HSEDA: A Heuristic Selection Approach Based on Estimation of Distribution Algorithm for the Travelling Thief Problem

Marcella S. R. Martins
Federal University of Technology - Paraná (UTFPR)
marcella@utfpr.com.br

Mohamed El Yafrani
LRIT URAC 29, Faculty of Science
Mohammed V University in Rabat
m.elyafrani@gmail.com

Myriam R. B. S. Delgado
Federal University of Technology - Paraná (UTFPR)
myriamdelg@utfpr.edu.br

Markus Wagner
Optimisation and Logistics
The University of Adelaide
markus.wagner@adelaide.edu.au

Belaïd Ahiod
LRIT URAC 29, Faculty of Science
Mohammed V University in Rabat
ahiod@fsr.ac.ma

Ricardo Lüders
Federal University of Technology - Paraná (UTFPR)
luders@utfpr.edu.br

ABSTRACT

Hyper-heuristics are high-level search techniques which improve the performance of heuristics operating at a higher heuristic level. Usually, these techniques automatically generate or select new simpler components based on the feedback received during the search. Estimation of Distribution Algorithms (EDAs) have been applied as hyper-heuristics, using a probabilistic distribution model to extract and represent interactions between heuristics and its low-level components to provide high-valued problem solutions. In this paper, we consider an EDA-based hyper-heuristic framework which encompasses a Heuristic Selection approach aiming to find best combinations of different known heuristics. A surrogate assisted model evaluates the new heuristic combinations sampled by the EDA probabilistic model using an approximation function. We compare our proposed approach named Heuristic Selection based on Estimation of Distribution Algorithm (HSEDA) with three state-of-the-art algorithms for the Travelling Thief Problem (TTP). The experimental results show that the approach is competitive, outperforming the other algorithms on most of the medium-sized TTP instances considered in this paper.

CCS CONCEPTS

•**Mathematics of computing** → **Combinatoric problems;** *Probabilistic representations;* •**Computing methodologies** → **Search methodologies;**

KEYWORDS

Heuristic selection, Estimation of Distribution Algorithms, Travelling Thief Problem, Multi-component problems

ACM Reference format:
Marcella S. R. Martins, Mohamed El Yafrani, Myriam R. B. S. Delgado, Markus Wagner, Belaïd Ahiod, and Ricardo Lüders. 2017. HSEDA: A Heuristic Selection Approach Based on Estimation of Distribution Algorithm for the Travelling Thief Problem. In *Proceedings of GECCO '17, Berlin, Germany, July 15-19, 2017,* 8 pages.
DOI: http://dx.doi.org/10.1145/3071178.3071235

GECCO '17, Berlin, Germany

DOI: http://dx.doi.org/10.1145/3071178.3071235

1 INTRODUCTION

Nowadays, a large number of meta-heuristics have been developed for efficiently solving optimization problems [10, 20]. However, meta-heuristics still present some challenging issues regarding their implementation and maintenance.

Due to the fact that simple heuristics are easier to implement and maintain, they have been employed instead of more sophisticated meta-heuristics based systems [23]. These heuristic methods can generate high quality solutions in a reasonable time budget. However, there are still some challenges such as the difficulty of adapting sophisticated meta-heuristics or efficiently combining simple heuristics to solve a complex problem.

Hyper-heuristics are high-level search techniques developed to automate through other (meta)heuristics or machine learning techniques the heuristic design process, selecting heuristics, generally called by Low-Level Heuristics (LLHs), or generating new ones through low level components of heuristics [4, 25].

Many works automatically adapt heuristic methods with hyper-heuristics. In [5] the authors use a heuristic selection approach on automated university course and exam timetabling. The approach memorizes the heuristics that have worked well in previous similar situations (like in off-line learning mode) and retrieves them for solving the addressed problem, providing good results and insights about that learning process for particular timetabling situations. In [22], a random iterative graph based hyper-heuristic produces a set of heuristic combinations. The approach was evaluated on benchmark exam timetabling and graph colouring problems and demonstrated good results.

Estimation of Distribution Algorithms (EDAs) [18] are strategies widely used in evolutionary optimization, and have been recently applied to the field of hyper-heuristics [23]. The main idea of EDAs is to extract and represent, using a probabilistic distribution model, the regularities shared by a subset of high-quality problem solutions. New solutions are then sampled by the model biasing the search to areas where optimal solutions are more likely to be found [11].

The proposed approach uses EDAs to evolve sequences or combinations of well known LLHs aiming to find a good heuristic set

specific for the problem instance at hand. The approach considers the probabilistic distribution of heuristics used at different stages within the fittest sequences. In the rest of this paper, we will refer to our approach as HSEDA, which stands for Heuristic Selection based on an Estimation of Distribution Algorithm.

Heuristic selection using EDAs has been previously discussed in [28], where the authors presented a framework for solving discrete dynamic environment problems, and recently, in [23], in a framework called EDA-HH.

HSEDA is designed and evaluated for the Travelling Thief Problem (TTP), a relatively new NP-hard benchmark problem [2]. TTP is an optimization problem that provides interdependent subproblems, which is a problem aspect often encountered in real-world applications [8].

Many heuristics have been proposed to deal with the TTP components separately, like local search techniques and meta-heuristic adaptations. In [15], the authors propose an off-line heuristic generation approach based on Genetic Programming for the TTP. However, despite the existence of multiple LLHs in the literature, we are not aware of any heuristic selection approach particulary designed for TTP, or for other multi-component problems.

In order to evaluate the efficiency of the HSEDA on this context, in this paper we consider a subset of medium-sized instances of the TTP library. Moreover, a Radial Basis Function Network (RBFN) is used as a surrogate to approximate the fitness computation for individuals sampled from the EDA model. HSEDA performance is compared with three of the best known state-of-the-art algorithms. The results show that the proposed approach is promising as it achieves very competitive results on the addressed instances.

The remainder of this paper is organized as follows. Section 2 briefly presents the use of hyper-heuristics in combinatorial problems, introduces the basic concepts of EDAs and RBFN, and presents the TTP problem. Section 3 describes our proposed approach. Section 4 describes the conducted experiments and Section 5 discusses the results. Finally, conclusions and future directions are presented in Section 6.

2 BACKGROUND

In this section, we revisit the basic concepts associated with EDAs in the context of hyper-heuristics. Moreover, we briefly present a Radial Basis Function Network (RBFN) surrogate model and introduce the TTP problem.

2.1 Estimation of distribution algorithms

EDA is considered a meta-heuristic approach that produces an offspring population by sampling a distribution, normally estimated by a Probabilistic Graphical Model (PGM). Bayesian networks are considered the most prominent PGMs and are often used for modeling multinomial data with discrete variables [11]. These models are also capable of capturing multivariate interactions between variables.

Each variable is a node associated with a conditional probability distribution of its values given different value settings of its parents. Mathematically, a Bayesian network with directed edges encodes a joint probability distribution which can be expressed using Equation 1:

$$p(\mathbf{X}) = \prod_{i=0}^{n-1} p(X_i | Pa_{X_i}) \quad (1)$$

where $\mathbf{X} = (X_0, ..., X_{n-1})$ is the vector of variables, Pa_{X_i} is the set of parents of X_i in the network (i.e., the set of nodes from which there exists an edge to X_i) and $p(X_i|Pa_{X_i})$ is the conditional probability of X_i given Pa_{X_i}. This distribution can be used to generate new candidate solutions using the marginal and conditional probabilities in a modeled data set.

Aiming to estimate the network structure, several algorithms can be used such as simple greedy algorithms, hill-climbing heuristics, and evolutionary algorithms [11]. In this work we adopt the *K2* algorithm, a greedy local search technique that applies the *K2* metric [7]. *K2* starts by assuming that a given node does not have parents. Then, at each step, it gradually adds the edges that increase the scoring metric the most until no further improvement is possible.

EDAs have been applied as a high-level search technique within a hyper-heuristic context in order to solve a number of optimization problems [23].

In [26] the author compares different hyper-heuristics methodologies including a Bayesian heuristic approach which determines the probability distribution of each heuristic based on its historical performance. This method was applied to a variety of discrete problems and showed promising results.

Uludağ et al. [27] proposed *HH-PBIL2*, a framework joining EDAs and hyper-heuristics for solving discrete dynamic environment problems. This approach uses multi-population combining off-line and online learning to deal with random and cyclic dynamic environments. The results show a good performance over different types of dynamic environments.

More recently, Qu et al. [23] presented a constructive EDA that searches for combinations of heuristics from a given set of LLHs based on non-domain-specific knowledge. This EDA's high-level search methodology can automatically select appropriate heuristics in different problem solving situations. The probability distribution of LLHs during the search process is used to evaluate their effectiveness aiming to facilitate more intelligent hyper-heuristic methods. The results demonstrated the generality for different variants of exam timetabling problems.

2.2 On the use of RBFNs as a surrogate model

Many researchers focus on the use of surrogate models within EDAs. Surrogate models are mathematical models usually used to approximate an expensive objective function, providing a computationally cheaper alternative function to evaluate solutions [13]. Statistics and machine learning techniques can be used to build surrogate models with approximated functions from data-points, like multi-variate regression, supervised learning and neural networks [17].

RBFN [3] is a shallow artificial neural network, with a three layers feedforward architecture that uses radial basis functions as activation functions at the hidden nodes. This technique is used to provide an alternative approximation method for surrogate modeling. It aims at building function approximations of the form

of Equation 2.

$$y(\mathbf{x}) = \omega_0 + \sum_{i=1}^{N} \omega_i \phi(||\mathbf{x} - \mathbf{x_i^p}||) \quad (2)$$

where $y(\mathbf{x})$ represents the output layer of the network, i.e., a linear combination of radial basis functions considering a set of N observed data points (training points), and $\mathbf{x}$ denotes a multi-dimensional untried point in M space.

Training data are supplied to the network in the form of pairs $(\mathbf{x_p}, \mathbf{t})$, of input and target vectors, where $p = 1, ..., N$ labels the individual training pairs [1].

The radial basis function approach introduces a set of N basis functions, one for each data point, which take the form $\phi(||\mathbf{x} - \mathbf{x^p}||)$ where $\phi(.)$ depends on the Euclidean distance between the training data point $\mathbf{x^p}$ and the untried point $\mathbf{x}$.

The radial basis function is commonly defined as Gaussian $\phi(||\mathbf{x} - \mathbf{x_p}||) = exp(-\beta||\mathbf{x} - \mathbf{x^p}||^2)$, where $\beta > 0$ is the width parameter. The weights ω_i and the bias ω_0 term are adaptive variables that are set during the training phase.

2.3 The Travelling Thief Problem

The Travelling Thief Problem (TTP) is a recently introduced combinatorial optimization problem that aims to provide testbeds for solving problems with multiple interdependent components [2]. The TTP combines two classical problems: the Travelling Salesman Problem and the Knapsack Problem. In [2] the authors show that it is impossible to solve the two components separately due to the dependencies between the two sub-problems.

In the TTP as redefined in [21], we are given a set of n cities, the associated matrix of distances d_{ij} between cities i and j, and a set of q items distributed among the n cities. Each item k is characterized by its profit p_k and weight w_k. A thief must visit all cities exactly once, stealing some items on the road, and return to the first city.

The total weight of the collected items must not exceed the knapsack capacity W. In addition, we consider a renting rate per time unit R that the thief must pay at the end of the travel, and the maximum and minimum velocities denoted v_{max} and v_{min} respectively. Each item is available in only one city, and we note $A_i \in \{1, \ldots, n\}$ the availability vector. A_i contains the reference to the city that contains the item i.

A TTP solution is coded in two parts. The first is the tour $c = (c_1, \ldots, c_n)$, a vector containing the ordered list of cities. The second is the picking plan $z = (z_1, \ldots, z_q)$, a binary vector representing the states of items (0 for packed, and 1 for unpacked).

To make the sub-problems mutually dependent, the TTP was designed such as the speed of the thief changes according to the knapsack weight. Therefore, the thief's velocity at city c is defined in Equation 3.

$$v_c = v_{max} - C \times w_c \quad (3)$$

where $C = \frac{v_{max} - v_{min}}{W}$ is a constant value, and w_c the weight of the knapsack at city c.

We note $g(z)$ is the total value of all collected items and $f(c, z)$ is the total travel time which are defined in Equations 4 and 5 respectively.

$$g(z) = \sum_q p_q \times z_q \text{ S.T. } \sum_q w_q \times z_q \leq W \quad (4)$$

$$f(c, z) = \sum_{i=1}^{n-1} t_{c_i, c_{i+1}} + t_{c_n, c_1} \quad (5)$$

where $t_{c_i, c_{i+1}} = \frac{d_{c_i, c_{i+1}}}{v_{c_i}}$ is the travel time from c_i to c_{i+1}.

The objective is to maximize the total travel gain function, as defined in Equation 6, by finding the best tour and picking plan.

$$G(c, z) = g(z) - R \times f(c, z) \quad (6)$$

Since the appearance of the TTP, several heuristic algorithms have been proposed to solve it. In [16] Mei et al. introduce a memetic algorithm named MATLS that is able to solve very large TTP instances. In the same work authors propose multiple speed up technique and fast greedy packing routines. The algorithm was later used to design a heuristic generation approach based on genetic programming in order to evolve packing routines [15].

Faulkner et al. [9] propose multiple local search algorithms and routines. Two families of heuristics are introduced: simple heuristics (S1-S5) and complex ones (C1-C6). The best performing heuristic according to the experimental study is S5. Although belonging to simple heuristics group, it is also able to surpass MATLS on most instances.

Wagner [30] investigate the use of longer tours using the Max-Min Ant System. This approach focuses on improving the tour accordingly to the overall TTP problem instead of using a Lin-Kernighan tour, which is designed for the TSP component independently. The approach is shown to be very efficient for small TTP instances.

Recently, El Yafrani and Ahiod have presented in [8] two heuristics. The first is a memetic algorithm using 2-opt and bit-flip local search heuristics, named MA2B. The second is a combination of a 2-opt local search and a simulated annealing based heuristic for efficient packing, named CS2SA. The two proposed heuristics have shown to be very competitive to other heuristics such as MATLS and S5.

For a computational comparison of 21 TTP algorithms and their use in algorithm portfolios, we refer the interested reader to [31].

3 THE PROPOSED APPROACH

The goal of our proposed approach is to consider EDA as a heuristic selection technique, aiming to evolve combinations of LLHs while looking for good problem solutions.

Our proposal is based on LLHs selection methodology as presented in [23]. Furthermore, in our implementation we explore the interactions (between the used LLHs) encoded into the bayesian network structure used as a PGM. Obtaining the probability distributions of related LLHs supports the generation of new solutions with correlated characteristics, besides providing a representative model for variables interactions. In addition, we apply a surrogate assisted model based on RBFNs to promptly evaluate the solutions sampled by the PGM.

In this section, we detail the HSEDA framework emphasizing its main characteristics.

3.1 Encoding scheme

Every solution is represented by an integer vector with M elements, $\mathbf{x} = (x_1, ..., x_M)$, denoting the decision variables, with element $x_m \in \{1, ..., L\}$, $m = 1...M$, L is the total number of LLHs and x_m indicates the particular LLH to be used at the m-th position in the sequence which the combination identifies.

Within the HSEDA context, a solution is equivalent to a combination of heuristics and a decision variable is equivalent to an LLH [23].

The LLHs are either a component heuristic or a disruptive operation. In the following, we present the list of candidate LLHs.

- KP_{BF}: A neighborhood search heuristic targeting the KP part. This heuristic uses a simple bit-flip local search empowered with speedup techniques. It is part of the memetic algorithm MATLS proposed in [16].
- KP_{SA}: A simulated annealing adapted to the KP sub-problem, which is used in CS2SA presented in [8].
- TSP_{2-opt}: A 2-opt based local search heuristic used for the TSP component. It is usually used in multiple TTP algorithms [8, 15].
- TSP^*_{swap}: A disruptive move for the TSP sub-problem that randomly swaps two cities.
- TSP^*_{4-opt}: A double bridge move for the TSP sub-problem that randomly selects the cities.
- $KP^*_{BF\psi}$: A disruptive routine that toggles the state of ψ% of the picking plan items, where $\psi \in \{20, 30, 40\}$.

The first three operators are local search heuristics obviously used for intensification purposes. While the remaining are disruptive operators designed to explore more efficiently the TTP search space.

The fitness of a heuristic combination depends on its performance on the given instance. The low-level heuristics resulting from a specific combination are sequentially applied to the problem instance, starting from an initial TTP solution. The initial solution is composed by a tour generated using the Lin-Kernighan heuristic [12], and a picking plan generated using the *PackIterative* routine [9]. The fitness of the heuristic combination $\mathbf{x}$ is set as the achieved TTP objective when the processing of all LLHs present in $\mathbf{x}$ is completed.

3.2 The framework

The proposed approach searches the space of possible combinations of LLHs. The main steps performed by HSEDA are described in Algorithm 1.

The *Initialization* phase loads the problem instance and randomly generates an initial population Pop^1 of N solutions. Each solution $\mathbf{x}$ is a sequence of LLHs of size M. By a sequence we mean an ordered set of LLHs in which repetition is possible.

The *EvaluateFitness* phase, Step 6 in Algorithm 1, calculates the fitness based on the TTP objective function, i.e. the fitness measures the quality of the sequence while attempting to improve the initial TTP solution.

In the *Selection* phase, a total of N_{PGM} solutions are selected from the population Pop^g, based on their fitness, to compose the Pop^g_{PGM}, where g is the current generation.

Algorithm 1 A simplified pseudo-code presenting the main components of HSEDA

INPUT: *Instance*: problem instance
N: population size
M: solution size
N_{PGM}: number of solutions selected to support the probabilistic model estimation
N_{smp}, number of solutions sampled from the probabilistic model
sr: survival rate
N_{sur}, number of solutions selected from the surrogate model
$Max_{runtime}$: maximum runtime
Max_{ger}: maximum number of generations
OUTPUT: C_{best}: the best found solution
{Initialization}
1: $I \leftarrow$ *LoadInstance*(*Instance*)
2: $Pop^1 \leftarrow$ *RandomGenerate*(N, I, M) {initial population}
3: $g \leftarrow 1$
{Main loop}
4: **while** $g \leq Max_{ger}$ and $Max_{runtime}$ is not exceeded do **do**
5: **for** each solution $\mathbf{x} \in Pop^g$ **do**
6: *fitness*($\mathbf{x}$) $\leftarrow$*EvaluateFitness*($\mathbf{x}$, I)
7: **end for**
{EDA: learning the probabilistic model}
8: $Pop^g_{\text{PGM}} \leftarrow$ Selection(Pop^g, N_{PGM}) {truncation selection}
9: $PGM \leftarrow$ *ProbabilisticModelEstimation*(Pop^g_{PGM})
{EDA: sampling}
10: $Pop_{\text{smp}} \leftarrow$ *Sampling*(PGM, N_{smp})
11: **for** each n solution $\mathbf{x} \in Pop_{\text{smp}}$ **do**
12: $fitness_n \leftarrow$*EvaluateFitnessRBFN*($\mathbf{x}_n$, I)
13: **end for**
{EDA: survival}
14: $Pop_{\text{smp}} \leftarrow$Selection($Pop^g$, N_{sur})
15: **for** each n solution $\mathbf{x} \in Pop_{\text{smp}}$ **do**
16: $fitness_n \leftarrow$*EvaluateFitness*($\mathbf{x}_n$, I)
17: **end for**
18: $Pop^{g+1} \leftarrow$*Survival*($\{Pop^g \cup Pop_{\text{smp}}\}$, N, sr) {new population}
19: $g \leftarrow g + 1$
20: **end while**
{Best found solution}
21: $C_{\text{best}} \leftarrow$ *SelectBest*(Pop^{g-1})

Afterward, HSEDA starts, at Step 9, the PGM construction phase in *ProbabilisticModelEstimation*, according to Pop^g_{PGM} population. Aiming to provide a probabilistic model, a bayesian network is modeled using *K2*-metric [7], and its structure and parameters (joint probability distributions) are estimated.

The PGM is used to sample the set of new solutions (Pop_{smp}) (Step 10). New solutions (N_{smp}), are generated from the joint distribution encoded by the network using the probabilistic logic sampling, and evaluated using a surrogate assisted model based on RBFNs. The RBFN aims to save computational resource that will be further used to extend the evolution. In our approach a solution vector from Pop_{smp} is an untried point for the RBFN model. The training set is composed by the set of solutions $\mathbf{x}$ for which the fitness is calculated based on the objective functions (instead of being approximated).

The solutions from Pop_{smp} are then sorted based on the surrogate function value and the N_{sur} best are selected to have their fitness recalculated by the real fitness function (Steps 14 to 17). The surrogate model is updated along the HSEDA evolutionary process every time the set of new solutions has its real fitness value calculated to compose the next training set.

This reduced sampled population (N_{sur} best individuals selected from Pop_{smp}), which has its fitness calculated by the TTP objective function, is joint with Pop^g to create the new population for the next generation. However, only N solutions are selected in the *Survival* process to proceed in the evolutionary process as a new population Pop^{g+1}. As shown in Step 18, $N * sr$ fittest solutions

from Pop^g, where sr is the survival rate, are selected followed by the remaining fittest ones from Pop_{smp}.

This process is iteratively performed until the termination criterion (Max_{ger} or $Max_{runtime}$, whichever comes first) is satisfied. Finally the best solution C_{best} is selected from the population, according the *SelectBest* function in Step 21.

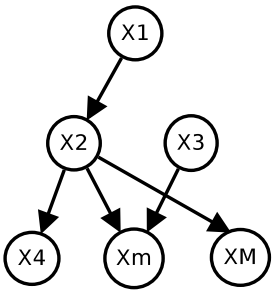

Figure 1: Example of a Bayesian Network Structure.

One example of an evolved Bayesian Network (BN) model can be seen in Figure 1, with node X_i representing the i-th low-level TTP heuristic to be applied and edges denoting relationships between nodes. The BN structure design considers the *K2* metric to build an edge between a node and its parents. In this case the BN structure indicates that the third heuristic (X_3) is conditionally independent from the first one (X_1). In the sample process, each node value depends on the conditional probability which is learned from the structure and the current set of best solutions. An example of an observation taken from this network is: $\mathbf{x} = [x_1, x_2, x_3, \ldots, x_M] = [KP_{BF}, TSP^*_{swap}, KP^*_{BF20}, KP_{SA}, KP_{BF}, TSP^*_{swap}]$. As the population of heuristic combinations is evolved along the generations, the structure and conditional probabilities are also updated. In this process, different BN structures can be generated at each generation for the given population.

4 EXPERIMENTS

The experiments conducted in this paper are performed on a comprehensive subset of the TTP benchmark instances[1] from [21] using Matlab[2]. The characteristics of these instances vary widely, and in this work we consider the following diversification parameters:

- The number of cities is based on TSP instances from the TSPlib, described in [24];
- For each TSP instance, there are three different types (we will refer to this parameter as T) of knapsack problems: uncorrelated (unc), uncorrelated with similar weights (usw), and bounded strongly correlated (bsc) types;
- For each TSP and KP combination, the number of items per city (item factor, denoted F) is $F \in \{01, 05, 10\}$;
- For each TTP configuration, we use 3 different knapsack capacities $C \in \{01, 05, 10\}$. C represents a capacity class.

To evaluate the proposed hyper-heuristic, we use five representative TTP instance groups: *eil51*, *kroA100*, *a280*, *pcb439* and *rat783*. Therefore, using this setting, a total of 135 instances are considered in this paper. While significantly larger TTP instances exist, our subset still spans 59% of the instances, when measured in the number of cities. In addition, we use the following notation to represent a TTP instance: $TSP\_base\_instance(F, T, C)$.

The experiments for HSEDA were performed for a maximum runtime of 10 minutes per instance[3].

The proposed framework has multiple parameters that need tuning. These parameters are presented in Table 1, and are offline tuned using the automated algorithm configuration package *irace* proposed by López-Ibáñez et al. [14]. The *irace* package is implemented in *R* and is based on the iterated racing procedure. *irace* is a state-of-the art algorithm configuration approach for solution quality scenarios. It iteratively applies a racing technique to a set of sampled configurations against each other until enough empirical evidence allows to reject all but one configuration using a statistical test.

Table 1: Parameters of HSEDA algorithm.

Description	Value
Population size N	10
Maximum runtime $Max_{runtime}$	600s
Maximum number of generations Max_{ger}	1000
Solution size M	6
Number of solutions selected from surrogate model N_{sur}	$7 * N$
Survival Rate sr	0.75
Number of solutions to the PGM N_{PGM}	N
Number of solutions sampled N_{smp}	1000

In the *irace* training phase, we use the following groups of small TTP instances: eil51, berlin52, eil76, and kroA100. These groups are different from the instances considered for our experiments, which were mentioned before and also includes eil51 and kroA100. This way we can test both HSEDA 's ability to create high-quality TTP solutions on the training set, and its ability to generalize to the test set. Additionally, we consider a budget for *irace* of 1000 experiments.

The surrogate model was evaluated by comparing the same solutions sorted by real fitness and approximated ones. Therefore, when the RBFN model is applied, we expect that the solutions appear in close order for both sets. Additionally, our preliminary experiments to assess the surrogate model accuracy show that at least 50% of the best solutions are selected.

5 RESULTS AND DISCUSSION

This section presents a comparison between our HSEDA approach and three other state-of-art algorithms for the TTP, namely, MMAS [30], MA2B [8], and S5 [9].

In order to provide a statistical analysis of the results, the Friedman test was applied to the obtained objectives values, considering that the results are not normally distributed, based on the Shapiro-Wilk normality test [6]. This test compares a set of instances of a problem and the results reveal all the algorithms have results values with no statistically significant difference can be rejected for all instances. All tests have been executed with a confidence

[1] All the TTP instances can be found in the website: http://cs.adelaide.edu.au/optlog/research/ttp.php.

[2] We used the bayes net toolbox for Matlab [19], publicly available at https://github.com/bayesnet/bnt.

[3] The HSEDA source code and raw result files are publicly available at https://bitbucket.org/marcella_engcomp/hseda-ttp

level of 95% ($\alpha = 0.05$) considering the 30 independent runs of each algorithm.

Table 2 shows the statistical analysis of pairwise comparisons between HSEDA and the state-of-art algorithms using Dunn-Sidak's post-hoc test with a significance level of $\alpha = 0.05$. When the test result is greater than α, there is no statistical difference between the two techniques. When the test result is less than α, there is statistical difference.

The entries representing a statistically significant difference between HSEDA and the other approach are emphasized (bold). When HSEDA shows better performance (average of its objective values) the background is highlighted.

Table 2: Results for pairwise comparisons among HSEDA and state-of-art algorithms using Friedman and Dunn-Sidak's post-hoc tests with $\alpha = 0.05$ for each group of instances.

Instance	HSEDA x MMAS	HSEDA x MA2B	HSEDA x S5
$eil51$	**0.0000**	0.4104	**0.0002**
$kroA100$	0.1519	0.8606	**0.0000**
$a280$	**0.0000**	**0.0017**	0.4260
$pr439$	**0.0002**	1.0000	0.4880
$rat783$	**0.0000**	**0.0001**	1.0000

We can observe from Table 2 that there are statistical differences between HSEDA and MMAS for almost all instances set, except for kroA100. HSEDA is better than MMAS for a280, pr429 and rat783 instances and worst for the eil51 group. In comparison with MA2B, HSEDA is better for a280 and rat783 instance sets. On the other hand, the results for S5 show statistical differences for the eil51 and kroA100 instances, where HSEDA is better.

The behaviours of MA2B, MMAS, and S5 are quite expected:

- MA2B is a memetic algorithm using expensive local search on a population of 40 individuals and explores longer tours implicitly though a *2-opt* local search heuristic. This makes the algorithm efficient for small instances, but its performance decreases when dealing with larger instances due to the 10 minutes time limit.
- MMAS uses artificial ants to explicitly explore longer tours by focusing on good TTP tours. This has been shown to be a very efficient strategy for small instances. However, its performance decreases significantly on instances having more than 250 cities and 2000 items as shown in [30].
- S5 has been shown to be very competitive on mid-size and large instances. Its good performance is mainly due to the PackIterative heuristics which is a fast search technique based on a greedy approach. However, S5 is very biased as it does not explore longer tours at all which explains its mediocre performance on small instances.

Aiming to present a performance measure to explore the HSEDA behavior regarding each specific instance, in this paper we have applied the A-test (Vargha-Delaney A measure [29]), a measure of stochastic superiority that ranks the samples based on their values. The A-test returns a value between 0 and 1. This value represents the probability that a randomly selected observation from one sample is better than a randomly selected observation from the other sample. In our case, the two samples are composed of the objective values from each algorithm run. The test was realized using 30 independent runs of each algorithm.

Tables 3, 4, 5, 6 and 7 show the A-test of a pairwise comparison between these algorithms for each instance, respectively.

When the A-measure is exactly 0.5, there is no difference between the two techniques. When the A-measure is less than 0.5 the first technique has the worse performance. Otherwise, the second technique is the worst performing one.

The cells representing when HSEDA is stochastically superior than another approach are emphasized (highlighted).

Table 3: A-test over *eil51* instances.

Instance	HSEDA x MMAS	HSEDA x MA2B	HSEDA x S5
$eil51(01, bsc, 01)$	0.0000	0.7794	1.0000
$eil51(05, bsc, 01)$	0.0000	0.5972	1.0000
$eil51(10, bsc, 01)$	0.0000	0.3000	1.0000
$eil51(01, bsc, 05)$	0.0000	0.4928	1.0000
$eil51(05, bsc, 05)$	0.0000	0.3878	1.0000
$eil51(10, bsc, 05)$	0.1856	0.0667	1.0000
$eil51(01, bsc, 10)$	0.0000	0.7378	1.0000
$eil51(05, bsc, 10)$	0.0000	0.6722	1.0000
$eil51(10, bsc, 10)$	0.0000	0.0000	1.0000
$eil51(01, usw, 01)$	0.0000	0.1572	1.0000
$eil51(05, usw, 01)$	0.0000	1.0000	1.0000
$eil51(10, usw, 01)$	0.0000	0.0878	1.0000
$eil51(01, usw, 05)$	0.0000	0.2522	1.0000
$eil51(05, usw, 05)$	0.0000	0.9000	1.0000
$eil51(10, usw, 05)$	0.0333	0.0000	1.0000
$eil51(01, usw, 10)$	0.0000	0.9333	1.0000
$eil51(05, usw, 10)$	0.0000	0.0000	1.0000
$eil51(10, usw, 10)$	0.0000	0.0300	1.0000
$eil51(01, unc, 01)$	0.0000	0.0000	1.0000
$eil51(05, unc, 01)$	0.0000	0.0000	1.0000
$eil51(10, unc, 01)$	0.0000	0.0000	1.0000
$eil51(01, unc, 05)$	0.0000	0.0333	1.0000
$eil51(05, unc, 05)$	0.0000	0.0000	1.0000
$eil51(10, unc, 05)$	0.0044	0.0000	1.0000
$eil51(01, unc, 10)$	0.0267	0.7567	1.0000
$eil51(05, unc, 10)$	0.0000	0.0000	1.0000
$eil51(10, unc, 10)$	0.0000	0.0000	1.0000

Table 4: A-test over *kroA100* instances.

Instance	HSEDA x MMAS	HSEDA x MA2B	HSEDA x S5
$kroA100(01, bsc, 01)$	0.0000	0.7989	1.0000
$kroA100(05, bsc, 01)$	0.0022	0.5878	0.9667
$kroA100(10, bsc, 01)$	0.0000	0.0333	0.0000
$kroA100(01, bsc, 05)$	0.0667	0.0333	1.0000
$kroA100(05, bsc, 05)$	0.9411	1.0000	1.0000
$kroA100(10, bsc, 05)$	0.8333	1.0000	1.0000
$kroA100(01, bsc, 10)$	0.4311	0.9489	1.0000
$kroA100(05, bsc, 10)$	1.0000	1.0000	1.0000
$kroA100(10, bsc, 10)$	0.9667	1.0000	1.0000
$kroA100(01, usw, 01)$	0.0000	0.7094	1.0000
$kroA100(05, usw, 01)$	0.0000	0.6667	1.0000
$kroA100(10, usw, 01)$	0.1256	0.7444	1.0000
$kroA100(01, usw, 05)$	0.3578	0.9178	1.0000
$kroA100(05, usw, 05)$	0.0000	0.8578	1.0000
$kroA100(10, usw, 05)$	0.1600	1.0000	1.0000
$kroA100(01, usw, 10)$	0.0000	0.8000	1.0000
$kroA100(05, usw, 10)$	0.0000	1.0000	1.0000
$kroA100(10, usw, 10)$	0.7878	1.0000	1.0000
$kroA100(01, unc, 01)$	0.0000	0.0000	0.0000
$kroA100(05, unc, 01)$	0.0000	0.0000	0.0000
$kroA100(10, unc, 01)$	0.0000	0.0000	1.0000
$kroA100(01, unc, 05)$	0.0000	0.9400	1.0000
$kroA100(05, unc, 05)$	1.0000	1.0000	1.0000
$kroA100(10, unc, 05)$	0.6900	0.9000	0.9000
$kroA100(01, unc, 10)$	0.0000	0.9467	1.0000
$kroA100(05, unc, 10)$	0.0000	0.0000	1.0000
$kroA100(10, unc, 10)$	0.0000	0.0044	1.0000

In Table 3, we note that MMAS presents the best results in comparison with HSEDA for *eil51* instances. MA2B also provides better results for most instances. However, HSEDA is better than S5 for all instances.

We can observe in Table 4 that HSEDA is better than MMAS for some instances with bounded strongly correlated profit/weight. On the other hand, we can see that HSEDA is better than MA2B and S5 for most *kroA100* instances.

Table 5: A-test over *a280* instances.

Instance	HSEDA x MMAS	HSEDA x MA2B	HSEDA x S5
a280(01, bsc, 01)	0.9089	1.0000	0.7533
a280(05, bsc, 01)	1.0000	1.0000	0.8344
a280(10, bsc, 01)	1.0000	1.0000	0.5633
a280(01, bsc, 05)	0.9967	1.0000	0.6722
a280(05, bsc, 05)	0.8189	0.9000	1.0000
a280(10, bsc, 05)	1.0000	1.0000	0.0000
a280(01, bsc, 10)	0.8756	0.9561	0.0000
a280(05, bsc, 10)	0.5433	0.8967	0.1433
a280(10, bsc, 10)	1.0000	1.0000	0.1767
a280(01, usw, 01)	0.0967	0.1783	0.5800
a280(05, usw, 01)	1.0000	0.4000	0.0000
a280(10, usw, 01)	0.8878	0.3361	0.5700
a280(01, usw, 05)	0.0000	0.3211	0.7544
a280(05, usw, 05)	0.9389	0.9911	0.6378
a280(10, usw, 05)	1.0000	1.0000	0.7133
a280(01, usw, 10)	1.0000	0.9333	1.0000
a280(05, usw, 10)	1.0000	1.0000	1.0000
a280(10, usw, 10)	1.0000	1.0000	0.6056
a280(01, unc, 01)	0.1844	0.5133	1.0000
a280(05, unc, 01)	0.9422	0.5011	0.9867
a280(10, unc, 01)	1.0000	0.5428	1.0000
a280(01, unc, 05)	1.0000	0.8978	1.0000
a280(05, unc, 05)	1.0000	0.9439	0.8089
a280(10, unc, 05)	1.0000	1.0000	0.6856
a280(01, unc, 10)	1.0000	0.6583	1.0000
a280(05, unc, 10)	1.0000	1.0000	0.6811
a280(10, unc, 10)	1.0000	1.0000	0.8378

Table 6: A-test over *pr439* instances.

Instance	HSEDA x MMAS	HSEDA x MA2B	HSEDA x S5
pr439(01, bsc, 01)	0.0122	0.0000	0.0000
pr439(05, bsc, 01)	0.9222	1.0000	0.0889
pr439(10, bsc, 01)	1.0000	1.0000	0.7089
pr439(01, bsc, 05)	0.9467	0.9289	0.1589
pr439(05, bsc, 05)	1.0000	1.0000	0.5333
pr439(10, bsc, 05)	1.0000	1.0000	0.4256
pr439(01, bsc, 10)	0.9333	0.0967	0.3900
pr439(05, bsc, 10)	1.0000	0.9667	0.1711
pr439(10, bsc, 10)	1.0000	0.6000	0.2333
pr439(01, usw, 01)	0.7867	0.5122	0.8956
pr439(05, usw, 01)	0.4967	0.1044	0.0156
pr439(10, usw, 01)	0.9900	0.6489	0.2178
pr439(01, usw, 05)	0.8322	0.0000	0.1467
pr439(05, usw, 05)	0.9667	0.0578	0.5278
pr439(10, usw, 05)	1.0000	0.1367	0.2356
pr439(01, usw, 10)	1.0000	0.8611	0.7133
pr439(05, usw, 10)	1.0000	0.5533	0.5800
pr439(10, usw, 10)	1.0000	0.4778	0.1756
pr439(01, unc, 01)	0.0400	0.0533	0.0000
pr439(05, unc, 01)	1.0000	1.0000	0.5978
pr439(10, unc, 01)	1.0000	0.9839	0.1633
pr439(01, unc, 05)	1.0000	0.0433	0.4811
pr439(05, unc, 05)	1.0000	0.2439	0.0878
pr439(10, unc, 05)	1.0000	0.3700	0.1744
pr439(01, unc, 10)	0.9967	0.2000	0.0711
pr439(05, unc, 10)	1.0000	0.2133	0.3300
pr439(10, unc, 10)	1.0000	0.2911	0.3367

Table 7: A-test over *rat783* instances.

Instance	HSEDA x MMAS	HSEDA x MA2B	HSEDA x S5
rat783(01, bsc, 01)	0.9911	0.9878	0.9711
rat783(05, bsc, 01)	1.0000	1.0000	0.8944
rat783(10, bsc, 01)	1.0000	1.0000	0.0456
rat783(01, bsc, 05)	0.9989	0.9522	0.9800
rat783(05, bsc, 05)	1.0000	1.0000	1.0000
rat783(10, bsc, 05)	1.0000	1.0000	0.9256
rat783(01, bsc, 10)	1.0000	0.9178	0.9789
rat783(05, bsc, 10)	1.0000	0.5667	0.0000
rat783(10, bsc, 10)	1.0000	0.5667	0.0000
rat783(01, usw, 01)	0.8311	0.7356	0.1489
rat783(05, usw, 01)	1.0000	1.0000	0.0000
rat783(10, usw, 01)	1.0000	1.0000	0.0000
rat783(01, usw, 05)	1.0000	0.8333	0.0000
rat783(05, usw, 05)	1.0000	1.0000	0.9667
rat783(10, usw, 05)	1.0000	1.0000	0.4667
rat783(01, usw, 10)	1.0000	1.0000	0.7667
rat783(05, usw, 10)	1.0000	1.0000	0.0333
rat783(10, usw, 10)	1.0000	1.0000	0.0000
rat783(01, unc, 01)	1.0000	1.0000	1.0000
rat783(05, unc, 01)	1.0000	0.9667	0.9667
rat783(10, unc, 01)	1.0000	1.0000	0.6000
rat783(01, unc, 05)	1.0000	0.9000	0.7667
rat783(05, unc, 05)	1.0000	1.0000	0.6267
rat783(10, unc, 05)	1.0000	1.0000	0.6333
rat783(01, unc, 10)	1.0000	0.6000	0.4333
rat783(05, unc, 10)	1.0000	1.0000	0.4333
rat783(10, unc, 10)	1.0000	0.9167	0.0000

In Table 5, HSEDA has a better performance than the other algorithms for most instances of the *a280* group.

Table 6 shows that HSEDA is better than MMAS for almost all *pr439* instances. However, HSEDA is less competitive than MA2B and S5 for this group of instances.

Finally, the results reported in Table 7 show that HSEDA is better than MMAS and MA2B for all *rat783* instances. We can also observe that HSEDA is able to outperform S5 for some instances.

Lastly, we compare all algorithms based on their approximation of the best solutions seen. For this, we compute the approximation ratio of the average performance to the best objective score found using Equation 7.

$$approxRatio_I^A = \frac{\sum_{r=1}^{R} fit_r / R}{fit_{best}} \quad (7)$$

where I is the instance, A is the considered approach, R is the total number of runs ($R = 30$ and $r \in \{1, ..., R\}$) and $fit_{best} = max(fit_{\forall r,A})$. This ratio allows us to compare the overall performances across the chosen set of instances, since the objective values vary across several magnitude orders [30].

In Figure 2, we show a summary of over 540 (135 instances and 4 algorithms) average approximation ratios as trend lines. The curves are polynomials of degree 6. The purpose of this visualization is to provide a general qualitative summary of the algorithm's performance in a multi-dimensional instance space.

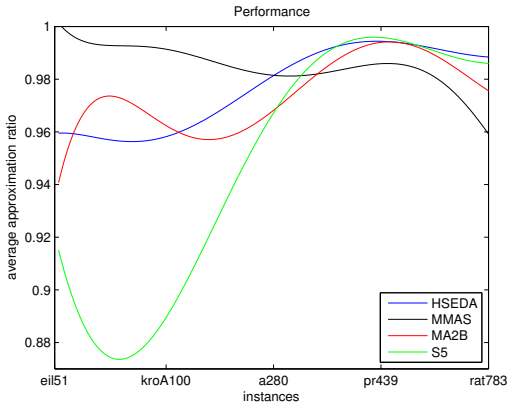

Figure 2: Performance comparison: Summary of results shown as trend lines.

We can see that MMAS still performs best on small TTP instances due to its ability to search for longer tours. For the mid-sized instances from the a280 class on (on which HSEDA was not trained), HSEDA is typically either comparable to the previous MA2B and S5, or it outperforms them.

6 CONCLUDING REMARKS

In this paper we have proposed an EDA framework as a heuristic selection approach within the hyper-heuristic context. The HSEDA approach was designed and evaluated for the Travelling Thief Problem.

The main issue investigated in this paper concerns whether a Bayesian Network as a Probabilistic Graphical Model (PGM) embedded in a Hyper-Heuristic generates promising solutions combining sub-heuristics and disruptive operators in a heuristic selection search technique. Thus, a sequence of these promising low-level heuristics (LLHs) can be sampled and evolved based on the estimation of variable distribution of promising solutions with correlated characteristics, even more it can provide a representative model for variables interactions.

We have analyzed the performance of the proposed approach on small and medium-sized TTP instances. We provided a statistical analysis of a pairwise comparison between HSEDA and three other state-of-the-art algorithms.

Although the reported results show that HSEDA is very competitive, experiments were limited to small and mid-size instances. This inconvenience is due to the following two reasons: (*i*) HSEDA is an online training approach, which means it focuses on the instance at hand and its abilities cannot be transferred to unseen instances; and (*ii*) the 10 minutes runtime limit makes it difficult to solve large TTP instances, even for very complex and sophisticated heuristics such as S5 and MA2B.

Based on the experiments with the TTP instances addressed in this work, we can conclude that the proposed heuristic selection is competitive when compared with the other investigated state-of-art algorithms, especially when the number of cities increases.

We believe that HSEDA competitive performance is due to three key factors: (*i*) the use of the Bayesian network, which has the ability to efficiently represent the dependencies and independencies between the LLHs; (*ii*) the use of two types of operators, namely local search heuristics, and disruptive operators; (*iii*) this guarantees a good balance between HSEDA's exploration and exploitation abilities; and (*iv*) the surrogate assisted model, which saves computational resource that is further used to extend the HSEDA evolution.

Within the experiments the processes have been simplified, but the methodology allows to extract much more knowledge at the end, which have not been explored here due the space limitation. In the future, we engage to explore probabilistic inferences in the PGM, combining prior knowledge, in causal form, and data, as well as incorporating more sophisticated machine learning techniques in surrogate assisted models.

7 ACKNOWLEDGEMENTS

M. Martins acknowledges CAPES/Brazil. M. Delgado acknowledges CNPq grant Nos.: 309197/2014-7 (Brazil Government). M. Wagner acknowledges the ARC Discovery Early Career Researcher Award DE160100850. This work was supported with supercomputing resources provided by the Phoenix HPC service at the University of Adelaide.

REFERENCES

[1] Chris Bishop. 1991. Improving the generalization properties of radial basis function neural networks. *Neural computation* 3, 4 (1991), 579–588.
[2] M. R. Bonyadi, Z. Michalewicz, and L. Barone. 2013. The travelling thief problem: The first step in the transition from theoretical problems to realistic problems. In *Proceedings of the 2013 IEEE Congress on Evolutionary Computation*. 1037–1044.
[3] D. S. Broomhead and D. Lowe. 1988. Multi-variable functional interpolation and adaptive networks. *Complex Systems* 2 (1988), 321–355.
[4] Edmund K Burke, Michel Gendreau, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and Rong Qu. 2013. Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society* 64, 12 (2013), 1695–1724.
[5] Edmund K. Burke, Sanja Petrovic, and Rong Qu. 2006. Case-based heuristic selection for timetabling problems. *Journal of Scheduling* 9, 2 (2006), 115–132.
[6] W.J. Conover. 1999. *Practical Nonparametric Statistics* (third ed.). Wiley.
[7] G. Cooper and E. Herskovits. 1992. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning* 9, 4 (1992), 309–347.
[8] Mohamed El Yafrani and Belaïd Ahiod. 2016. Population-based vs. Single-solution Heuristics for the Travelling Thief Problem. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016 (GECCO'16)*. 317–324.
[9] Hayden Faulkner, Sergey Polyakovskiy, Tom Schultz, and Markus Wagner. 2015. Approximate approaches to the traveling thief problem. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation (GECCO'15)*. ACM, 385–392.
[10] Fred W Glover and Gary A Kochenberger. 2006. *Handbook of metaheuristics*. Vol. 57. Springer Science & Business Media.
[11] P. Larrañaga and J. A. Lozano. 2002. *Estimation of distribution algorithms: A new tool for evolutionary computation*. Vol. 2. Springer, Netherlands.
[12] Shen Lin and Brian W Kernighan. 1973. An effective heuristic algorithm for the traveling-salesman problem. *Operations research* 21, 2 (1973), 498–516.
[13] Bo Liu, Qingfu Zhang, and Georges GE Gielen. 2014. A Gaussian process surrogate model assisted evolutionary algorithm for medium scale expensive optimization problems. *IEEE Transactions on Evolutionary Computation* 18, 2 (2014), 180–192.
[14] M. López-Ibáñez, J. Dubois-Lacoste, T. Stützle, and M. Birattari. 2011. *The irace Package: Iterated Racing for Automatic Algorithm Configuration*. IRIDIA Technical Report Series 2011-004. Universit? Libre de Bruxelles, Bruxelles,Belgium.
[15] Yi Mei, Xiaodong Li, Flora Salim, and Xin Yao. 2015. Heuristic evolution with genetic programming for traveling thief problem. In *Proceedings of the 2015 IEEE Congress on Evolutionary Computation, CEC*. IEEE, 2753–2760.
[16] Yi Mei, Xiaodong Li, and Xin Yao. 2014. Improving efficiency of heuristics for the large scale traveling thief problem. In *Proceedings of the Asia-Pacific Conference on Simulated Evolution and Learning*. Springer, 631–643.
[17] Alberto Moraglio and Ahmed Kattan. 2011. Geometric generalisation of surrogate model based optimisation to combinatorial spaces. In *European Conference on Evolutionary Computation in Combinatorial Optimization*. Springer, 142–154.
[18] H. Mühlenbein and G. Paab. 1996. From Recombination of Genes to the Estimation of Distributions I. Binary parameters. In *Parallel Problem Solving from Nature-PPSN IV (Lecture Notes in Computer Science 1411)*. 178–187.
[19] Kevin Murphy and others. 2001. The bayes net toolbox for matlab. *Computing science and statistics* 33, 2 (2001), 1024–1034.
[20] Sergio Nesmachnow. 2014. An overview of metaheuristics: accurate and efficient methods for optimisation. *International Journal of Metaheuristics* 3, 4 (2014), 320–347.
[21] Sergey Polyakovskiy, Mohammad Reza Bonyadi, Markus Wagner, Zbigniew Michalewicz, and Frank Neumann. 2014. A Comprehensive Benchmark Set and Heuristics for the Traveling Thief Problem. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation (GECCO'14)*. 477–484.
[22] Rong Qu, Edmund K. Burke, and Barry McCollum. 2009. Adaptive automated construction of hybrid heuristics for exam timetabling and graph colouring problems. *European Journal of Operational Research* 198, 2 (2009), 392 – 404.
[23] Rong Qu, Nam Pham, Ruibin Bai, and Graham Kendall. 2015. Hybridising heuristics within an estimation distribution algorithm for examination timetabling. *Applied Intelligence* 42, 4 (2015), 679–693.
[24] Gerhard Reinelt. 1991. TSPLIB?A Traveling Salesman Problem Library. *ORSA Journal on Computing* 3, 4 (1991), 376–384.
[25] Peter Ross. 2005. *Hyper-Heuristics*. Springer US, Boston, MA, 529–556.
[26] Eric Soubeiga. 2003. *Development and Application of Hyperheuristics to Personeel Scheduling*. Phd Thesis. School of Computer Science and Information Technology, University of Nottingham.
[27] Gönül Uludağ, Berna Kiraz, A. Şima Etaner-Uyar, and Ender Özcan. 2012. *A Framework to Hybridize PBIL and a Hyper-heuristic for Dynamic Environments*. Springer, Berlin, Heidelberg, 358–367.
[28] Gönül Uludağ, Berna Kiraz, A. Şima Etaner-Uyar, and Ender Özcan. 2012. Heuristic selection in a multi-phase hybrid approach for dynamic environments. In *2012 12th UK Workshop on Computational Intelligence (UKCI)*. 1–8.
[29] András Vargha and Harold D. Delaney. 2000. A Critique and Improvement of the CL Common Language Effect Size Statistics of McGraw and Wong. *Journal of Educational and Behavioral Statistics* 25, 2 (2000), 101–132.
[30] Markus Wagner. 2016. Stealing Items More Efficiently with Ants: A Swarm Intelligence Approach to the Travelling Thief Problem. In *Proceedings of the 10th International Conference on Swarm Intelligence, ANTS 2016*. Springer, Brussels, Belgium, 273–281.
[31] Markus Wagner, Marius Lindauer, Mustafa Mısır, Samadhi Nallaperuma, and Frank Hutter. 2017. A case study of algorithm selection for the traveling thief problem. *Journal of Heuristics* (2017), 1–26.