm 0$)	σ	84783	147242	40878	0	84235	46482
		$CI_{0.05}$	$\pm$31659	$\pm$54982	$\pm$15264.38	$\pm$0	$\pm$31454	$\pm$17357
Salomon	$f(\boldsymbol{x}^*)$	μ	**0.1**	**0.1**	0.3	0.31	**0.1**	**0.1**
	(Tolerance $= f(\boldsymbol{x}*) \pm 0$)	σ	0.02	0	0.07	0.07	0	0
		$CI_{0.05}$	$\pm$0.007	$\pm$0	$\pm$0.026	$\pm$0.026	$\pm$0	$\pm$0
Shifted Rotated	$f(\boldsymbol{x}^*)$	μ	97.93	97.31	**96.68**	**95.87**	97.77	**96.94**
Weierstrass Function	(Tolerance $= f(\boldsymbol{x}*) \pm 0.01$)	σ	1.82	1.77	1.76	1.47	1.75	1.91
		$CI_{0.05}$	$\pm$0.680	$\pm$0.661	$\pm$0.657	$\pm$0.549	$\pm$0.653	$\pm$0.713
Rotated hybrid	$f(\boldsymbol{x}^*)$	μ	**290**	**280.64**	**287.57**	**274.32**	**275.92**	**275.21**
composition function	(Tolerance $= f(\boldsymbol{x}*) \pm 0.01$)	σ	38.32	35.69	46.57	37.09	31.94	24.16
with noise in fitness		$CI_{0.05}$	$\pm$14.309	$\pm$13.327	$\pm$17.390	$\pm$13.850	$\pm$11.927	$\pm$9.022

TABLE III

INVESTIGATING ALTERNATIVE SELECTION AND ACCEPTANCE STRATEGIES ON LARGE SIZED PROBLEMS ($D_1, D_4, D_5 = 50$ AND $D_2, D_3 = 100$).

Problem	**Measure**		**Alg(1,1)**	**Alg(1,2)**	**Alg(2,1)**	**Alg(2,2)**	**Alg(3,1)**	**Alg(3,2)**
Shifted spherical	$f(\boldsymbol{x}^*)$	μ	-449.05	-449.05	-449.05	-449.03	-449.06	-449.06
		σ	0.04	0.06	0.04	0.03	0.06	0.04
		$CI_{0.05}$	$\pm$0.015	$\pm$0.022	$\pm$0.015	$\pm$0.011	$\pm$0.022	$\pm$0.015
	FE	μ	90333	128790	**53270**	98070	62260	156207
	(Tolerance $= f(\boldsymbol{x}*) \pm 1 \times 10^{-6}$)	σ	7455	12511	6376	10918	4377	18423
		$CI_{0.05}$	$\pm$2783.79	$\pm$4671.77	$\pm$2380.88	$\pm$4076.92	$\pm$1634.43	$\pm$6879.39
Rastrigin	$f(\boldsymbol{x}^*)$	μ	**72.14**	**55.85**	168.11	245.49	**54.92**	472.42
	(Tolerance $= f(\boldsymbol{x}*) \pm 0$)	σ	41.5	12.17	23.21	44.8	19.76	133.86
		$CI_{0.05}$	$\pm$15.5	$\pm$4.54	$\pm$8.67	$\pm$16.73	$\pm$7.38	$\pm$49.99
Salomon	$f(\boldsymbol{x}^*)$	μ	0.19	0.18	0.99	1.06	**0.1**	**0.1**
	(Tolerance $= f(\boldsymbol{x}*) \pm 0$)	σ	0.03	0.04	0.14	0.13	0.02	0
		$CI_{0.05}$	$\pm$0.011	$\pm$0.015	$\pm$0.052	$\pm$0.049	$\pm$0.007	$\pm$0
Shifted Rotated	$f(\boldsymbol{x}^*)$	μ	157.15	157.77	159.96	**152.98**	162.37	159.10
Weierstrass Function	(Tolerance $= f(\boldsymbol{x}*) \pm 0.01$)	σ	2.93	4.34	4.71	5.42	2.13	3.93
		$CI_{0.05}$	$\pm$1.094	$\pm$1.621	$\pm$1.759	$\pm$2.024	$\pm$0.795	$\pm$1.468
Rotated hybrid	$f(\boldsymbol{x}^*)$	μ	**384.56**	**356.96**	466.71	**432.62**	**425.30**	463.61
composition function	(Tolerance $= f(\boldsymbol{x}*) \pm 0.01$)	σ	101.15	90.61	143.48	118.28	106.93	125.44
with noise in fitness		$CI_{0.05}$	$\pm$37.77	$\pm$33.83	$\pm$53.58	$\pm$44.17	$\pm$39.93	$\pm$46.84