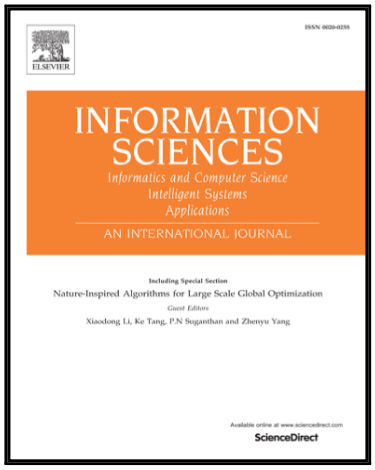
Accepted Manuscript



An evolutionary algorithm based hyper-heuristic framework for the set packing problem

Sachchida Nand Chaurasia, Joong Hoon Kim

PII: S0020-0255(19)30684-X
DOI: https://doi.org/10.1016/j.ins.2019.07.073
Reference: INS 14721

To appear in: *Information Sciences*

Received date: 13 June 2018
Revised date: 17 July 2019
Accepted date: 19 July 2019

Please cite this article as: Sachchida Nand Chaurasia, Joong Hoon Kim, An evolutionary algorithm based hyper-heuristic framework for the set packing problem, *Information Sciences* (2019), doi: https://doi.org/10.1016/j.ins.2019.07.073

Highlights

- An evolutionary algorithm based hyper-heuristic is proposed for the set packing problem (SPP) and the minimum weight dominating set (MWDS) problem.
- Self-learning concept is employed in hyper-heuristic for a selection of heuristic.
- Dynamic selection of parameter is adopted to make the approach, as much as possible, less dependent on parameter values. The empirical study shows the effectiveness of Dynamic selection of parameter.
- The proposed approach has been compared with the respective state-of-the-art approaches for both the SPP and MWDS problem.
- Computational results show the superiority of the proposed approach.

An evolutionary algorithm based hyper-heuristic framework for the set packing problem

Sachchida Nand Chaurasia[a, b]

[a]*Research Center for Disaster Prevention Science and Technology, Korea University, 02841, Seoul, South Korea*
[b]*Operational Research Lab, School of Electronic Engineering and Computer Science, Queen Mary University of London, London E1 4FZ, United Kingdom*

Joong Hoon Kim[c*]

[c]*School of Civil, Environmental and Architectural Engineering, Korea University, 02841, Seoul, South Korea*

Abstract

In recent years, hyper-heuristics have received massive attention from the research community as an alternative of meta-heuristics. In a hyper-heuristic, generation or selection of an effective heuristic among a pool of heuristics is an important and challenging task in the search process. At each iteration, a suitable heuristic can take the search process toward the global optimal solution. Moreover, some additional factors such as quality and the number of heuristics also affect the performance. In this paper, we propose an evolutionary algorithm based hyper-heuristic framework that incorporates dynamic selection of parameters. To test its generality, effectiveness and robustness, we apply this approach on two different $\mathcal{NP}$*-hard* problems - set packing problem (SPP) and minimum weight dominating set (MWDS) problem. The proposed approach for the SPP and the MWDS problem has been evaluated respectively on their respective set of benchmark instances. Computational results show that the proposed approach for the SPP and MWDS problem perform much better than their respective state-of-the-art approaches in terms of the solution quality and computational time.

Keywords: Set packing problem, Estimation of distribution algorithm, Guided-mutation, Heuristic, Hyper-heuristic, Minimum weight dominating set problem

1. Introduction

*Corresponding author
Email addresses: sachchidanand.mca07@gmail.com (Sachchida Nand Chaurasia[a, b]), jaykim@korea.ac.kr (Joong Hoon Kim[c])

The set packing problem (SPP), considered to closely resemble a set covering problem [20, 37], is a classical combinatorial optimization problem and has been proven to be $\mathcal{NP}$-*hard* [18] in nature. We followed the same notational representation as in [14] to represent the SPP and is defined as follows: Consider a finite set $\mathcal{I} = \{1, \ldots, \mathbf{N}\}$ of $\mathbf{N}$ objects, $\mathcal{T}_j$, $j \in \mathcal{J} = \{1, \ldots, \mathbf{M}\}$ a list of $\mathbf{M}$ subsets of $\mathcal{I}$, and a packing $\mathcal{P} \subseteq \mathcal{I}$ is a subset of set $\mathcal{I}$ such that $|\mathcal{T}_j \cap \mathcal{P}| \leqslant 1, \forall j \in \mathcal{J}$, i.e., at most one object of set $\mathcal{T}_j$ can be in packing $\mathcal{P}$. Each set $\mathcal{T}_j, j \in \mathcal{J} = \{1, \ldots, \mathbf{M}\}$ is considered as an exclusive constraint between some objects of set $\mathcal{I}$. A weight function assigns a positive integer weight, c_i, to each object i in set $\mathcal{I}$. The objective of the SPP is to find a subset $\mathcal{P} \subseteq \mathcal{I}$ that maximizes the sum of the weights of the objects in set $\mathcal{P}$. Mathematical model of the SPP is given as follows:

$$Max \ \mathbf{Z} = \sum_{i \in \mathcal{I}} c_i x_i, \tag{1}$$

$$\sum_{i \in \mathcal{I}} t_{i,j} x_i \leqslant 1, \ \forall j \in \mathcal{J} \tag{2}$$

$$x_i \in \{0, 1\}, \ \forall i \in \mathcal{I} \tag{3}$$

$$t_{i,j} \in \{0, 1\}, \ \forall i \in \mathcal{I}, \ \forall j \in \mathcal{J} \tag{4}$$

- a vector $\mathcal{X} = (x_i)$, where $x_i = \begin{cases} 1, & \text{if } i \in \mathcal{P} \\ 0, & \text{otherwise} \end{cases}$
- a vector $\mathcal{C} = (c_i)$, where c_i= weight of object $i, \forall i \in \mathcal{I}$
- a matrix $\mathcal{T} = (t_{i,j})$, where $t_{i,j} = \begin{cases} 1, & \text{if } i \in \mathcal{T}_j, \\ 0, & \text{otherwise} \end{cases}$

Equation (1) is the objective function that calculates the sum of the weights of the objects in packing $\mathcal{P}$. Equation (2) ensures that at most one object from set $\mathcal{T}_j$ can be packed in packing $\mathcal{P}$. In Equation (3), x_i is a binary variable. If object i is in packing $\mathcal{P}$, then x_i=1; otherwise, x_i=0. In Equation (4), $t_{i,j}$ is a binary variable and its value is 1 if object i belongs to set $\mathcal{T}_j$; otherwise, it is 0.

A special case of the SPP is a node packing problem represented in Equation (5), where

constraints are defined between a pair of items.

$$\begin{bmatrix} Max\ \mathbf{Z} = \sum_{i \in \mathcal{I}} c_i x_i, \\ x_i + x_j \leqslant 1, \forall \text{ pair } (i, j) \text{ of incompatible items}, \\ x_i \in \{0, 1\}, \forall i \in \mathcal{I} \end{bmatrix} \tag{5}$$

1.1. Survey on the set packing problem and related problems

The SPP is an $\mathcal{NP}$-*hard* problem and its hardness is explained in detail in [20, 37]. The Branch-and-cut algorithm, uses polyhedral theory [39], was the best exact approach for solving the SPP. However, owing to the solving limitation of an exact approach, only small instances are solved to optimality. For large instances, other approaches such as meta-heuristic approaches should be considered, which play an important role in determining the solution of acceptable quality in a reasonable amount of computational time. Considering the limitations of the exact approaches, Delorme et al. [14] proposed a greedy randomized adaptive search procedure (GRASP) which operates in two main phases. In the first phase, called a construction phase, an initial solution is generated using a greedy randomized procedure in which randomness allows to explore different search areas. In the second phase, a local search is applied to each solution generated in the first phase to further improve it. Further, an intensification phase based on the path relinking method[41] was applied. In the study reported in [14], two different types of instances were used: real railway problem and random instances. In [17], an ant colony optimization (ACO) was proposed for solving the SPP, and only random instances were used for investigating the performance of the ACO. Further, in [16] and [34], two different versions of ACO were proposed for the SPP, and both were tested on the railway problem instances. The state-of-the-art approach for solving the SPP is an evolutionary algorithm (EA) with guided mutation (GM) (i.e., EA/G) [11]. The EA/G approach is considered as a cross between traditional genetic algorithm and estimation of distribution algorithm. The GM operator uses the cross information to generate offspring. The solutions generated by the EA/G approach are subjected to a local search algorithm to further improve the solution quality. The EA/G with local search was applied to unit-cost and multi-cost random instances.

Many real-world applications of the SPP have been reported in the literature. In [42], Rönnqvist presented the use of a SPP to formulate the cutting stock problem and solved it using a combination of the Lagrangian relaxation method and sub-gradient optimization. In [30],

Kim and Lee mapped a ship scheduling problem as the SPP and solved it using LINDO software. In [35], a SPP formulation was used to obtain the bounds for a resource-constrained project scheduling problem using the greedy method. In [47], Tajima and Misono formulated an airline seat allocation/reallocation problem as a SPP and solved it using the IBM optimization subroutine library. In [43], Rossi and Smriglio described the development of a SPP-based model for a ground holding problem and solved it using a Branch-and-cut method. In [15], Delorme et al. used a unit-cost SPP to model the railway infrastructure saturation problem and solved it using a GRASP meta-heuristic.

In [33], Lusby et al. formulated a routing of trains through junctions problem as a SPP and solved it using a branch-and-price algorithm. In the given problem, the tracks and time are divided into sections and time periods, respectively. Further, pairs of track sections and time periods are mapped as objects into a SPP with the objective to maximize the number of trains that can be assigned to pairs of track sections and time periods. It ensures that at any particular time period at most one train can be assigned to a pair of track section and time period. It also ensures that at any particular time period only one train can be assigned to any particular track section. Pairs of track sections and time periods are considered as objects in the SPP. The constraints of the SPP are the conflicts among different pairs of track sections and time periods that cannot be put together in the solution. A set of trains without any priority or with equal priority will address a unit-cost SPP, whereas with different priority will address multi-cost SPP. In [49], Xu and Zhang proposed a path set packing problem, is a variant of a SPP, with the goal is to find a maximum number of edge-disjoint paths in a given graph, and the goal was achieved using a polynomial time optimal algorithm.

Several other versions of SPP are reported in the literature. In [27], Weijia et al. studied a m-set packing problem, which is an extension of SPP, where the size of the set is bounded by m, and was solved using an efficient parametrized algorithm. In [38], Tri-Dung studied a complete set packing problem (CSPP). The CSPP has applications in combinatorial auctions and cooperative game theory. A winner determination problem in spectrum auctions and the coalition structure generation problem in coalition skill games is modelled as the CSPP and solved it using a fast approximation algorithm. In [23], Gulek and Toroslu studied a tree-like weighted set packing problem and solved it using a dynamic programming algorithm. This problem has been proven to be $\mathcal{P}$ and the complexity is $\mathcal{O}(k^2n)$. This problem has application in the

task assignment of hierarchical organizations. In [45], Maxim and Justin studied a maximum un-weighted set packing problem in which set cardinality is upper bounded by a constant k. The goal of the problem is to find a packing with maximum cardinality. A large neighborhood local search algorithm is proposed to solve this problem. In [21], Gottlob and Greco presented a maximum-weight set packing problem which mapped a winner determination problem in combinatorial auctions in which it determines the allocation of items to the bidders such that the sum of the accepted bid prices is maximum. In [25], Olga and Ralf proposed a generalization of odd set inequalities for a SPP for hyper-graph by employing cliques and odd set inequalities. In the context of hyper-graph, a SPP becomes an edge packing problem.

In this paper, we present an evolutionary algorithm (EA) based hyper-heuristic (EA-HH) framework to solve the SPP. The EA is inspired by the EA/G approach proposed in [50]. In the EA-HH approach, the concept of estimation of the distribution of the solutions in the solution space is used to estimate the distribution of heuristics in the search space of heuristics. Hyper-heuristics [5] are recently developed methods that operate directly on the search space of the heuristics and are independent of the problem structure. In general, almost, all meta-heuristics operate directly on the solution space of the problem under consideration and their performance may change on different problems in the same domain, and it may outperform/underperform with a small change in the same problem. A hyper-heuristic is considered as an automated heuristic search method that provides sufficient flexibility to the process of selection, generating, combining or adapting several simple or problem specific heuristics to efficiently solve the computationally hard problem. Other important aspects of any heuristic are the scalability, stability, robustness, and generality. A well-designed approach can have the ability to perform well irrespective of problem domains. The novelty of the proposed approach is that it uses univariate marginal distribution model and population based incremental learning model to initialize and update, respectively, the probability vector which estimates the distribution of heuristics in the search space of the heuristics. Further, to investigate the scalability and stability of the EA-HH approach, the minimum weight dominating set (MWDS) problem is considered which is also a class of subset selection problem but has opposite objective than the SPP. The EA-HH approach is applied to solve the MWDS problem with the purpose to investigate the scalability and generality capability.

The remainder of this paper is organized as follows. Section 2 and Section 3 present an

overview of the EA/G approach and hyper-heuristics, respectively. Section 4 explains hyper-heuristics method. Section 5 describes the proposed EA-HH approach for the SPP. The computational results are reported in Section 6. Section 7 presents the MWDS problem, its components, and the computational results. Section 8 concludes the paper.

2. Overview of evolutionary algorithm with guided mutation (EA/G)

The EA/G [50] is a recent addition to the class of evolutionary algorithms. It was developed by Zhang et al. [50] in 2005 with the intention of overcoming the shortcomings of traditional GAs [19] and (EDAs) [4, 31]. In other words, the purpose was to take advantage of the features of GAs and EDAs together. GAs use the current location information of the solutions found so far in the solution space to generate a new offspring through standard operators such as crossover and mutation. GAs do not use, directly, the global statistical information in the process of generating an offspring. On the other hand, EDAs use the global statistical information, which is stored in the form of probability, to generate a new offspring. A new offspring is generated by sampling the probability vector. In contrast to GAs, EDAs do not directly use location information in the process of generating a new offspring.

In the EA/G, the *guided mutation (GM)* operator uses the features of both GAs and EDAs to generate a new offspring. The GM operator generates a new offspring either directly copying from the best solution or randomly sampling the probability vector.

More recently, several modified and improved versions of EA/G were developed and tested on a different domain of combinatorial optimization problems. In the study in [11], an improved version of EA/G was proposed to solve the SPP. The problem domain knowledge is used to initialize the probability of objects in the set. In the study in [8], the size of the *parent* was modified, whereas, in the study in [9], both the size of the *parent* and the number of solutions to update the probability vector were modified. In [10], the authors modified the original EA/G and applied it to an order acceptance and scheduling problem, which is a subset and permutation problem. For detailed study, please refer to [24, 11, 8, 10, 9].

3. Overview of hyper-heuristic

A hyper-heuristic is a heuristic search technique that automates the search process and also allows to combine or generate a suitable problem solver in each generation [5]. Generally, hyper-heuristic consists of two levels, namely; *higher level* and *lower level* [5]. A High-level

search strategy and a set of low-level heuristics reside at the higher level and lower level, respectively. The high-level search strategy directly operates on the search space of the heuristics. In each iteration, high-level strategy applies a heuristic selection or heuristic generation method to select/generate a heuristic from the lower level and subsequently, this heuristic is applied to generate a new solution. The lower level consists of problem-specific/generic heuristics, called low-level heuristics that directly operate on the solution space of the problem under consideration. But the major challenge in hyper-heuristics is the selection of a heuristic from the heuristics pool. In the literalure, several strategies have been developed to select/generate a heuristic from the heuristics pool such as simple random selection [13], greedy selection [13] and a choice function [13].

Several other versions of hyper-heuristic are presented in the literature. In [3], a tensor-based machine learning technique is proposed for selection hyper-heuristic. Further, a random heuristic selection with the naive, improving and equal move acceptance method are combined in a selection of low-level heuristic. In the study in [44], a Monte Carlo tree search method is proposed to generate a sequence of low-level heuristics. Further, the Monte Carlo is coupled with a population of solutions to improve the effectiveness of the search tree. In [22], Jacomine et al. put an effort to investigate the concept of heuristic space diversity and management of it under various strategies under meta-hyper-heuristic approach. In [2], Shahriar et al. applied hyper-heuristic with the combination of Monte-Carlo to solve the multi-mode resource-constrained multi-project scheduling problem. The hybrid version of the proposed approach is a combination of Monte-Carlo tree search, neighborhood moves, memetic algorithms, and hyper-heuristic. In [1], Almutairi et al. investigated six different existing selection hyper-heuristics namely: Sequence-based selection hyper-heuristic, Dominance-based and random descent hyper-heuristic, Robinhood (round-robin neighbourhood) hyper-heuristic, Modified choice function, Fuzzy late acceptance-based hyper-heuristic [26], and Simple Random-Great Deluge in HyFlex problem domains such as 0-1 Knapsack, Quadratic Assignment, and Max-Cut problem. In [29], Ahmed and Ender proposed an iterated multi-stage selection hyper-heuristic and investigated on one-dimensional bin-packing, personal scheduling, permutation flow-shop, travelling salesman problem, and vehicle routing problem. In [12], a reinforcement learning technique is incorporated in hyper-heuristic and the performance is evaluated on six different problem domains. Several state-of-the-art hyper-heuristics [6, 12] are also developed

for other domain problems.

4. Proposed evolutionary algorithm based hyper-heuristic

This paper presents a novel evolutionary algorithm based hyper-heuristic inspired by the EA/G approach proposed by Zhang et al. [50]. As mentioned in Section 2, EDAs work on the concept of the estimation of the distribution of the promising solutions in the solution space. In each generation, it estimates the distribution of the promising solutions in the solution space and subsequently stores their distribution in the form of probability. On the other side, the higher level strategy of EA-HH estimates the distribution of promising heuristics in the search space. The main components of the EA-HH are discussed below.

4.1. Higher level of the hyper-heuristic

An evolutionary algorithm is employed as a high-level strategy at the higher level of the hyper-heuristic. The main components of the higher level are explained in the following sub-sections:

4.1.1. High-level search strategy

The high-level of a hyper-heuristic is a search strategy that operates on the search space of the heuristics and selects or generates a heuristic by combining two or more heuristics from the heuristics pool. This section presents search strategy that is adopted at the higher level of the hyper-heuristic. Matrix in 6 is a heuristics population matrix of $\mathbf{N_p} \times \mathbf{N_H}$, where $\mathbf{N_p}$ is the heuristics population size and $\mathbf{N_H}$ is the number of low-level heuristics. The value of $f_{i\mathcal{H}_j}$, $i = 1, 2, \ldots, \mathbf{N_p}$, $j = 1, 2, \ldots \mathbf{N_H}$, $\mathcal{H}_j \in$ {LH_1, LH_2, ..., LH_$\mathbf{N_H}$} is determined by using Equation (7).

$$\mathcal{F}_{Credit} = \begin{bmatrix} f_{1\mathcal{H}_1} & f_{1\mathcal{H}_2} & \cdots & f_{1\mathcal{H}_{\mathbf{N_H}}} \\ f_{2\mathcal{H}_1} & f_{2\mathcal{H}_2} & \cdots & f_{2\mathcal{H}_{\mathbf{N_H}}} \\ \vdots & \vdots & \ddots & \vdots \\ f_{\mathbf{N_p}\mathcal{H}_1} & f_{\mathbf{N_p}\mathcal{H}_2} & \cdots & f_{\mathbf{N_p}\mathcal{H}_{\mathbf{N_H}}} \end{bmatrix} \tag{6}$$

$\mathcal{F}_{Credit}$ is a credit matrix that stores the fitness difference between the new solution and the current solution generated by each low-level heuristic in each generation in the search process. The difference is calculated as follows:

$$f_{i\mathcal{H}_j} = \begin{cases} f_{New} - f_{Current}, & \text{if } (f_{New} - f_{Current}) > 0 \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

where $f_{Current}$ and f_{New} are the fitness of the current solution and the fitness of the solution generated in the neighbourhood of the current solution, respectively.

4.1.2. Initialization and update of probability vector $\mathcal{P}_{Heuristic}$

In [36, 50], the univariate marginal distribution (UMD) model was used to estimate the distribution of the candidate solutions in the population. In the UMD model, each decision variable is treated as an independent variable and the conditional probability of a variable does not depend on the probability of other variables. In the context of heuristics, the UMD model estimates the distribution of heuristics in the search space of the heuristics. Each heuristic is treated as an independent variable and probability vector $\mathcal{P}_{Heuristic} = \{\mathcal{P}_{\mathcal{H}_1}, \mathcal{P}_{\mathcal{H}_2}, \ldots, \mathcal{P}_{\mathcal{H}_k}\} \in [0, 1]^{\mathbf{N_H}}$ characterizes the distribution of the promising heuristics in the search space, where $\mathcal{P}_{\mathcal{H}_k}$ is the probability of $\mathcal{H}_k{}^{\text{th}}$ heuristic in the heuristics pool. Equation (8) is used to assign a probability to each heuristic $\mathcal{H}_i$.

$$\mathcal{P}_{\mathcal{H}_i} = \frac{\sum_{j=1}^{\mathbf{N_p}} f_{j\mathcal{H}_i}}{\sum_{k=1}^{\mathbf{N_H}} \sum_{j=1}^{\mathbf{N_p}} f_{j\mathcal{H}_k}}, \quad i = 1, 2, \ldots \mathbf{N_H}, \tag{8}$$

After the probability vector $\mathcal{P}_{\mathcal{H}_i}$ has been initialized, at each generation g the probability distribution of each heuristic $\mathcal{H}_i$ is updated using the best $\frac{\mathbf{N_p}}{2}$ fitness values from Equation (6). A population-based incremental learning algorithm (PBILA) [4] is used to update the probability vector. Equation (9) updates the probability vector at each generation. In Equation (9), ζ is a learning variable that learns from the past statistical information stored in the form of the probability vector in the current population. The larger the value of ζ, the greater is the contribution from the current population, whereas the smaller the value of ζ, the greater is the contribution from the probability vector.

$$\mathcal{P}_{\mathcal{H}_i} = (1 - \zeta) \times \mathcal{P}_{\mathcal{H}_i} + \zeta \times \frac{\sum_{k=1}^{\frac{\mathbf{N_p}}{2}} f_{k\mathcal{H}_i}}{\sum_{k=1}^{\mathbf{N_H}} \sum_{j=1}^{\mathbf{N_p}} f_{j\mathcal{H}_k}}, i = 1, 2, \ldots \mathbf{N_H}, \tag{9}$$

The worst case time complexity of both the initialization and update of the probability vector is $\mathcal{O}(\mathbf{N_p}^2\mathbf{N_H})$.

4.1.3. Heuristic selection rule

Heuristic selection is an important component of any hyper-heuristic. It determines the direction of the search process in the search space. An intelligent heuristic selection method can take the search process toward the global best solution, whereas random selection may lead the search process toward the local best solution or may require a greater number of generations to reach the global best. Several heuristic selection rules have been developed such as random selection, choice function, and greedy selection. In this paper, we propose roulette wheel selection (RWS) method based heuristic selection rule. However, we tried all the mentioned methods but the RWS has, overall, the best performance. Therefore, we choose the RWS method as a heuristic selection rule.

Equation (8) estimates the distribution of the heuristics in terms of their performance in each generation and stores the distribution in the form of probability in the probability vector $\mathcal{P}_{Heuristic}$. The probability $\mathcal{P}_{\mathcal{H}_i}$ of heuristic $\mathcal{H}_i$ is directly proportional to its performance. The RWS method, also called the fitness proportionate selection method, considers the probability of each heuristic as a fitness of heuristic $\mathcal{H}_i$ and its probability of being selected is calculated as

$$\varepsilon_i = \frac{\mathcal{P}_{\mathcal{H}_i}}{\sum_{j=1}^{\mathbf{N_H}} \mathcal{P}_{\mathcal{H}_j}}, \; i = 1, 2, \ldots, \mathbf{N_H} \tag{10}$$

The selection of a heuristic is determined by the RWS method. The pseudo-code of the RWS method is presented in Algorithm 1.

Algorithm 1 Roulette wheel selection method

```
heu ← 0;
r ← rand(0, 1);
while (ε_heu ⩽ r) do
    heu ← heu+1;
end while
return H_heu
```

4.2. *Lower level of the hyper-heuristic*

The lower level of the hyper-heuristic is a set of low-level, can be standard or problem specific, heuristics that directly operates on the search space of the solution of the problem under consideration. Table 1 presents six low-level heuristics.

Table 1: Low-level heuristics

Heuristic	Generation of heuristic	Heuristic	Generation of heuristic
LH_1	*Crossover1* (Section 4.2.1)	LH_2	*Crossover2* (Section 4.2.1)
LH_3	*Crossover3* (Section 4.2.1)	LH_4	*Construction heuristic* (Section 4.2.2)
LH_5	*Guided mutation* (Section 4.2.3)	LH_6	*One_one* (Section 4.2.4) + *One_two* (Section 4.2.5)

4.2.1. *Crossover heuristics*

Three versions of uniform crossover operator (UXO) [46] are developed to generate an offspring. In the first version, called *Crossover1* (LH_1), the global best solution and the current solution are used as two parents and a child solution is generated by applying UXO. In UXO, for each location a random number is generated uniformly in the interval [0, 1]. If the number is smaller than the predefined value $\mathbf{C_P}$, then the corresponding object from the best solution is added to the child solution; otherwise, the corresponding object from the current solution is added.

In the second version of UXO, called *Crossover2* (LH_2), the current solution and a solution different from the current solution is chosen using binary tournament selection (BTS) method and subsequently, as in *Crossover1*, a new child solution is generated. In the third version of UXO, *Crossover3* (LH_3), two parents are selected randomly using BTS method and subsequently, similar to *Crossover1*, a new child is generated. Each UXO version has an advantage and also helps to maintain diversification in the solution population. *Crossover1* and *Crossover2* have both exploration and exploitation features, whereas *Crossover3* has only exploration ability.

4.2.2. *Construction heuristic (LH_4)*

The purpose of the construction heuristic is to maintain diversity in the population. It performs exploration. At each iteration, an object, $i \in \mathcal{I}$, is added to partial solution, say $\mathcal{S}_p$, with the probability 0.50. Subsequently, the solution is passed through the repair operator (Section 5.3)

to make it feasible if not and thereafter through the improvement operator (Section 5.4) for the possible improvement. The pseudo-code of the *Construction heuristic* is presented in Algorithm 2.

Algorithm 2 Construction heuristic (LH_4)

```
1: S_p ← φ;
2: for each object i ∈ I in some random order do
3:     r_1 ← rand(0, 1);
4:     if (r_1 ⩽ 0.50) then
5:         S_p ← S_p ∪ {i};
6:     end if
7: end for
8: return S_p;
```

4.2.3. Guided mutation (LH_5)

In any heuristic, it is always good to take advantages of problem domain knowledge to improve their performance. The proposed guided mutation (GM) heuristic uses the cost and the cardinality of conflict set of objects to give more weight to an object whose conflict size is smaller. For each object $i \in \mathcal{I}$, $\aleph_i$ and $|\aleph_i|$ are the conflict set and the conflict count (number of objects conflicting with object i), respectively. An empty set indicates that the particular object has no conflicting objects. The pseudo-code of determining the conflict set of objects is presented in Algorithm 3.

Algorithm 3 Determination of the conflict set of each object $i \in \mathcal{I}$

```
1: for (Each object i ∈ I) do
2:     ℵ_i ← φ;
3:     for (j = 1 to M) do
4:         if (|{i} ∩ T_j| > 0) then
5:             ℵ_i ← {ℵ_i ∪ T_j} \ {i};
6:         end if
7:     end for
8: end for
```

For initialization of the probability vector, an UMD model is used to estimate the distribution of the candidate solutions in the solution space of the problem. A ratio, say v_i, of cost, say c_i, and $|\aleph_i|$ of an object i is added to increase the probability. A probability vector, say $\mathcal{P}_{Mutation} = \{p_1, p_2, \ldots, p_{|\mathcal{I}|}\} \in [0, 1]^{|\mathcal{I}|}$, is inilialized where p_i is the probability of the i^{th} object in set $\mathcal{I}$ and $|\mathcal{I}|$ is the cardinality of set $\mathcal{I}$. The pseudo-code of the initialization of the probability vector using the UMD model is given in Algorithm 4.

Algorithm 4 Probability vector initialization

1: Compute $v_i \leftarrow \frac{c_i}{|\aleph_i|}, \forall i \in \mathcal{I}$;
2: Compute $y_i \leftarrow$ number of initial solutions containing object $i, \forall\, i \in \mathcal{I}$;
3: Compute $p_i \leftarrow \frac{(y_i+v_i)}{(\mathbf{N_p}+v_i)}, \forall i \in \mathcal{I}$;

Further, the population-based incremental learning (PBIL) model [4] is used to update the probability vector. At each generation g, a parent set, say *Parent(g)*, is formed by selecting the best $\frac{\mathbf{N_p}}{2}$ candidate solutions from the current population *Pop(g)*. Variable ζ is the learning rate, which changes dynamically with the generation. As ζ controls the contributions of location and global information and helps to maintain a balance between exploitation and exploration. The pseudo-code of the probability update vector using the PBIL model is given in Algorithm 5.

Algorithm 5 Probability vector update

1: Compute $z_i \leftarrow$ number of solutions in *Parent(g)* containing object $i, \forall i \in \mathcal{I}$;
2: Compute $p_i \leftarrow (1-\zeta) \times p_i + \zeta \times \frac{z_i}{\frac{\mathbf{N_p}}{2}}, \forall i \in \mathcal{I}$;

After the probability vector $\mathcal{P}_{Mutation}$ is updated, the GM operator is applied to generate a new solution. Set $\mathcal{S}_p$, initially empty, is a partial solution in which objects are added either by sampling the probability vector $\mathcal{P}_{Mutation}$ or directly copying from the global best solution, say $\mathcal{B}_s$. Variable $\beta \in (0,\ 1)$ is a control parameter that controls the contribution from probability vector $\mathcal{P}_{Mutation}$ and the global best solution $\mathcal{B}_s$. The large the value of β, the greater is the number of objects included by sampling the probability vector $\mathcal{P}_{Mutation}$, whereas the smaller the value of β, the greater is the number of objects copied from $\mathcal{B}_s$. The pseudo-code of the GM operator is presented in Algorithm 6.

Algorithm 6 Guided mutation

$\mathcal{S}_p \leftarrow \phi$;
for each object $i \in \mathcal{I}$ in some random order **do**
  $r_1 \leftarrow rand(0, 1)$;
  if ($r_1 < \beta$) **then**
    $r_2 \leftarrow rand(0, 1)$;
    if ($r_2 < p_i$) **then**
      $\mathcal{S}_p \leftarrow \mathcal{S}_p \cup \{i\}$;
    end if
  else
    if ($i \in \mathcal{B}_s$) **then**
      $\mathcal{S}_p \leftarrow \mathcal{S}_p \cup \{i\}$;
    end if
  end if
end for
return $\mathcal{S}_p$;

4.2.4. *One_one swap heuristic (LH_6)*

One_one swap heuristic is a modified version of the *1-1 exchange local search* presented in [11]. It also operates on the first improvement strategy. In each iteration, it swaps one object from solution $\mathcal{S}_p$ with an object not in the solution ($\mathcal{I}^* = \mathcal{I} \setminus \mathcal{S}_p$) if it improves the solution. A function, say $Find_{1\_1}(\mathcal{S}_p)$, identifies an object, say j, that does not violate the feasibility constraint. $Swap_{1\_1}(\mathcal{S}_P, i, j)$ is a function that replaces object i with objects j, i.e., $\mathcal{S}_p^* = (\mathcal{S}_p \cup \{j\}) \setminus \{i\}$. *Fitness()* is a function that calculates the sum of the weights of the objects in solution $\mathcal{S}_p^*$. If the new solution $\mathcal{S}_p^*$ is better than $\mathcal{S}_p$, then the solution is updated; otherwise, it moves to the next object in $\mathcal{S}_p$. This process continues until all objects in solution $\mathcal{S}_p$ are attempted. The *1-1 exchange local search* is applied at most 4 successful exchanges, whereas the *One_one* swap exchanges objects until all objects in set $\mathcal{S}_p$ are attempted. The pseudo-code of the *One_one* swap heuristic (LH_6) is presented in Algorithm 7.

Algorithm 7 *One_one* swap heuristic

$\mathcal{F} \leftarrow Fitness(\mathcal{S}_p)$;
$\mathcal{I}^* \leftarrow \mathcal{I} \setminus \mathcal{S}_p$;
$\mathcal{C}_s \leftarrow |\mathcal{S}_p|$;
$i \leftarrow 1$;
while $(i \leqslant \mathcal{C}_s)$ **do**
    $j \leftarrow Find_{1\_1}(\mathcal{I}^*)$;
    $\mathcal{S}_p^* \leftarrow Swap_{1\_1}(\mathcal{S}_{p_i}, j)$;
    $\mathcal{F}^* \leftarrow Fitness(\mathcal{S}_p^*)$;
    if $(\mathcal{F}^* > \mathcal{F})$ **then**
        $\mathcal{S}_p \leftarrow \mathcal{S}_p^*$;
        $\mathcal{F} \leftarrow \mathcal{F}^*$;
        $i \leftarrow 1$;
    else
        $i \leftarrow i$+1;
    end if
end while

4.2.5. *One_two swap heuristic (LH_6)*

One_two swap heuristic is a modified version of the *1-2 exchange local search* presented in [11]. Similar to *One_one* swap heuristic, function $Find_{1\_2}(\mathcal{I}^*)$ identifies two objects, say i and j, in set $\mathcal{I}^*$ that are not in solution $\mathcal{S}_p$ and do not violate the feasibility constraint. Function $Swap_{1\_2}(\mathcal{S}_P, \mathcal{S}_{p_i}, j, k)$ replaces object $\mathcal{S}_{p_i}$ with objects j and k, i.e., $\mathcal{S}_p^* = (\mathcal{S}_p \cup \{j, k\})$. If the new solution $\mathcal{S}_p^*$ is better than $\mathcal{S}_p$, then the solution is updated; otherwise, it moves to the next object in $\mathcal{S}_p$. This process continues until all the objects in $\mathcal{S}_p$ are attempted. The *1-2 exchange local search* exchanges at most 6 objects, whereas the *One_two* swap exchanges objects until all objects in set $\mathcal{S}_p$ are tried. The pseudo-code of the *One_two* swap heuristic (LH_6) is presented in Algorithm 8.

Algorithm 8 *One_two* swap heuristic

$\mathcal{F} \leftarrow Fitness(\mathcal{S}_p)$;
$\mathcal{I}^* \leftarrow \mathcal{I} \setminus \mathcal{S}_p$;
$\mathcal{C}_s \leftarrow |\mathcal{S}_p|$;
$i \leftarrow 1$;
while $(i \leqslant \mathcal{C}_s)$ **do**
    $(j, k) \leftarrow Find_{1\_2}(\mathcal{I}^*)$;
    $\mathcal{S}_p^* \leftarrow Swap_{1\_2}(\mathcal{S}_P, \mathcal{S}_{p_i}, j, k)$;
    $\mathcal{F}^* \leftarrow Fitness(\mathcal{S}_p^*)$;
    if $(\mathcal{F}^* > \mathcal{F})$ **then**
        $\mathcal{S}_p \leftarrow \mathcal{S}_p^*$;
        $\mathcal{F} \leftarrow \mathcal{F}^*$;
        $i \leftarrow 1$;
    else
        $i \leftarrow i$+1;
    end if
end while

5. EA-HH approach for the SPP

The proposed EA-HH approach for the SPP was inspired by the approaches presented in [50] for the maximum clique problem and classification of hyper-heuristics [5]. The pseudo-code and flowchart of the EA-HH approach are presented in Algorithm 9 and Figure 1, respectively. The main components of the EA-HH approach are presented below.

Algorithm 9 Evolutionary algorithm based hyper-heuristic

At generation $g \leftarrow 0$, an initial population *Pop(0)* consisting of $\mathbf{N_p}$ solutions is generated randomly;
Apply each low-level heuristic $\mathcal{H}_i \in \{\text{LH\_1}, \text{LH\_2} \ldots, \text{LH\_}\mathbf{N_H}\}$ $\mathbf{N_p}$ number of times to form $\mathbf{N_p} \times \mathbf{N_H}$ fitness matrix $\mathcal{F}_{Credit}$ (Equation (6));
Select the best $\mathbf{N_p}$ number of solutions from $(Pop(g) + \mathbf{N_p} \times \mathbf{N_H})$ solutions to form population *Pop(0)*;
Initialize the probability vector $\mathcal{P}_{Mutation}$ (Algorithm 4) and $\mathcal{P}_{Heuristic}$ (Equation (8));
while $(g < \mathbf{GEN})$ **do**
    Select the best $\frac{\mathbf{N_p}}{2}$ solution from *Pop(g)* to form a *Parent(g)*;
    Update $\zeta(g)$ (Equation (11)), $\beta(g)$ (Equation (12)), $\mathcal{P}_{Mutation}$ (Algorithm 5), and $\mathcal{P}_{Heuristic}$ (Equation (9));
    for (i =1 to $\frac{\mathbf{N_p}}{2}$) **do**
        Select a heuristic $\mathcal{H}_i \in \{\text{LH\_1}, \text{LH\_2} \ldots, \text{LH\_}\mathbf{N_H}\}$ by applying the RWS method on the probability vector $\mathcal{P}_{Heuristic}$;
        Apply heuristic $\mathcal{H}_i$ to generate a new solution;
        Update the fitness matrix $\mathcal{F}_{Credit}$ using First in first out scheme;
    end for
    Add $\frac{\mathbf{N_p}}{2}$ newly generated solutions to *Parent(g)* to form *Pop(g+1)*;
    $g \leftarrow g + 1$;
end while
return Best solution;

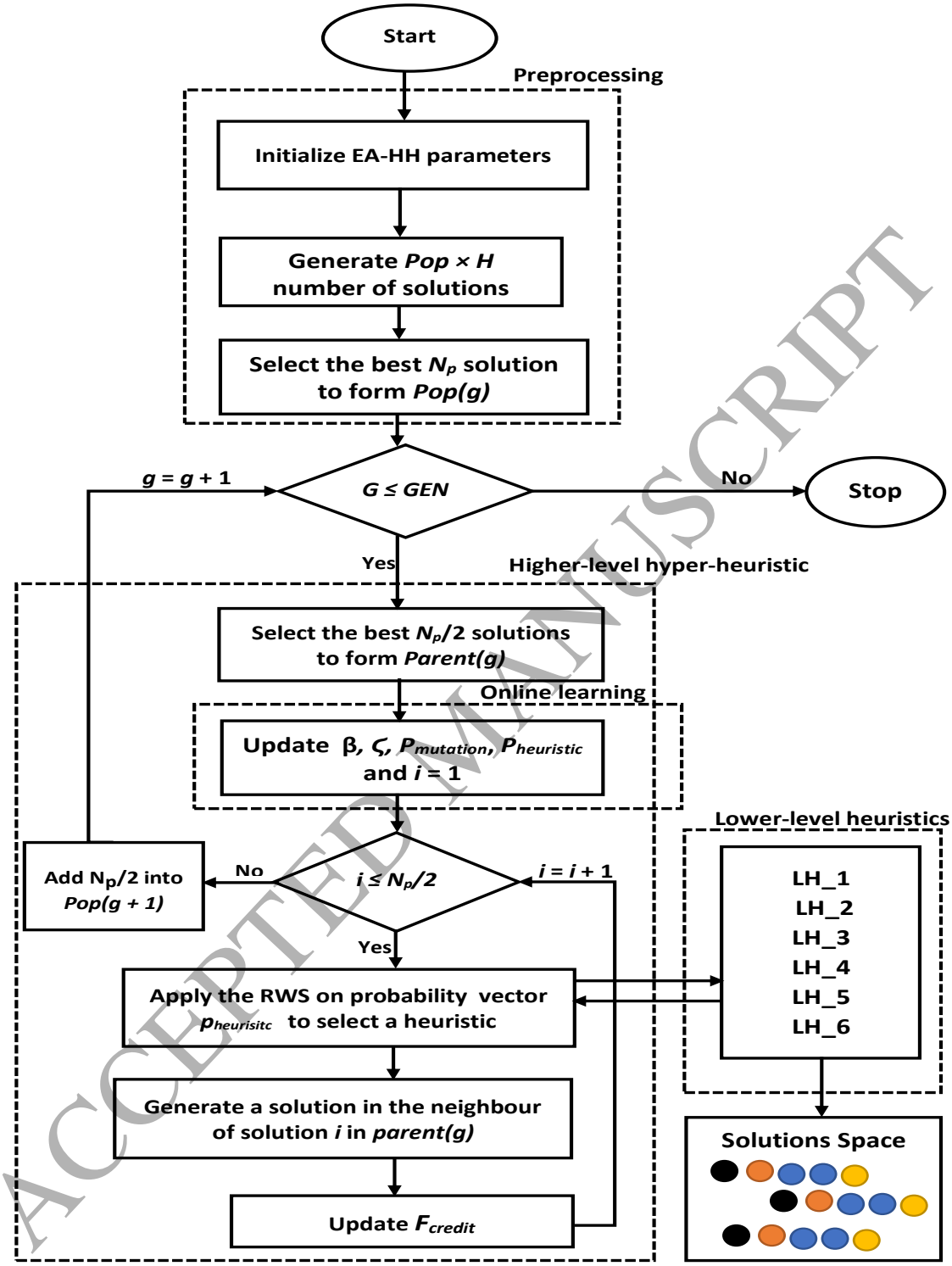


Figure 1: Flowchart of the EA-HH approach

5.1. Solution encoding

Solution encoding is an important part of any meta-heuristic algorithm and it directly affects the computational performance. As the SPP is a class of subset selection problems; therefore, a subset encoding is adopted to represent a solution. Each solution, say $\mathcal{S}_p$, is represented directly by the objects it contains, i.e., $\mathcal{S}_p = \{o_1, o_2, \ldots, o_{|\mathcal{S}_p|}\}$, where $|\mathcal{S}_p|$ is the cardinality of the solution $\mathcal{S}_p$.

5.2. Initial solution

An initial population is generated randomly. Each object is included randomly in the partial solution, say $\mathcal{S}_p$, with equal probability. However, only those objects that satisfy the feasibility criteria are included in $\mathcal{S}_p$.

5.3. Repair operator

The proposed repair operator is a modified version of the *Repair Operator* presented in [11]. A solution $\Gamma = \{\Gamma_1, \Gamma_2, \ldots, \Gamma_{|\Gamma|}\}$ is infeasible if $\sum_{i=1}^{|\Gamma|} |\aleph_{\Gamma_i} \cap \Gamma| > 0$, where $\aleph_{\Gamma_i}$ is the conflict set of an object Γ_i. Each infeasible solution Γ generated through heuristic $\mathcal{H}_i$ is passed through the repair operator to make it feasible. In the *Repair Operator* of [11], the conflicting objects are deleted with the help of roulette wheel selection method where the probability of selection of an object is proportional to the ratio of $|\aleph_i|$ to the weight of object i, i.e., c_i, whereas in the modified repair operator conflicting objects are selected in the order in which they are presented in the solution. The pseudo-code of the repair operator is presented in Algorithm 10.

Algorithm 10 Repair operator

```
1: i ← 1;
2: while (i ⩽ |Γ|) do
3:     δ ← ℵ_{Γ_i} ∩ Γ;
4:     if (|δ| > 0) then      ▷ |δ| will be greater than zero if there is at least one object conflicting with object Γ_i in Γ
5:         Γ ← Γ − δ;          ▷ Remove the conflicting objects from Γ
6:         |Γ| ← |Γ|;          ▷ Update the cardinality of set Γ
7:     end if
8:     i ← i+1;
9: end while
```

5.4. Improvement operator

Each feasible solution is passed through the improvement operator to further improve the solution quality. The proposed improvement operator is the modified version of the *Improvement Operator* presented in [11]. Suppose δ is a set of objects which are not in solution Γ and $\aleph_{\Gamma_i}$ is the conflict set of an object Γ_i. In [11], roulette wheel selection method is used to add unpacked object to the solution, where the probability of selection of object δ_i is directly proportional to the ratio of the weight of object i to $|\aleph_i|$, whereas the proposed improvement operator selects an object in the same order as they presented in δ. The improvement operator iteratively attempts to add an unpacked object without violating the feasibility criteria. The pseudo-code of the improvement operator is presented in Algorithm 11.

Algorithm 11 Improvement operator

1: $\delta \leftarrow (\mathcal{I} - \Gamma)$;
2: **for** (For each object $\delta_i \in \delta$) **do**
3: **if** ($|\aleph_{\delta_i} \cap \Gamma| = 0$) **then**
4: $\Gamma \leftarrow \Gamma \cup \delta_i$;
5: $\delta \leftarrow (\delta \setminus \aleph_{\delta_i}) \setminus \{i\}$;
6: **end if**
7: **end for**

6. Computational results for the SPP

The proposed EA-HH approach was implemented in **C language** and executed on a **Core 2 Duo system with 2 GB RAM** under **Ubuntu operating system in a 3.0 GHz** environment. A **gcc 4.4.4-10** compiler with `-O3` flag was used to compile the **C program**. For the fair comparison, the same system is used as that of used in the study in [11]. The EA-HH parameters and their values are listed in Table 2, and all these values were fixed empirically after numerous trials. In this study, two types of random test instances: unit-cost and multi-cost are used to investigate the performance of the proposed approach. The characteristics of the test instances are reported in Tables 7 to 9. In these three tables, for each instance, the columns with the headings `Var`, `Cnst`, `Density`, `Max_One`, and `Weight` show the number of objects ($|\mathcal{I}|$), number of constraints ($|\mathcal{J}|$), percentage of non-null elements in the constraint matrix, size of the largest set $|\mathcal{T}_j|$ over all $j \in \{1, 2, \ldots, \mathbf{M}\}$, and weight $c_i \in$ [1-1] or [1-20] of object $i \in \mathcal{I}$, respectively. All the test instances can be downloaded from `http://www3.inrets.fr/~delorme/Instances-fr.html`. In Table 7, the column with the heading `Opt` lists the optimal values returned by the **CPLEX 6.0 solver** and in Tables 8 and 9, the column with the heading BKV shows the best

known value (BKV) obtained by **CPLEX 6.0 solver** indicated with an asterisk ("*"). For the GRASP [14], ACO [17], EA/G [11], and the proposed EA-HH approach, the columns with the headings **Best, Avrg**, and **ATET** show the best solution, average solution, and average execution time in seconds, respectively. For the EA-HH approach, we report also the number of objects in the best solution and the average number of objects in the average solution.

Table 2: Parameters and their values for the EA-HH approach

Parameter	Value	Description
$\mathbf{N_p}$	30	Population size
ζ	$\in [0.001, 0.99]$	Learning rate
β	$\in [0.001, 0.99]$	Is the control parameter that controls the contribution from the current population and the best solution
$\mathbf{N_H}$	6	Number of low-level heuristics
RUN	16	Number of independent runs for each instance for the SPP
RUN	1	Number of independent runs for each instance for the MWDS problem
GEN	100	Number of generations in each run
$\mathbf{C_P}$	0.50	Uniform crossover rate

6.1. *Effect of dynamic selection of β and ζ parameters*

Parameter tuning is one of the most challenging and difficult tasks in any meta-heuristic approach. Therefore, an intelligent meta-heuristic always attempts to be as independent of parameters as possible.

$$\zeta(g) = \zeta_{min} + \frac{\zeta_{max} - \zeta_{min}}{\mathbf{GEN}} \times g \tag{11}$$

$$\beta(g) = \beta_{min} + \frac{\beta_{max} - \beta_{min}}{\mathbf{GEN}} \times g \tag{12}$$

The values of ζ and β change dynamically with generation g. It also helps to maintain a balance between exploration and exploitation. In the beginning of the generation, the search process performs the exploration whereas at the end of the generation performs exploitation. Dynamic nature of parameter ζ allows to learn from the global statistical information, whereas at the end of the generation, it allows to learn from the current location in the population. Similarly, a smaller value of β forces a contribution from the global best solution, whereas a larger value forces a contribution from the current population. In Equations (11) and (12), g, ζ_{min} and ζ_{max}, and **GEN** are the current generation, minimum and maximum values of ζ, and the maximum number of generations, respectively.

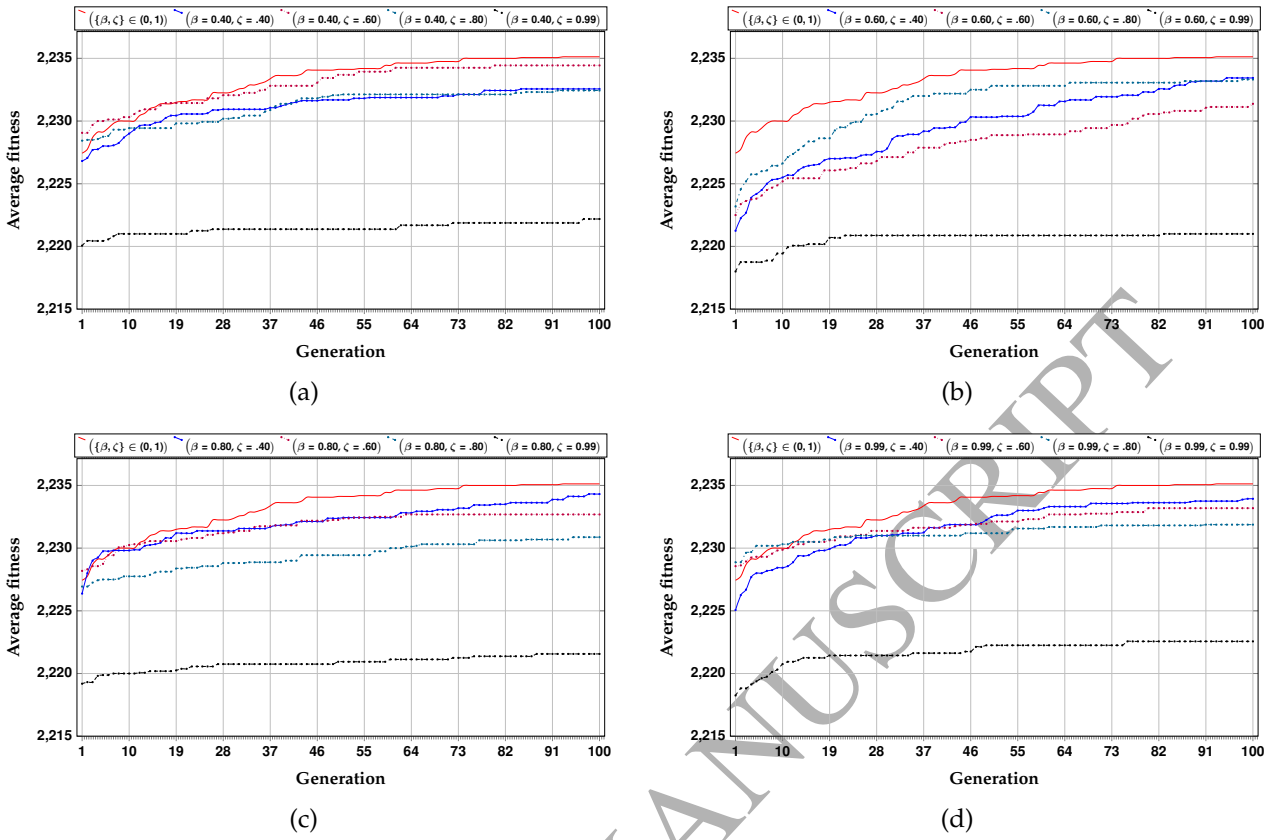


Figure 2: Convergence behavior of the EA-HH approach with different values of β and ζ on instance `pb500rnd09`

In Figures 2 and 3, instances `pb500rnd09` and `pb1000rnd07` are used to show the influence of the dynamic selection of parameters ζ and β on the solution quality. These figures show the convergence speed of each combination of parameter values. The values of ζ, $\beta \in \{0.40, 0.60, 0.80, 0.99\}$, and other combinations, where the values change dynamically according to Equation (11) and Equation (12) are plotted. In Figures 2(a) to 2(d), and Figures 3(a) to 3(d) the values of β are 0.40, 0.60, 0.80, and 0.99, respectively and for each value of β four different values of $\zeta \in \{0.01, 0.40, 0.60, 0.80, 0.99\}$ are passed. The figures show that the dynamic selection of parameters β and ζ yields a better convergence speed and these values reach the BKV faster than the fixed values.

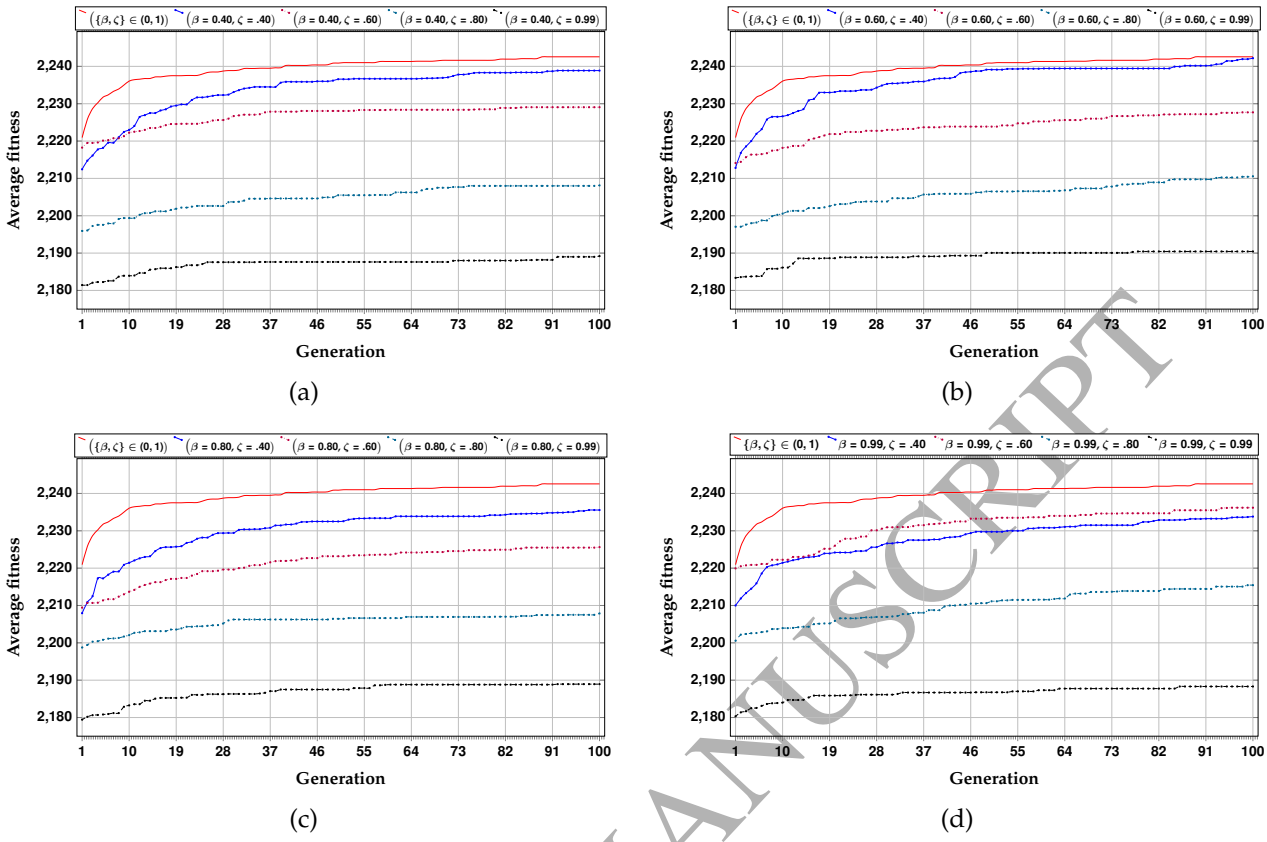


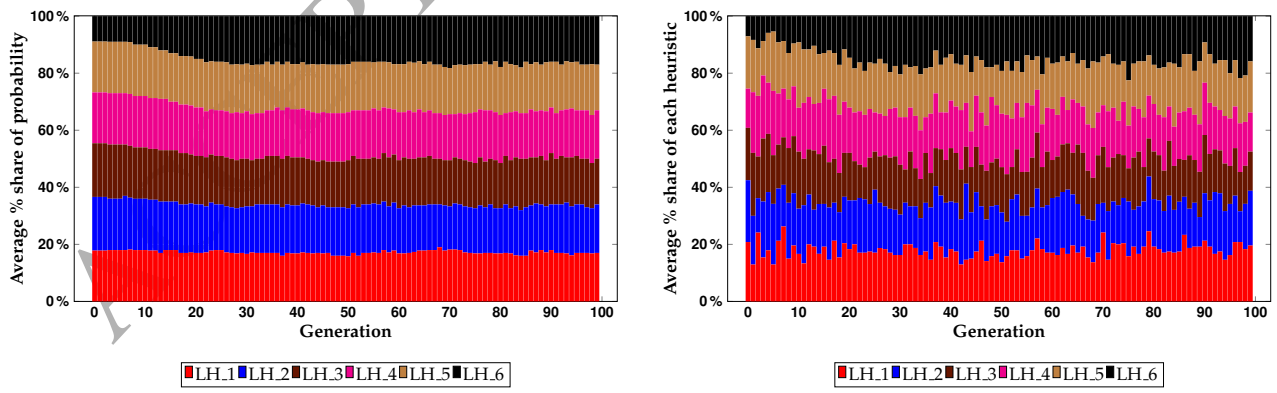
Figure 3: Convergence behaviour of the EA-HH approach with different values of β and ζ on instance `pb1000rnd07`

One of the reasons of success is that the dynamic selection of parameters allows the search process to explore different areas of search space at the beginning of the generation, and then end of the generation it allows to exploit the search area. On the other side, fixed parameter value restricts the search process to explore in the fixed direction. In these figures, we can see that plots with larger values of parameters unable to converge to the BKV, and the reason is that the search process started exploiting the search area from the beginning of generation and got stuck on local optima. Similarly, When one parameter has smaller value and other has a larger value, then the search process got stuck on local optima. Furthermore, if the parameters values are close to each other, then the search process is able to perform both exploration and exploitation. Based on experimental results and analysis it can be concluded that a proper trade-off among parameters values can take the search process towards global optimum in reasonable amount of computational time.

6.2. *Behaviour of the EA-HH approach in the search space of the heuristics*

In this section, we investigate the behavior of the components of the EA-HH approach. In Figure 4(a), the portions of the probability shared by low-level heuristics LH_1, LH_2, LH_3, LH_4, LH_5, and LH_6 in each generation for instance `pb1000rnd07` are plotted. In the figure, the *Generation* axis represents generations from 1 to 100 and the *Probability* axis represents the percentage share of the probability by each heuristic in each generation. The probability value of each heuristic is directly proportional to the performance of the particular heuristic in any generation. The probability values help in the selection of a low-level heuristic in the next generation. In the figure, it can be observed that the probability share changes with the generation and this directly affects the heuristic selection rule. The larger its probability value, the greater is the chance of the heuristic being selected in the next generation. The average probability shares of heuristics LH_1, LH_2, LH_3, LH_4, LH_5, and LH_6 are 0.160, 0.162, 0.160, 0.184, 0.174, and 0.159, respectively.

Figure 4(b) shows the distribution of heuristics in each generation. In the figure, the *Generation* axis represents the generations from 1 to 100 and the *Heuristic* axis represents the percentage share of frequencies of each heuristic in each generation. The average number of times heuristics LH_1, LH_2, LH_3, LH_4, LH_5, and LH_6 were selected was 242, 240, 255, 265, 281, and 217, respectively. The average number of frequencies of the heuristics shows that the search process is not dependent on any one of the heuristics and each heuristic contributed to the search process.



(a) Changes in probability of heuristics with generations (b) Changes in frequency of heuristics with generations

Figure 4: Changes in probability and heuristic frequency with generation on instance `pb1000rnd07`

From Figures 4(a) and 4(b), it can be observed that the % frequency of each heuristic in each

generation is directly affected by the % probability share of each heuristic in the corresponding generation. However, the average probability of the heuristics does not have direct influence on the corresponding average frequency of heuristic. The reason of this is randomness in the RWS method.

6.3. *Analysis of exploration and exploitation nature of the heuristics*

Exploration and exploitation are the important features of any heuristic. Exploratory heuristic searches new space of the search space, whereas exploitative heuristic searches in the neighborhood of the current location of the search space. Exploration tries to take the search process towards the new space, whereas exploitation exploits the current location of the search space. A delicate balance between exploration and exploitation is an important factor of any heuristic to get a satisfactory solution in a reasonable amount of computational time. On the other hand, an improper balance may take the search process towards a local optimum. Furthermore, more exploration may need more number of iterations to reach the global optimum, whereas more exploitation, may be stuck into local optimum. In the EA-HH, *Crossover1*, *Crossover2*, *Crossover3*, and *Construction* are exploratory heuristics, whereas *One_one* and *One_two* are exploitative heuristics. *Crossover1* heuristic explores a new space in the neighbor of the current global best solution, *Crossover2* heuristic explores a new space in the neighbor of one of the best $\frac{N_p}{2}$ solutions, whereas *Crossover3* heuristic explores a new space in the neighbor of one of the worst $\frac{N_p}{2}$ solutions in the population. The advantage of these heuristics is that they explore diverse new space simultaneously. *Construction* heuristic explores the search space randomly and it helps to maintain diversity in the solution population. In the case of *Guided mutation* heuristic, it doses both exploration and exploitation together. It explores the search space using the global statistical information which is stored in the form of probability, on the other hand, it exploits the search space in the neighbor of the current global best solution.

6.4. *Population evolution with generations*

Figure 5 presents the evolution of the solution population with the generations. In the figure, all the generated solutions are plotted. It can be observed that the solution population evolves with the generations and converges toward the BKV. As in the EA-HH approach, *Pop(g+1)* is composed of the best $\frac{N_p}{2}$ solutions from *Pop(g)* and $\frac{N_p}{2}$ new solutions. In the figure, the solutions generated in each generation are represented by black dots. Each blue and red dot represents the best solution and the best $\frac{N_p}{2}^{\text{th}}$ solution, respectively, in the corresponding generation. In

the figure, it can be observed that the search procedure maintains diversification in the solution population throughout the generations, and this helps in avoiding a situation where the search becomes trapped in local optima as the other heuristics are trapped in a local optimum. It can also be observed that the best $\frac{N_p}{2}$ solutions move toward the global best solution and this helps in exploiting the neighborhood space of the global best solution, as the low-level heuristics use the best $\frac{N_p}{2}$ and the worst $\frac{N_p}{2}$ solutions to generate a new solution. The reason for considering the worst $\frac{N_p}{2}$ solutions is the nature of the problem: the SPP problem is a subset selection problem and the number of objects in solution is very small compared to the size of the problem. Furthermore, it is also quite evident that the set of best solutions will have many common objects and will force the operator to generate a solution similar to the best solution. In resultant, it will take the search procedure toward a local optimum.

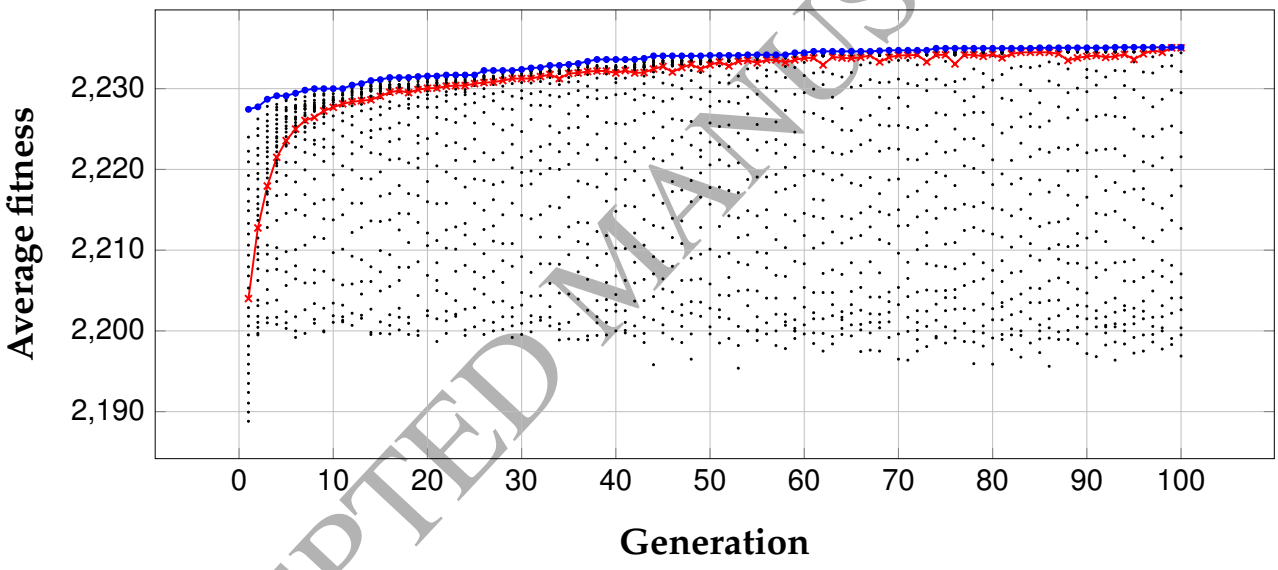


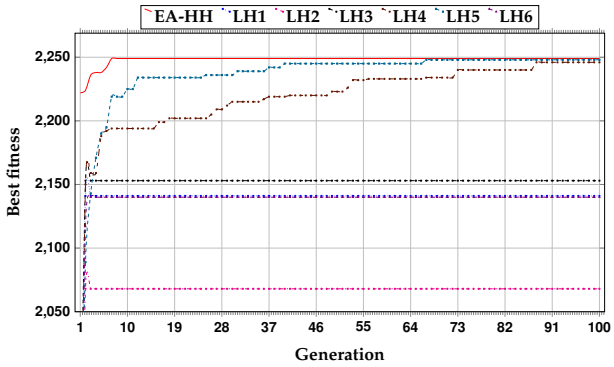
Figure 5: Population evolution of the EA-HH approach with generations on instance `pb1000rnd07`

6.5. *Comparison of the convergence speed and success rate*

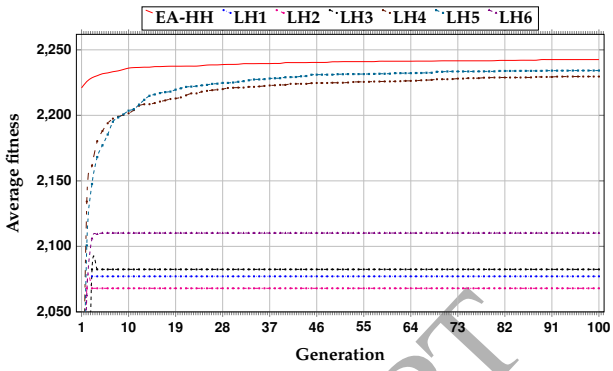
In Figure 6, we compare the convergence speed and success rate to reaching the BKV of the EA-HH approach, with heuristics LH_1, LH_2, LH_3, LH_4, LH_5, and LH_6 on instances `pb1000rnd07`, `pb500rnd15`, and `pb500rnd09`. In these figures, the convergence behavior of the best run and the average of runs are presented. In the figure, the vertical axis represents the fitness returned by each heuristic and the horizontal axis represents generations. In Figures 6(a), 6(c) and 6(e), the best solution of each generation returned by each approach are plotted, whereas in Figures 6(b), 6(d) and 6(f), the average of 16 runs are plotted. The purpose

of plotting the best run and the average of runs is to show that, even for a particular run other approaches can outperform the EA-HH approach in terms of the average performance, the EA-HH approach is always superior. From the figures, it is evident that the convergence speed of the EA-HH approach is higher and it reaches the BKV, whereas other heuristics are slower and they are unable to reach the BKV.

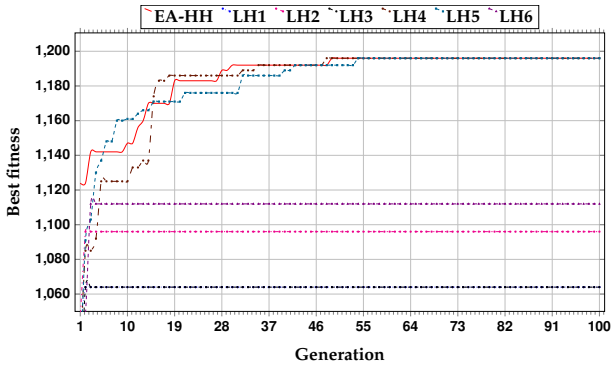
In Figures 6(a) and 6(b), it is also evident that the convergence speed of the EA-HH approach is higher than that of the other approaches. Similarly, Figures 6(c) and 6(d) and Figures 6(e) and 6(f) show that the convergence speed of the EA-HH approach is higher than that of the other approaches. In Figure 6(f), heuristics LH_1, LH_2, LH_3 are not plotted, because their convergence speed are far inferior than EE-HH, LH_4 and LH_5. If we compare the success rate in terms of reaching the BKV, EA-HH approach has higher than that of the other approaches.



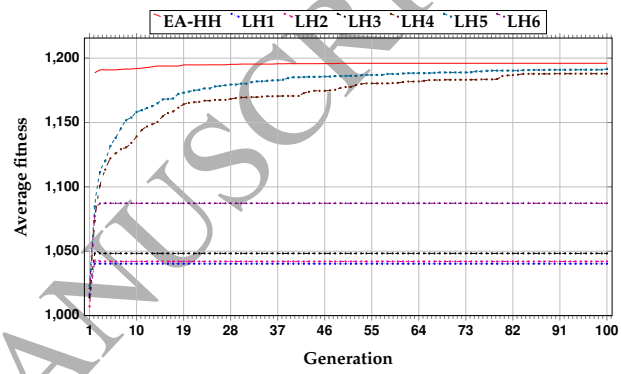
(a) Convergence speed of the best run on instance pb1000rnd07



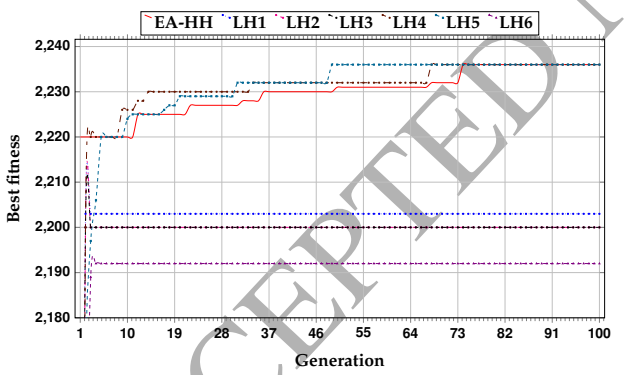
(b) Convergence speed of the average of runs on instance pb1000rnd07



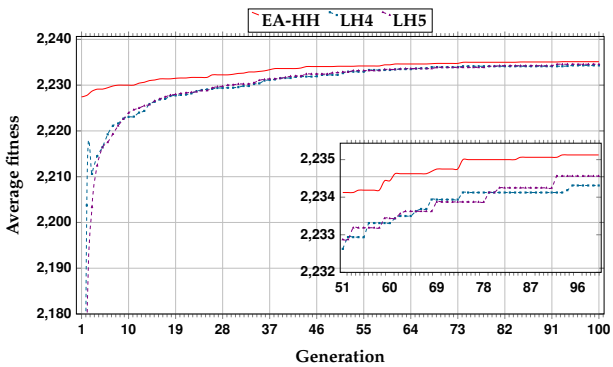
(c) Convergence speed of the best run on instance pb500rnd15



(d) Convergence speed of the average of runs on instance pb500rnd15



(e) Convergence speed of the best run on instance pb500rnd09



(f) Convergence speed of the average runs on instance pb500rnd09

Figure 6: Convergence speed of EA-HH , LH_1, LH_2, LH_3, LH_4, LH_5, and LH_6 approaches on instances pb1000rnd07, pb500rnd15, and pb500rnd09

6.6. *Performance comparison of the individual heuristic with the EA-HH approach*

This section presents the performance comparison of heuristics LH_1, LH_2, LH_3, LH_4, LH_5, and LH_6 when they are applied individually. Each individual heuristic has its own advantage and limitation: some of them have exploration ability and others have exploitation

ability. Exploration may take the search process away from the global optima, whereas exploitation may takes the search process toward the local optima. Therefore, it is necessary to achieve a balanced trade-off between exploration and exploitation in the search process to get a satisfactory solution in a reasonable amount of computational time. Tables 4 to 6 present the results of individual heuristics; the results show that each individual heuristic underperformed because of skewed nature of either exploration or exploitation, whereas the EA-HH approach utilized the exploitative and exploratory features of the individual heuristic to reach the best-known solutions.

Tables 4 to 6 present the standard deviation (SD), average solution (Avrg), and average total execution time (ATET) in seconds for heuristics LH_1, LH_2, LH_3, LH_4, LH_5, and LH_6 under the same conditions as those adopted for the EA-HH approach. The percentage (%) difference between the EA-HH approach and each individual heuristic is reported in the last six columns. The results with positive values indicate the EA-HH approach has better than that of a particular heuristic, whereas with the negative values indicate that the EA-HH has inferior performance. The results with zero values indicate that the performances of both the approaches are the same. In Tables 4 to 6, it can be observed that the EA-HH approach outperformed all the other heuristics under the same conditions. From the results, it can be concluded that the individual heuristics could not perform well owing to their limitation, whereas the EA-HH approach could determine a good solution owing to the better utilization of the individual heuristics with each generation. In other words, it can be stated that an appropriate balance in the selection of the heuristic in each generation increases the search ability. From these tables, it is clear that heuristics LH_4 and LH_5 outperformed the other heuristics in terms of the average and standard deviation. The two-tailed Wilcoxon signed-ranked test with the significance criterion 1% (p-$value \leqslant 0.01$) is used to investigate the significance level with the EA-HH approach. An online calculator[1] was used to perform this test. Table 3, presents the results of the Wilcoxon-signed-ranked test and from the outcomes, it is evident that the EA-HH approach is significant than the heuristics when they are used individually. The mean rank of heuristics LH_1, LH_2, LH_3, LH_4, LH_5 and LH_6 are 2.70, 2.55, 2.34, 5.69, 5.53 and 3.56, respectively. In the mean rank values, smaller value indicates smaller rank and higher value indicates higher rank. From the mean rank value, it is clear that LH_4 has the best performance

[1] https://mathcracker.com/wilcoxon-signed-ranks.php

among the heuristics. If we compare the robustness of the heuristics, the standard deviation shows that these heuristics have no robustness. In terms of computational time, the mean rank test of heuristics LH_1, LH_2, LH_3, LH_4, LH_5 and LH_6 are 4.27, 4.20, 3.97, 1.87, 4.02, 3.87, respectively. Whereas, in terms of standard variation, the mean rank test of the heuristics LH_1, LH_2, LH_3, LH_4, LH_5 and LH_6 are 4.25, 4.14, 4.03, 2.22, 3.42, 4.45, respectively. In the case of computation time and standard deviation, small mean rank value and large mean rank value indicate the higher rank and the lower rank, respectively.

Table 3: Wilcoxon-signed-rank test for the EA-HH with the other heuristics for the SPP

Approach	EA-HH vs.						
	N	W^+	W^-	Z	Z_c	*p-value*	*Significance*
LH_1	64	0.00	1953.00	-6.87	-2.33	0.00	*yes*
LH_2	64	0.00	1953.00	-6.87	-2.33	0.00	*yes*
LH_3	64	0.00	1953.00	-6.87	-2.33	0.00	*yes*
LH_4	64	43.50	659.50	-4.65	-2.33	0.00	*yes*
LH_5	64	25.00	1151.00	-5.77	-2.33	0.00	*yes*
LH_6	64	0.00	1953.00	-6.87	-2.33	0.00	*yes*

Table 4: Comparison on instances with up to 200 variables

Instance	LH_1	LH_2	LH_3	LH_4	LH_5	LH_6	% improvement by the EA-HH approach over					
							LH_1	LH_2	LH_3	LH_4	LH_5	LH_6
pb100rnd01	$^{(5.50)}357.88_{(0.21)}$	$^{(8.99)}358.81_{(0.21)}$	$^{(5.59)}361.31_{(0.21)}$	$^{(0.00)}372.00_{(0.12)}$	$^{(0.00)}372.00_{(0.17)}$	$^{(4.92)}367.38_{(0.11)}$	3.80	3.55	2.87	0.00	0.00	1.24
pb100rnd02	$^{(0.83)}33.06_{(0.23)}$	$^{(0.85)}32.69_{(0.23)}$	$^{(0.71)}33.00_{(0.22)}$	$^{(0.00)}34.00_{(0.12)}$	$^{(0.00)}34.00_{(0.20)}$	$^{(0.24)}33.94_{(0.18)}$	2.76	3.85	2.94	0.00	0.00	0.18
pb100rnd03	$^{(2.56)}193.25_{(0.16)}$	$^{(1.83)}194.12_{(0.21)}$	$^{(1.83)}194.88_{(0.21)}$	$^{(0.00)}203.00_{(0.11)}$	$^{(1.94)}202.50_{(0.14)}$	$^{(3.86)}198.06_{(0.14)}$	4.80	4.37	4.00	0.00	0.25	2.43
pb100rnd04	$^{(0.43)}15.25_{(0.18)}$	$^{(0.33)}15.12_{(0.18)}$	$^{(0.24)}15.06_{(0.18)}$	$^{(0.24)}15.94_{(0.09)}$	$^{(0.24)}15.94_{(0.12)}$	$^{(0.46)}15.31_{(0.16)}$	4.69	5.50	5.88	0.38	0.38	4.31
pb100rnd05	$^{(5.46)}627.19_{(0.15)}$	$^{(5.46)}627.19_{(0.15)}$	$^{(6.04)}626.88_{(0.14)}$	$^{(0.00)}639.00_{(0.09)}$	$^{(0.00)}639.00_{(0.16)}$	$^{(2.38)}638.06_{(0.10)}$	1.85	1.85	1.90	0.00	0.00	0.15
pb100rnd06	$^{(0.66)}62.94_{(0.13)}$	$^{(0.66)}62.94_{(0.12)}$	$^{(0.56)}62.75_{(0.12)}$	$^{(0.00)}64.00_{(0.07)}$	$^{(0.00)}64.00_{(0.11)}$	$^{(0.00)}64.00_{(0.09)}$	1.66	1.66	1.95	0.00	3.55	0.00
pb100rnd07	$^{(0.00)}496.00_{(0.07)}$	$^{(0.00)}496.00_{(0.07)}$	$^{(0.00)}496.00_{(0.07)}$	$^{(0.00)}499.00_{(0.09)}$	$^{(0.97)}502.75_{(0.20)}$	$^{(2.23)}500.31_{(0.16)}$	1.39	1.39	1.39	0.80	3.85	0.53
pb100rnd08	$^{(0.48)}37.62_{(0.27)}$	$^{(0.66)}38.06_{(0.27)}$	$^{(0.66)}37.75_{(0.27)}$	$^{(0.00)}39.00_{(0.10)}$	$^{(0.00)}39.00_{(0.20)}$	$^{(0.50)}38.44_{(0.15)}$	3.54	2.41	3.21	0.00	0.00	1.44
pb100rnd09	$^{(0.00)}455.00_{(0.35)}$	$^{(0.00)}455.00_{(0.35)}$	$^{(6.05)}453.44_{(0.33)}$	$^{(0.00)}463.00_{(0.09)}$	$^{(2.17)}461.75_{(0.18)}$	$^{(4.32)}455.94_{(0.14)}$	1.73	1.73	2.06	0.00	0.27	1.52
pb100rnd10	$^{(0.43)}38.75_{(0.24)}$	$^{(0.46)}38.69_{(0.24)}$	$^{(0.33)}38.88_{(0.24)}$	$^{(0.00)}40.00_{(0.10)}$	$^{(0.00)}40.00_{(0.20)}$	$^{(0.39)}39.19_{(0.16)}$	3.13	3.28	2.80	0.00	0.00	2.03
pb100rnd11	$^{(0.00)}306.00_{(0.05)}$	$^{(0.00)}306.00_{(0.05)}$	$^{(0.00)}306.00_{(0.05)}$	$^{(0.00)}306.00_{(0.05)}$	$^{(0.00)}306.00_{(0.07)}$	$^{(2.90)}303.25_{(0.14)}$	0.00	0.00	0.00	0.00	0.00	0.9
pb100rnd12	$^{(0.50)}22.44_{(0.23)}$	$^{(0.63)}22.19_{(0.21)}$	$^{(0.50)}22.50_{(0.24)}$	$^{(0.00)}23.00_{(0.09)}$	$^{(0.00)}23.00_{(0.11)}$	$^{(0.24)}22.94_{(0.16)}$	2.43	3.52	2.17	0.00	0.00	0.26
pb200rnd01	$^{(4.44)}379.06_{(0.59)}$	$^{(6.28)}380.56_{(0.68)}$	$^{(4.15)}378.75_{(0.58)}$	$^{(0.00)}416.00_{(0.51)}$	$^{(0.56)}414.94_{(0.70)}$	$^{(9.08)}400.00_{(0.64)}$	8.88	8.52	8.95	0.00	0.25	3.85
pb200rnd02	$^{(0.93)}29.38_{(0.85)}$	$^{(0.68)}29.31_{(0.86)}$	$^{(0.50)}29.44_{(0.86)}$	$^{(0.00)}32.00_{(0.48)}$	$^{(0.00)}32.00_{(0.91)}$	$^{(0.66)}30.25_{(0.72)}$	8.19	8.41	8.00	0.00	0.00	5.47
pb200rnd03	$^{(1.65)}686.69_{(0.41)}$	$^{(0.33)}686.12_{(0.36)}$	$^{(1.92)}686.75_{(0.50)}$	$^{(2.12)}724.00_{(0.51)}$	$^{(5.43)}720.31_{(1.21)}$	$^{(9.61)}703.06_{(0.66)}$	4.52	4.60	4.51	**-0.67**	**-0.16**	2.24
pb200rnd04	$^{(0.66)}59.75_{(0.94)}$	$^{(1.00)}59.50_{(0.94)}$	$^{(0.77)}59.69_{(0.94)}$	$^{(0.00)}63.00_{(0.45)}$	$^{(0.24)}63.06_{(0.97)}$	$^{(0.48)}61.38_{(0.70)}$	5.25	5.65	5.34	0.10	0.10	2.66
pb200rnd05	$^{(4.87)}167.12_{(0.20)}$	$^{(1.94)}165.50_{(0.12)}$	$^{(3.12)}166.50_{(0.20)}$	$^{(0.00)}173.00_{(0.20)}$	$^{(3.64)}174.38_{(0.29)}$	$^{(3.18)}165.00_{(0.43)}$	9.17	10.05	9.51	5.98	5.23	10.33
pb200rnd06	$^{(0.56)}12.25_{(0.58)}$	$^{(0.43)}12.25_{(0.58)}$	$^{(0.48)}12.12_{(0.58)}$	$^{(0.24)}13.06_{(0.27)}$	$^{(0.46)}13.69_{(0.37)}$	$^{(0.71)}12.50_{(0.59)}$	12.5	12.50	13.43	6.71	2.21	10.71
pb200rnd07	$^{(5.46)}960.38_{(0.95)}$	$^{(6.62)}961.81_{(1.04)}$	$^{(6.43)}960.31_{(0.87)}$	$^{(1.06)}1003.44_{(0.34)}$	$^{(2.36)}1002.31_{(0.98)}$	$^{(5.42)}983.44_{(0.63)}$	4.34	4.20	4.35	0.06	0.17	2.05
pb200rnd08	$^{(0.75)}79.06_{(0.95)}$	$^{(1.01)}79.19_{(0.95)}$	$^{(1.07)}79.19_{(0.95)}$	$^{(0.00)}83.00_{(0.31)}$	$^{(0.43)}82.75_{(0.77)}$	$^{(0.53)}81.19_{(0.58)}$	4.75	4.59	4.59	0.00	0.30	2.18
pb200rnd09	$^{(10.41)}1316.31_{(1.37)}$	$^{(10.41)}1316.31_{(1.37)}$	$^{(10.41)}1316.31_{(1.37)}$	$^{(0.00)}1324.00_{(0.31)}$	$^{(0.78)}1323.62_{(0.80)}$	$^{(5.46)}1315.12_{(0.42)}$	0.58	0.58	0.58	0.00	0.03	0.67
pb200rnd10	$^{(0.93)}117.12_{(0.54)}$	$^{(0.78)}116.88_{(0.55)}$	$^{(0.95)}116.81_{(0.52)}$	$^{(0.00)}118.00_{(0.24)}$	$^{(0.00)}118.00_{(0.53)}$	$^{(0.48)}117.62_{(0.37)}$	0.75	0.95	1.01	0.00	0.00	0.32
pb200rnd11	$^{(0.00)}536.00_{(0.18)}$	$^{(0.00)}536.00_{(0.18)}$	$^{(0.00)}536.00_{(0.18)}$	$^{(0.00)}545.00_{(0.39)}$	$^{(1.09)}543.94_{(0.75)}$	$^{(8.70)}526.31_{(0.66)}$	1.65	1.65	1.65	0.00	0.19	3.43
pb200rnd12	$^{(0.24)}42.06_{(0.24)}$	$^{(0.00)}42.00_{(0.20)}$	$^{(0.00)}42.00_{(0.21)}$	$^{(0.00)}43.00_{(0.33)}$	$^{(0.00)}43.00_{(0.53)}$	$^{(0.83)}41.94_{(0.63)}$	2.19	2.33	2.33	0.00	0.00	2.47
pb200rnd13	$^{(9.18)}523.69_{(0.75)}$	$^{(8.19)}522.06_{(0.67)}$	$^{(8.13)}522.81_{(0.71)}$	$^{(0.00)}571.00_{(0.54)}$	$^{(2.50)}569.00_{(0.88)}$	$^{(7.31)}553.00_{(0.63)}$	8.29	8.57	8.44	0.00	0.35	3.15
pb200rnd14	$^{(0.43)}42.25_{(1.11)}$	$^{(0.43)}42.25_{(1.11)}$	$^{(0.33)}42.12_{(1.11)}$	$^{(0.00)}45.00_{(0.47)}$	$^{(0.00)}45.00_{(0.80)}$	$^{(0.90)}43.06_{(0.69)}$	6.11	6.11	6.40	0.00	0.00	4.31
pb200rnd15	$^{(6.11)}892.38_{(0.68)}$	$^{(5.95)}893.75_{(0.79)}$	$^{(6.85)}892.06_{(0.63)}$	$^{(0.00)}926.00_{(0.46)}$	$^{(0.00)}926.00_{(0.89)}$	$^{(12.67)}902.69_{(0.63)}$	3.63	3.48	3.67	0.00	0.00	2.52
pb200rnd16	$^{(0.39)}75.19_{(1.23)}$	$^{(0.56)}75.25_{(1.23)}$	$^{(0.61)}75.44_{(1.22)}$	$^{(0.24)}78.06_{(0.42)}$	$^{(0.46)}78.31_{(0.98)}$	$^{(0.90)}76.75_{(0.66)}$	4.36	4.29	4.04	0.71	0.39	2.38
pb200rnd17	$^{(6.41)}231.25_{(0.58)}$	$^{(4.46)}234.62_{(0.75)}$	$^{(6.70)}230.56_{(0.53)}$	$^{(0.00)}255.00_{(0.33)}$	$^{(4.92)}252.69_{(0.54)}$	$^{(3.54)}236.25_{(0.59)}$	9.31	7.99	9.58	0.00	0.91	7.35
pb200rnd18	$^{(0.00)}18.00_{(0.15)}$	$^{(0.00)}18.00_{(0.15)}$	$^{(0.00)}18.00_{(0.15)}$	$^{(0.24)}18.94_{(0.37)}$	$^{(0.48)}18.38_{(0.45)}$	$^{(0.39)}17.81_{(0.67)}$	5.26	5.26	5.26	0.32	3.26	6.26

$^{(SD)}Avrg_{(ATET)}$. SD, Avrg, and ATET are standard deviation, average solution, and average total execution time, respectively

Table 5: Comparison on instances with up to 500 and 1000 variables

Instance	LH_1	LH_2	LH_3	LH_4	LH_5	LH_6	% improvement by the EA-HH approach over					
							LH_1	LH_2	LH_3	LH_4	LH_5	LH_6
pb500rnd01	$^{(8.32)}297.25_{(3.14)}$	$^{(7.34)}293.31_{(2.38)}$	$^{(6.41)}292.69_{(2.64)}$	$^{(0.00)}319.00_{(2.05)}$	$^{(6.68)}315.00_{(2.68)}$	$^{(5.35)}305.31_{(3.73)}$	7.97	9.19	9.38	1.24	2.48	5.48
pb500rnd02	$^{(0.43)}21.25_{(4.58)}$	$^{(0.48)}21.38_{(4.58)}$	$^{(0.75)}21.25_{(4.58)}$	$^{(0.50)}23.56_{(2.30)}$	$^{(0.50)}23.56_{(3.94)}$	$^{(0.75)}21.94_{(4.08)}$	11.68	11.14	11.68	2.08	2.08	8.81
pb500rnd03	$^{(0.00)}723.00_{(0.89)}$	$^{(0.97)}723.25_{(1.14)}$	$^{(0.00)}723.00_{(0.89)}$	$^{(1.70)}773.50_{(3.86)}$	$^{(7.24)}768.44_{(5.84)}$	$^{(16.42)}721.31_{(4.37)}$	6.59	6.56	6.59	0.06	0.72	6.81
pb500rnd04	$^{(0.85)}54.31_{(4.73)}$	$^{(0.87)}54.00_{(4.73)}$	$^{(1.27)}54.00_{(4.73)}$	$^{(0.86)}60.38_{(3.76)}$	$^{(0.50)}60.50_{(6.32)}$	$^{(1.17)}56.44_{(4.61)}$	10.6	11.11	11.11	0.61	0.41	7.09
pb500rnd05	$^{(0.24)}110.06_{(2.37)}$	$^{(2.02)}110.69_{(2.36)}$	$^{(2.48)}110.81_{(2.37)}$	$^{(0.00)}120.00_{(0.45)}$	$^{(3.68)}115.81_{(0.76)}$	$^{(1.51)}109.19_{(2.33)}$	9.33	8.81	8.71	1.14	4.59	10.04
pb500rnd06	$^{(0.33)}7.12_{(0.70)}$	$^{(0.33)}7.12_{(0.69)}$	$^{(0.43)}7.25_{(1.01)}$	$^{(0.24)}7.94_{(0.93)}$	$^{(0.50)}7.56_{(1.04)}$	$^{(0.43)}7.25_{(2.64)}$	11.00	11.00	9.38	0.75	5.50	9.38
pb500rnd07	$^{(0.00)}1055.00_{(0.92)}$	$^{(0.00)}1055.00_{(0.93)}$	$^{(4.60)}1056.19_{(1.45)}$	$^{(1.26)}1138.69_{(3.43)}$	$^{(2.00)}1138.50_{(6.65)}$	$^{(15.90)}1087.75_{(4.01)}$	7.42	7.42	7.32	0.08	0.09	4.55
pb500rnd08	$^{(0.00)}82.00_{(0.92)}$	$^{(0.24)}82.06_{(1.15)}$	$^{(0.00)}82.00_{(0.91)}$	$^{(0.46)}88.31_{(3.19)}$	$^{(0.61)}88.56_{(5.82)}$	$^{(1.32)}83.50_{(4.12)}$	7.87	7.80	7.87	0.78	0.49	6.18
pb500rnd09	$^{(14.69)}2188.94_{(9.53)}$	$^{(13.91)}2190.62_{(10.33)}$	$^{(12.99)}2177.50_{(4.66)}$	$^{(1.80)}2234.50_{(3.03)}$	$^{(1.54)}2234.56_{(8.94)}$	$^{(17.24)}2162.81_{(4.06)}$	2.07	1.99	2.58	0.03	0.03	3.24
pb500rnd10	$^{(2.36)}172.31_{(2.36)}$	$^{(1.68)}171.75_{(2.14)}$	$^{(2.34)}172.12_{(2.14)}$	$^{(0.48)}178.38_{(2.60)}$	$^{(0.39)}178.81_{(7.42)}$	$^{(1.40)}172.31_{(3.88)}$	3.74	4.05	3.84	0.35	0.11	3.74
pb500rnd11	$^{(8.89)}380.12_{(6.33)}$	$^{(7.07)}381.00_{(6.78)}$	$^{(12.13)}379.12_{(5.43)}$	$^{(0.66)}418.75_{(2.61)}$	$^{(2.74)}417.12_{(4.31)}$	$^{(8.43)}395.06_{(4.09)}$	9.82	9.61	10.05	0.65	1.04	6.27
pb500rnd12	$^{(0.77)}30.31_{(3.78)}$	$^{(0.73)}30.19_{(3.78)}$	$^{(0.56)}30.06_{(3.79)}$	$^{(0.00)}33.00_{(2.00)}$	$^{(0.00)}33.00_{(3.54)}$	$^{(0.77)}31.31_{(4.21)}$	8.15	8.52	8.91	0.00	0.00	5.12
pb500rnd13	$^{(8.85)}420.12_{(6.38)}$	$^{(9.09)}418.25_{(6.39)}$	$^{(6.00)}416.94_{(6.38)}$	$^{(1.73)}461.44_{(1.65)}$	$^{(5.12)}462.62_{(4.09)}$	$^{(8.18)}438.94_{(4.15)}$	11.37	11.76	12.04	2.65	2.40	7.40
pb500rnd14	$^{(0.53)}34.19_{(0.93)}$	$^{(0.33)}34.12_{(0.93)}$	$^{(0.24)}34.06_{(0.71)}$	$^{(0.00)}37.00_{(2.66)}$	$^{(0.43)}36.75_{(4.45)}$	$^{(0.75)}34.75_{(4.52)}$	7.59	7.78	7.95	0.00	0.68	6.08
pb500rnd15	$^{(22.37)}1040.31_{(5.75)}$	$^{(25.41)}1042.06_{(5.74)}$	$^{(19.27)}1048.31_{(7.32)}$	$^{(0.00)}1196.00_{(4.09)}$	$^{(8.93)}1191.62_{(6.98)}$	$^{(13.54)}1087.25_{(4.07)}$	13.02	12.87	12.35	0.00	0.37	9.09
pb500rnd16	$^{(0.99)}80.38_{(6.65)}$	$^{(1.09)}79.94_{(6.30)}$	$^{(1.69)}79.88_{(5.94)}$	$^{(0.43)}87.75_{(4.24)}$	$^{(0.63)}87.19_{(7.06)}$	$^{(1.49)}81.88_{(4.45)}$	8.66	9.16	9.23	0.28	0.92	6.95
pb500rnd17	$^{(0.97)}170.25_{(2.74)}$	$^{(0.33)}170.12_{(2.74)}$	$^{(0.00)}170.00_{(2.73)}$	$^{(2.49)}177.06_{(0.79)}$	$^{(7.27)}178.12_{(1.72)}$	$^{(4.09)}169.12_{(3.04)}$	10.28	10.35	10.41	6.69	6.13	10.87
pb500rnd18	$^{(0.46)}11.31_{(1.33)}$	$^{(0.39)}11.81_{(2.87)}$	$^{(0.50)}11.56_{(2.11)}$	$^{(0.00)}13.00_{(1.39)}$	$^{(0.00)}13.00_{(2.34)}$	$^{(0.43)}11.94_{(3.53)}$	13.00	9.15	11.08	0.00	0.00	8.15
pb1000rnd100	$^{(4.68)}51.50_{(2.78)}$	$^{(5.58)}53.44_{(4.52)}$	$^{(4.49)}51.19_{(3.07)}$	$^{(0.00)}59.00_{(0.37)}$	$^{(5.39)}53.06_{(0.61)}$	$^{(4.58)}51.94_{(4.71)}$	23.13	20.24	23.6	11.94	20.81	22.48
pb1000rnd200	$^{(0.24)}3.06_{(1.19)}$	$^{(0.24)}3.06_{(1.19)}$	$^{(0.00)}3.00_{(0.92)}$	$^{(0.00)}3.00_{(0.22)}$	$^{(0.00)}3.00_{(0.25)}$	$^{(0.00)}3.00_{(4.19)}$	22.34	22.34	23.86	23.86	23.86	23.86
pb1000rnd300	$^{(27.11)}570.00_{(17.12)}$	$^{(27.11)}570.00_{(17.13)}$	$^{(0.00)}591.00_{(26.63)}$	$^{(1.06)}631.50_{(11.02)}$	$^{(8.34)}636.88_{(20.15)}$	$^{(10.21)}576.31_{(16.45)}$	11.96	11.96	8.72	2.46	1.63	10.99
pb1000rnd400	$^{(0.43)}41.25_{(24.25)}$	$^{(0.58)}41.31_{(24.25)}$	$^{(0.60)}41.38_{(24.24)}$	$^{(0.24)}47.94_{(12.91)}$	$^{(1.11)}46.88_{(23.18)}$	$^{(0.87)}42.50_{(17.69)}$	14.06	13.94	13.79	0.13	2.33	11.46
pb1000rnd500	$^{(3.39)}193.12_{(12.26)}$	$^{(3.36)}191.94_{(10.54)}$	$^{(3.10)}193.12_{(12.29)}$	$^{(1.00)}211.00_{(5.06)}$	$^{(3.20)}213.31_{(8.93)}$	$^{(9.27)}198.75_{(13.58)}$	13.01	13.54	13.01	4.95	3.91	10.47
pb1000rnd600	$^{(0.39)}12.81_{(10.79)}$	$^{(0.48)}12.62_{(8.56)}$	$^{(0.46)}12.69_{(9.30)}$	$^{(0.00)}15.00_{(4.75)}$	$^{(0.71)}14.00_{(8.56)}$	$^{(0.00)}13.00_{(14.77)}$	14.60	15.87	15.40	0.00	6.67	13.33
pb1000rnd700	$^{(24.14)}2077.12_{(10.97)}$	$^{(0.00)}2068.00_{(3.67)}$	$^{(30.15)}2082.44_{(14.62)}$	$^{(8.00)}2232.31_{(22.50)}$	$^{(7.39)}2234.19_{(38.58)}$	$^{(23.23)}2110.19_{(16.88)}$	7.38	7.78	7.14	0.46	0.37	5.90
pb1000rnd800	$^{(3.26)}154.19_{(17.57)}$	$^{(2.60)}152.50_{(10.35)}$	$^{(1.98)}151.75_{(6.74)}$	$^{(0.86)}171.12_{(20.14)}$	$^{(0.78)}171.38_{(32.80)}$	$^{(1.70)}161.00_{(16.81)}$	10.77	11.75	12.19	0.98	0.83	6.83

$^{(SD)}Avrg_{(ATET)}$. SD, Avrg, and ATET are standard deviation, average solution, and average total execution time, respectively

ACCEPTED MANUSCRIPT

Table 6: Comparison on instances with up to 2000 variables

Instance	LH_1	LH_2	LH_3	LH_4	LH_5	LH_6	% improvement by the EA-HH approach over					
							LH_1	LH_2	LH_3	LH_4	LH_5	LH_6
pb2000rnd0100	$^{(2.03)}33.38_{(10.64)}$	$^{(1.58)}32.62_{(8.73)}$	$^{(1.37)}32.50_{(8.10)}$	$^{(1.21)}39.69_{(1.15)}$	$^{(3.21)}37.06_{(5.36)}$	$^{(1.96)}33.31_{(11.14)}$	16.55	18.45	18.75	0.78	7.35	16.73
pb2000rnd0200	$^{(0.00)}2.00_{(4.19)}$	$^{(0.00)}2.00_{(4.19)}$	$^{(0.00)}2.00_{(4.19)}$	$^{(0.00)}2.00_{(2.16)}$	$^{(0.00)}2.00_{(2.13)}$	$^{(0.00)}2.00_{(9.84)}$	0.00	0.00	0.00	0.00	0.00	0.00
pb2000rnd0300	$^{(9.85)}392.56_{(52.62)}$	$^{(7.20)}390.81_{(47.16)}$	$^{(6.96)}391.44_{(52.63)}$	$^{(0.00)}437.00_{(31.42)}$	$^{(12.85)}450.94_{(65.11)}$	$^{(10.40)}413.06_{(69.75)}$	16.12	16.49	16.36	6.62	3.65	11.74
pb2000rnd0400	$^{(0.61)}27.44_{(87.07)}$	$^{(0.70)}27.38_{(87.09)}$	$^{(0.79)}27.50_{(87.09)}$	$^{(0.48)}30.62_{(32.76)}$	$^{(0.60)}30.38_{(85.26)}$	$^{(0.60)}28.12_{(75.77)}$	14.25	14.44	14.06	4.31	5.06	12.13
pb2000rnd0500	$^{(0.00)}131.00_{(40.02)}$	$^{(0.00)}131.00_{(40.01)}$	$^{(0.00)}131.00_{(40.02)}$	$^{(0.99)}133.62_{(4.84)}$	$^{(0.78)}131.12_{(8.46)}$	$^{(0.00)}131.00_{(41.92)}$	1.50	1.50	1.50	**-0.47**	1.41	1.50
pb2000rnd0600	$^{(0.00)}8.00_{(38.73)}$	$^{(0.24)}8.06_{(38.73)}$	$^{(0.24)}8.06_{(38.74)}$	$^{(0.33)}8.88_{(9.82)}$	$^{(0.43)}8.75_{(34.08)}$	$^{(0.33)}8.12_{(47.12)}$	11.11	10.44	10.44	1.33	2.78	9.78
pb2000rnd0700	$^{(9.21)}1577.19_{(20.11)}$	$^{(0.00)}1574.00_{(6.80)}$	$^{(0.00)}1574.00_{(6.80)}$	$^{(7.18)}1752.88_{(71.07)}$	$^{(13.54)}1765.38_{(138.98)}$	$^{(29.25)}1604.19_{(72.99)}$	11.81	11.99	11.99	1.99	1.29	10.30
pb2000rnd0800	$^{(2.42)}114.88_{(15.62)}$	$^{(2.54)}114.94_{(15.58)}$	$^{(0.48)}114.12_{(10.78)}$	$^{(0.61)}130.56_{(73.03)}$	$^{(0.95)}131.19_{(130.83)}$	$^{(1.73)}118.38_{(73.38)}$	13.05	13.00	13.62	1.18	0.70	10.40

$^{(SD)}Avrg_{(ATET)}$. SD, Avrg, and ATET are standard deviation, average solution and average total execution time, respectively

ACCEPTED MANUSCRIPT

6.7. *Comparison analysis of computational results of the GRASP, ACO, EA/G and EA-HH*

This section presents a comparative analysis of the combinational results of the exact approach, GRASP [14], ACO [17], EA/G [11], and the EA-HH approach that are presented in Tables 7 to 9. The results in these tables show the superiority of the EA-HH approach over GRASP, ACO, and EA/G in terms of the computational results and time. Table 7 presents the computational results of the instances with 100 and 200 objects. The EA-HH approach achieved the optimal values on all 30 instances in terms of the best solution and on 28 instances in terms of the average solution. As compared with GRASP, the EA-HH approach is better on 13 instances, poorer on 2 instances, and the same on the remaining 15 instances in terms of the best solution. In terms of the average solution, the EA-HH approach is better on 10 instances, poorer on 1 instance, and the same on 19 instances. In terms of computational time, the EA-HH approach achieved the optimal as well as a better average solution in computationally less time than GRASP approach. In comparison with ACO, the EA-HH approach is better on 2 instances and the same on 28 instances in terms of the optimal solution. In terms of the average solution, the EA-HH approach is better on 17 instances, poorer on 1 instance, and the same on 12 instances. In terms of computational time, the EA-HH approach is always faster than ACO.

EA/G is the state-of-the-art approach for the SPP and the EA-HH approach outperformed in terms of both the average solution and the computational time. Out of 30 instances, the EA-HH approach is better on 10 instances, poorer on 1 instance, and the same on 19 instances in terms of the average solution. Here, notably, the EA-HH approach achieved the optimal solution in all the runs on all the 30 instances, whereas the EA/G could achieve the optimal solution on 19 instances only. Therefore, the computational results show that the EA-HH approach outperformed the EA/G approach.

Table 7: Comparison of the EA-HH approach with GRASP [14], ACO [17], and EA/G [11] on instances with up to 200 variables

Instance	Characteristics					0/1 Solution		GRASP			ACO			EA/G			EA-HH			
	Var	Cnst	Density	Max One	Weight	Opt	TET	Best	Avrg	ATET	Best	Avrg	ATET	Best	Avrg	ATET	Best$_{(a)}$	Avrg$_{(b)}$	SD	ATET
pb100rnd01	100	500	2.0%	2	[1-20]	372	2.92	372	372.00	1.97	372	372.00	3.33	372	372.00	0.38	372$_{(28)}$	372.00$_{(28.00)}$	0.00	**0.16**
pb100rnd02	100	500	2.0%	2	[1-1]	34	0.60	34	34.00	1.31	34	34.00	2.00	34	34.00	0.22	34$_{(34)}$	34.00$_{(34.00)}$	0.00	**0.16**
pb100rnd03	100	500	3.0%	4	[1-20]	203	7.81	203	203.00	1.14	203	203.00	2.00	203	203.00	0.37	203$_{(14)}$	203.00$_{(14.00)}$	0.00	**0.14**
pb100rnd04	100	500	3.0%	4	[1-1]	16	52.86	16	16.00	1.29	16	15.56	0.67	16	15.69	0.18	16$_{(16)}$	**16.00**$_{(16.00)}$	0.00	**0.12**
pb100rnd05	100	100	2.0%	2	[1-20]	639	0.01	639	639.00	0.80	639	639.00	1.67	639	639.00	0.35	639$_{(57)}$	639.00$_{(57.00)}$	0.00	**0.14**
pb100rnd06	100	100	2.0%	2	[1-1]	64	0.01	64	64.00	0.69	64	64.00	1.00	64	64.00	0.14	64$_{(64)}$	64.00$_{(64.00)}$	0.00	**0.10**
pb100rnd07	100	100	2.9%	4	[1-20]	503	0.00	503	503.00	1.00	503	503.00	1.00	503	503.00	0.38	503$_{(39)}$	503.00$_{(39.00)}$	0.00	**0.15**
pb100rnd08	100	100	3.1%	4	[1-1]	39	0.02	39	38.75	0.57	39	38.68	0.67	39	38.81	0.21	39$_{(39)}$	**39.00**$_{(39.00)}$	0.00	**0.17**
pb100rnd09	100	300	2.0%	2	[1-20]	463	0.49	463	463.00	1.26	463	463.00	1.67	463	463.00	0.38	463$_{(35)}$	463.00$_{(35.00)}$	0.00	**0.18**
pb100rnd10	100	300	2.0%	2	[1-1]	40	1.13	40	40.00	1.28	40	39.62	1.00	40	39.88	0.24	40$_{(40)}$	**40.00**$_{(40.00)}$	0.00	**0.15**
pb100rnd11	100	300	3.1%	4	[1-20]	306	0.48	306	306.00	0.68	306	306.00	1.67	306	306.00	0.40	306$_{(20)}$	306.00$_{(19.50)}$	0.00	**0.06**
pb100rnd12	100	300	3.0%	4	[1-1]	23	6.80	23	23.00	1.13	23	22.93	0.33	23	23.00	0.21	23$_{(23)}$	**23.00**$_{(23.00)}$	0.00	**0.13**
pb200rnd01	200	1000	1.5%	4	[1-20]	416	8760.73	416	415.18	7.32	416	415.25	27.33	416	416.00	1.66	416$_{(30)}$	**416.00**$_{(30.00)}$	0.00	**0.59**
pb200rnd02	200	1000	1.5%	4	[1-1]	32	156109.36	32	32.00	7.35	32	31.56	14.67	32	32.00	0.76	32$_{(32)}$	**32.00**$_{(32.00)}$	0.00	**0.64**
pb200rnd03	200	1000	1.0%	2	[1-20]	731	5403.23	726	722.81	10.81	729	725.12	44.33	731	**727.00**	1.85	**731**$_{(56)}$	719.19$_{(58.38)}$	6.38	**0.82**
pb200rnd04	200	1000	1.0%	2	[1-1]	64	63970.91	63	63.00	9.12	64	62.93	24.33	63	62.75	0.92	63$_{(63)}$	**63.00**$_{(63.00)}$	0.00	**0.70**
pb200rnd05	200	1000	2.5%	8	[1-20]	184	1211.37	184	184.00	4.62	184	182.56	16.00	184	184.00	1.07	184$_{(11)}$	**184.00**$_{(11.00)}$	0.00	**0.47**
pb200rnd06	200	1000	2.5%	8	[1-1]	14	8068.20	14	13.37	3.48	14	12.87	4.00	14	13.50	0.55	14$_{(14)}$	**14.00**$_{(14.00)}$	0.00	**0.36**
pb200rnd07	200	200	1.5%	4	[1-20]	1004	0.02	1002	1001.12	4.20	1004	1003.50	6.33	1004	1003.94	1.61	**1004**$_{(77)}$	**1004.00**$_{(77.00)}$	0.00	**0.88**
pb200rnd08	200	200	1.5%	4	[1-1]	83	0.04	83	82.87	2.71	83	82.75	2.67	83	82.81	0.80	83$_{(83)}$	**83.00**$_{(83.00)}$	0.00	0.92
pb200rnd09	200	200	1.0%	2	[1-20]	1324	0.01	1324	1324.00	3.75	1324	1324.00	7.33	1324	1324.00	1.31	1324$_{(113)}$	1324.00$_{(113.00)}$	0.00	**1.13**
pb200rnd10	200	200	1.0%	2	[1-1]	118	0.02	118	118.00	3.64	118	118.00	4.00	118	118.00	0.76	118$_{(118)}$	118.00$_{(118.00)}$	0.00	**0.59**
pb200rnd11	200	200	2.5%	8	[1-20]	545	0.33	545	544.75	2.36	545	545.00	4.33	545	545.00	1.65	545$_{(38)}$	**545.00**$_{(38.00)}$	0.00	**0.90**
pb200rnd12	200	200	2.6%	8	[1-1]	43	1.70	43	43.00	1.01	43	43.00	1.33	43	43.00	0.80	43$_{(43)}$	43.00$_{(43.12)}$	0.00	**0.65**
pb200rnd13	200	600	1.5%	4	[1-20]	571	830.39	571	566.43	6.01	571	568.50	20.33	571	570.75	1.74	571$_{(41)}$	**571.00**$_{(41.00)}$	0.00	**1.02**
pb200rnd14	200	600	1.5%	4	[1-1]	45	10066.91	45	45.00	3.92	45	44.43	8.67	45	44.62	0.75	45$_{(45)}$	**45.00**$_{(45.00)}$	0.00	0.99
pb200rnd15	200	600	1.0%	2	[1-20]	926	12.20	926	926.00	4.22	926	926.00	27.00	926	926.00	1.72	926$_{(76)}$	926.00$_{(76.00)}$	0.00	**1.08**
pb200rnd16	200	600	1.0%	2	[1-1]	79	14372.85	79	78.31	6.80	79	78.37	15.33	79	78.12	0.98	79$_{(79)}$	**78.62**$_{(78.62)}$	0.48	1.18
pb200rnd17	200	600	2.5%	8	[1-20]	255	741.52	255	251.31	3.61	255	253.25	11.00	255	254.19	1.22	255$_{(18)}$	**255.00**$_{(18.00)}$	0.00	**0.72**
pb200rnd18	200	600	2.6%	8	[1-1]	19	19285.06	19	18.06	2.35	19	18.12	3.00	19	18.19	0.65	19$_{(19)}$	**19.00**$_{(19.00)}$	0.00	0.69

(*a*) indicates number of objects in the best solution (*b*) indicates average number of objects over 16 runs

Table 8 presents the computational results on the instances with 500 and 1000 objects. The exact approach failed to determine the optimal solution within the specified computational time on most of the instances, and therefore, the solution is the BKV and is indicated by "*". Out of 26 instances, the exact approach determined the optimal solution on 9 instances and on the remaining 17 instances, it achieved solution is the BKV. The EA-HH approach achieved the optimal value on eight out of nine instances, whereas in terms of the BKV, out of 17 instances, the EA-HH approach achieved the same results as the BKV on 13 instances, poorer than the BKV on 1 instance, and better than the BKV on 3 instances. As compared with GRASP, the EA-HH approach is better on 9 instances and the same on 17 instances in terms of the best solution. In terms of the average solution, the EA-HH approach is better on 21 instances, poorer on 1 instance, and the same on 4 instances. As compared with ACO approach, the EA-HH approach outperformed this approach. Out of 26 instances, the EA-HH approach is superior on 5 instances and inferior on 2 instances in terms of the best solution. In terms of the average solution, the EA-HH approach is superior on 25 instances and inferior on 1 instance. In terms of the computational time, the EA-HH approach achieved a better solution in computationally less time than ACO approach. The EA-HH approach also outperformed the EA/G, which outperformed GRASP and ACO approaches. In terms of the best solution, the EA-HH approach determined the same solution on all the instances, whereas in terms of the average solution, the EA-HH approach is better on 21 instances, poorer on 4 instances, and the same on 1 instance. At the beginning of the computational section, it was mentioned that the EA-HH approach was executed on the same system as that used for the EA/G approach. Therefore, from Table 8, it can be observed that wherever the EA-HH approach has the same solution fitness, it took less computational time and on some instances, it achieved a better solution in computationally less time than the EA/G approach. Out of 26 instances, on 15 instances, the EA-HH approach returned a better solution than the EA/G approach. On 11 instances, the EA-HH approach achieved a better solution fitness at the cost of extra computational time. As already mentioned, both the EA-HH and EA/G approaches generated the same number of solutions and also the same number of fitness evolutions. However, EA/G approach could not achieve a better solution than the EA-HH approach. The reason is that EA/G approach converged to the local optima and could not explore the other areas of the solution space, whereas the EA-HH approach could explore many areas of the solution space. It is evident in Section 6.4 that the

solution population always maintains diversity.

Table 8: Comparison of the EA-HH approach with GRASP [14], ACO [17], and EA/G [11] on instances with 500 and 1000 variables

Instance	**Characteristics**					**0/1 Solution**	**GRASP**			**ACO**			**EA/G**			**EA-HH**			
	Var	Cnst	Density	Max One	Weight	BKV	Best	Avrg	ATET	Best	Avrg	ATET	Best	Avrg	ATET	Best$_{(a)}$	Avrg$_{(b)}$	SD	ATET
pb500rnd01	500	2500	1.2%	10	[1-20]	323*	323	319.38	32.08	323	319.87	154.67	323	321.31	7.03	323$_{(23)}$	**323.00**$_{(23.00)}$	0.00	**4.54**
pb500rnd02	500	2500	1.2%	10	[1-1]	24*	24	23.69	25.62	24	23.06	26.67	25	23.56	5.16	25$_{(25)}$	**24.06**$_{(24.06)}$	0.24	**4.65**
pb500rnd03	500	2500	0.7%	5	[1-20]	776*	772	767.63	70.33	776	772.75	244.00	776	771.38	10.61	774$_{(58)}$	**774.00**$_{(58.00)}$	0.00	**6.21**
pb500rnd04	500	2500	0.7%	5	[1-1]	61*	61	60.13	57.30	61	60.06	69.00	62	60.38	6.06	62$_{(62)}$	**60.75**$_{(60.75)}$	0.56	6.48
pb500rnd05	500	2500	2.2%	20	[1-20]	122	122	121.50	15.48	122	120.62	71.00	122	**121.56**	3.92	122$_{(7)}$	**121.38**$_{(6.94)}$	2.42	**2.13**
pb500rnd06	500	2500	2.2%	20	[1-1]	8*	8	8.00	12.08	8	7.87	9.67	8	7.88	2.27	8$_{(8)}$	**8.00**$_{(8.00)}$	0.00	2.41
pb500rnd07	500	500	1.2%	10	[1-20]	1141	1141	**1141.00**	13.43	1141	**1141.00**	60.00	1141	1139.94	9.64	1141$_{(80)}$	1139.56$_{(80.44)}$	1.41	**7.57**
pb500rnd08	500	500	1.2%	10	[1-1]	89	89	88.25	15.80	89	88.12	21.67	89	88.94	4.75	89$_{(89)}$	**89.00**$_{(89.00)}$	0.00	6.25
pb500rnd09	500	500	0.7%	5	[1-20]	2236	2235	2235.00	23.44	2236	2234.43	84.33	2236	2232.81	12.49	**2236**$_{(165)}$	**2235.12**$_{(165.31)}$	1.83	**11.58**
pb500rnd10	500	500	0.7%	5	[1-1]	179	179	178.06	18.20	179	178.31	44.67	179	178.56	6.12	179$_{(179)}$	**179.00**$_{(179.00)}$	0.00	7.03
pb500rnd11	500	500	2.3%	20	[1-20]	424	423	419.31	19.25	424	418.25	37.33	424	**421.56**	7.92	424$_{(30)}$	**421.50**$_{(30.00)}$	1.94	**6.79**
pb500rnd12	500	500	2.2%	20	[1-1]	33*	33	33.00	11.91	33	32.62	8.00	33	32.88	4.00	33$_{(33)}$	**33.00**$_{(33.00)}$	0.00	**3.92**
pb500rnd13	500	1500	1.2%	10	[1-20]	474*	474	470.00	32.88	474	468.00	105.00	474	468.56	8.90	474$_{(31)}$	**474.00**$_{(31.00)}$	0.00	**5.97**
pb500rnd14	500	1500	1.2%	10	[1-1]	37*	37	36.94	20.77	37	36.50	25.66	38	36.88	4.39	38$_{(38)}$	**37.00**$_{(37.00)}$	0.50	5.20
pb500rnd15	500	1500	0.7%	5	[1-20]	1196*	1196	1186.94	59.36	1196	1190.93	161.67	1196	1185.62	10.68	1196$_{(79)}$	**1196.00**$_{(79.00)}$	0.00	**8.32**
pb500rnd16	500	1500	0.7%	5	[1-1]	88*	88	86.63	36.31	88	86.12	60.67	88	87.00	5.12	88$_{(88)}$	**88.00**$_{(88.00)}$	0.00	7.54
pb500rnd17	500	1500	2.2%	20	[1-20]	192*	192	**191.75**	18.38	192	188.31	57.00	192	191.38	5.55	192$_{(11)}$	**189.75**$_{(11.00)}$	1.98	**3.06**
pb500rnd18	500	1500	2.2%	20	[1-1]	13*	13	13.00	12.03	13	12.43	9.00	13	12.81	2.85	13$_{(13)}$	**13.00**$_{(13.00)}$	0.00	2.96
pb1000rnd100	1000	5000	2.60%	50	[1-20]	67	67	65.50	53.50	67	64.00	117.67	67	67.00	10.64	67$_{(4)}$	**67.00**$_{(4.00)}$	0.00	**4.48**
pb1000rnd200	1000	5000	2.59%	50	[1-1]	4	4	3.15	39.30	4	3.81	15.00	4	3.44	3.58	4$_{(4)}$	**3.94**$_{(3.94)}$	0.24	**2.44**
pb1000rnd300	1000	5000	0.60%	10	[1-20]	661*	649	639.50	221.20	661	640.50	700.67	661	642.81	32.92	648$_{(43)}$	**647.44**$_{(42.88)}$	2.18	**16.01**
pb1000rnd400	1000	5000	0.60%	10	[1-1]	48*	48	46.83	149.70	47	45.06	108.33	48	46.50	18.12	48$_{(48)}$	**48.00**$_{(48.00)}$	0.00	25.82
pb1000rnd500	1000	1000	2.60%	50	[1-20]	222*	222	217.98	64.80	222	219.62	85.67	222	217.88	23.11	222$_{(14)}$	**222.00**$_{(14.00)}$	0.00	**12.59**
pb1000rnd600	1000	1000	2.65%	50	[1-1]	15*	14	13.68	41.40	15	13.37	8.00	15	14.06	12.03	15$_{(15)}$	**15.00**$_{(15.00)}$	0.00	13.04
pb1000rnd700	1000	1000	0.58%	10	[1-20]	2260	2222	2214.10	119.70	2248	2239.56	296.67	2253	2242.06	44.48	2249$_{(163)}$	**2242.56**$_{(163.62)}$	6.43	49.83
pb1000rnd800	1000	1000	0.60%	10	[1-1]	175*	172	170.81	82.60	173	170.50	94.00	174	172.62	21.21	173$_{(173)}$	**172.81**$_{(172.81)}$	0.39	37.59

(*a*) indicates number of objects in the best solution (*b*) indicates average number of objects over 16 runs

Table 9 presents the computational results for the instances with 2000 objects. For the large instances, the exact approach and GRASP could achieve the optimal solution on only 2 out of 8 instances. The EA-HH approach found the new BKV for the instances `pb2000rnd0700` and `pb2000rnd0800`. GRASP approach reported only the best solution out of 16 runs and it could achieve BKV on 5 instances. As compared with GRASP, the EA-HH approach is better on 2 instances, poorer on 1 instance, and the same on the remaining 4 instances in terms of the best solution. As compared with EA/G, the EA-HH approach is better on 2 instances, poorer on 2 instances, and the same on the remaining 4 instances in terms of the best solution. In terms of the average solution, the EA-HH approach is better on 4 instances, poorer on 1 instance, and the same on the remaining 3 instances. Notably, the EA-HH approach achieved a better solution at the cost of extra computational time on 3 instances. This shows that the EA-HH approach has the ability to explore a different area of the solution space of the problem under consideration.

Table 9: Comparison of the EA-HH approach with GRASP [14], and EA/G [11] on instances with 2000 variables

Instance	Characteristics					**0/1 Solution**	GRASP	EA/G			EA-HH			
	Var	Cnst	Density	Max One	Weight	BKV	Best	Best	Avrg	ATET	Best$_{(a)}$	Avrg$_{(b)}$	SD	ATET
pb2000rnd0100	2000	10000	2.54%	100	[1-20]	40	*Yes*	40	40.00	31.66	40$_{(2)}$	40$_{(2.00)}$	0.00	**10.69**
pb2000rnd0200	2000	10000	2.55%	100	[1-1]	2	*Yes*	2	2.00	10.38	2$_{(2)}$	2$_{(2.00)}$	0.00	**5.08**
pb2000rnd0300	2000	10000	0.55%	20	[1-20]	478*	*No*	478	463.75	129.86	475$_{(27)}$	**468**$_{(27.25)}$	12.12	**85.55**
pb2000rnd0400	2000	10000	0.55%	20	[1-1]	32*	*Yes*	32	30.25	81.58	32$_{(32)}$	**32**$_{(32.00)}$	0.00	86.00
pb2000rnd0500	2000	2000	2.55%	100	[1-20]	140*	*Yes*	140	**135.50**	61.03	133$_{(8)}$	133$_{(8.00)}$	0.00	**22.97**
pb2000rnd0600	2000	2000	2.56%	100	[1-1]	9*	*Yes*	9	8.50	34.23	9$_{(9)}$	**9**$_{(9.00)}$	0.00	44.22
pb2000rnd0700	2000	2000	0.56%	20	[1-20]	1784*	*No*	1794	1765.94	160.14	**1796**$_{(119)}$	**1788.38**$_{(121.19)}$	5.25	**150.01**
pb2000rnd0800	2000	2000	0.56%	20	[1-1]	131*	*No*	132	130.38	73.84	**133**$_{(133)}$	**132.12**$_{(132.12)}$	1.45	131.39

(*a*) indicates number of objects in the best solution (*b*) indicates average number of objects over 16 runs

The superiority of the EA-HH approach over other approaches such as GRASP, ACO, and EA/G can be attributed to the utilization of the evolutionary algorithm in the hyper-heuristic framework which allows to take advantage of low-level heuristics to explore/exploit the solution space of the problem under consideration. However, the computational results show that on the some instances the EA-HH approach failed to achieve the same solution achieved by the other approaches and this asserts the "no free lunch theory" [48]. Another observation that can be made concerning computational time is the number of objects in the solution which is directly affected by the density of objects. Number of object is inversly proportional to the density. As a result of the higher density, the conflict size of objects becomes higher, whereas as the result of low density, the conflict size becomes smaller. And also, the number of objects in the solution

directly affects the computational time. As the number of objects increases, the computational time also increases. The reason for the increase in the computational time is that the search procedure spends more time in eliminating the conflicting objects from the infeasible solution. For example, the characteristics of instances `pb1000rnd05` and `pb1000rnd07` are {*Var* = 1000, *Cnst* = 1000, *Density* = 2.60%} and {*Var* = 1000, *Cnst* = 1000, *Density* = 0.58%}, respectively. The analysis of the computational results of these instances reveals that the number of objects is 14 and 163, respectively, and the computational time (ATET) is 12.50s and 49.83s, respectively. Therefore, it is evident that the computational time for the instances with higher density is less than that of an instance with lower density. Similarly, the computational time of the groups of instances reveals that all the groups show a similar trend of change in computational time.

6.8. The Wilcoxon signed-rank test

The two-tailed Wilcoxon signed-ranked test with the significance criterion of 1% (*p-value* $\leqslant$ 0.01). Table 10 presents the results of the Wilcoxon-signed-ranked test. The results in the table show that the EA-HH approach is significant compared to the other approaches. The Friedman mean ranks of the EA-HH, EA/G, GRASP, and ACO approaches are 3.3, 2.5, 2.3 and 1.8, respectively. A larger value indicates better performance. The mean ranks show that the EA-HH exhibits better performance than the other approaches.

Table 10: Wilcoxon-signed-rank test of the EA-HH approach with the other approaches for the SPP

Approach	EA-HH vs.						
	N	W^+	W^-	Z	C_N	Z_c	*Significance*
GRASP [14]	56	557.5	72.5	-3.972	n = 35 > 30	-2.58	*yes*
ACO [17]	56	877	69	-4.878	n = 43 > 30	-2.58	*yes*
EA/G [11]	64	852.5	137.5	-4.172	n = 44 > 30	-2.58	*yes*
LH_1	64	1953	0	-6.846	n = 62 > 30	-2.58	*yes*
LH_2	64	1953	0	-6.846	n = 62 > 30	-2.58	*yes*
LH_3	64	1953	0	-6.846	n = 62 > 30	-2.58	*yes*
LH_4	64	703	0	-5.303	n = 37 > 30	-2.58	*yes*
LH_5	64	1173.5	2.5	-6.005	n = 48 > 30	-2.58	*yes*
LH_6	64	1953	0	-6.846	n = 62 > 30	-2.58	*yes*

6.9. Box plot analysis

This section examines the dispersion of fitness values from the median for the EA-HH, LH_1, LH_2, LH_3, LH_4, LH_5 and LH_6 approaches. The purpose of this analysis is to ob-

serve the capabilities of the approaches in terms of searching for a good solution in the solution space. The boxplots are used to show the robustness of the proposed approach. In Figures 7(a) to 7(d) and Figures 8(a) to 8(d), the boxplots of the fitness values returned by the EA-HH , LH_1, LH_2, LH_3, LH_4, LH_5, and LH_6 approaches are shown. For the sake of a better representation, the fitness values are scaled between 0 and 1. In the figures, the "`Fitness`" axis represents the scaled fitness value and the "`Instance`" axis represents instances. The analysis is focused on the instances for which the standard deviation of the approaches is not zero. The reason for this is to determine the dispersion of fitness values. In Tables 4 to 6, it can be observed that the performances of heuristics LH_4 and LH_5 are better than those of LH_1, LH_2, and LH_3. In the boxplots in Figures 7(a) to 7(d) and Figures 8(a) to 8(d), it can be observed that, on most of the instances, all the heuristics, except EA-HH, have more dispersion from the median. This shows that the other heuristics determined the best solution among 16 runs randomly and they are not robust.

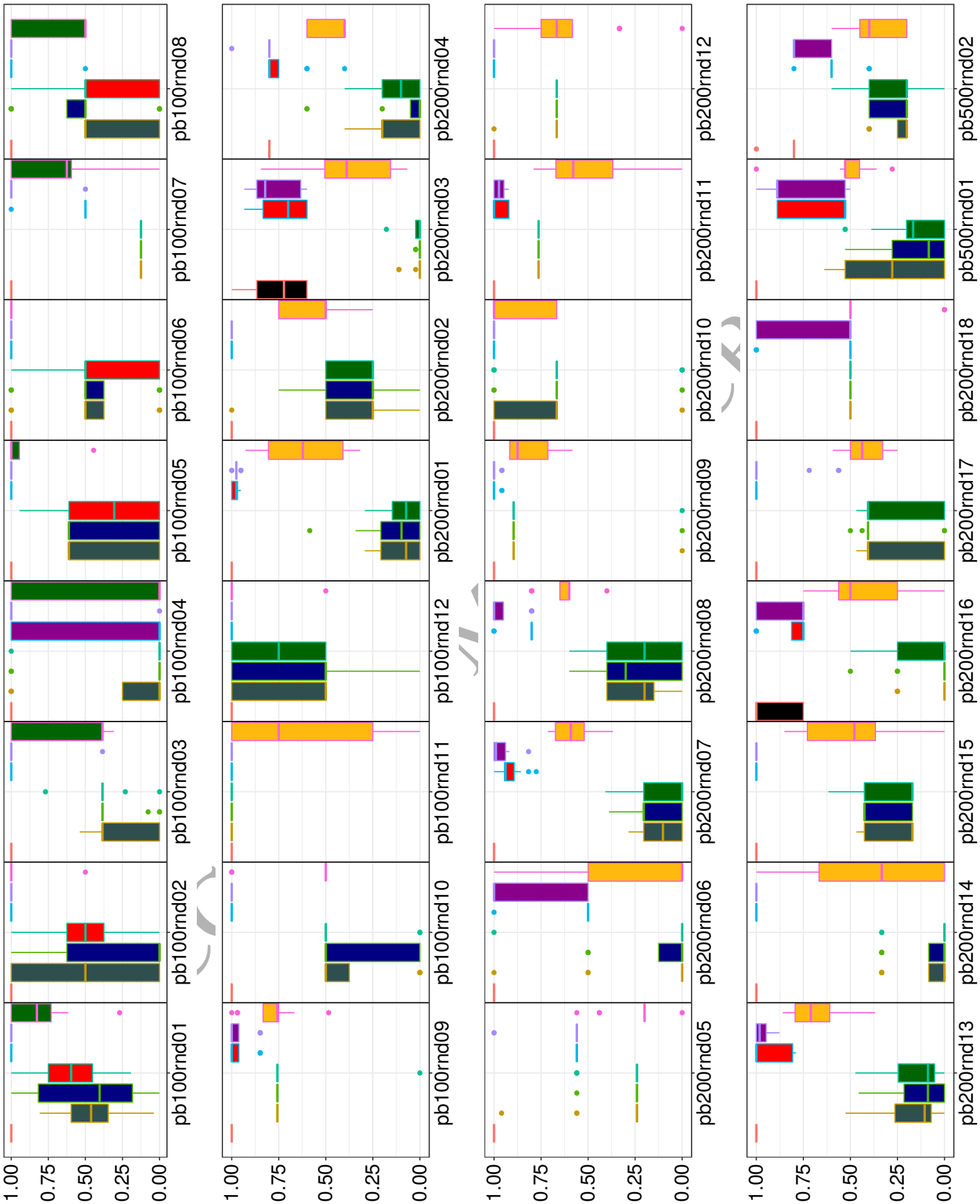


Figure 7: Boxplots of the approaches EA-HH , LH_1, LH_2, LH_3, LH_4, LH_5, and LH_6

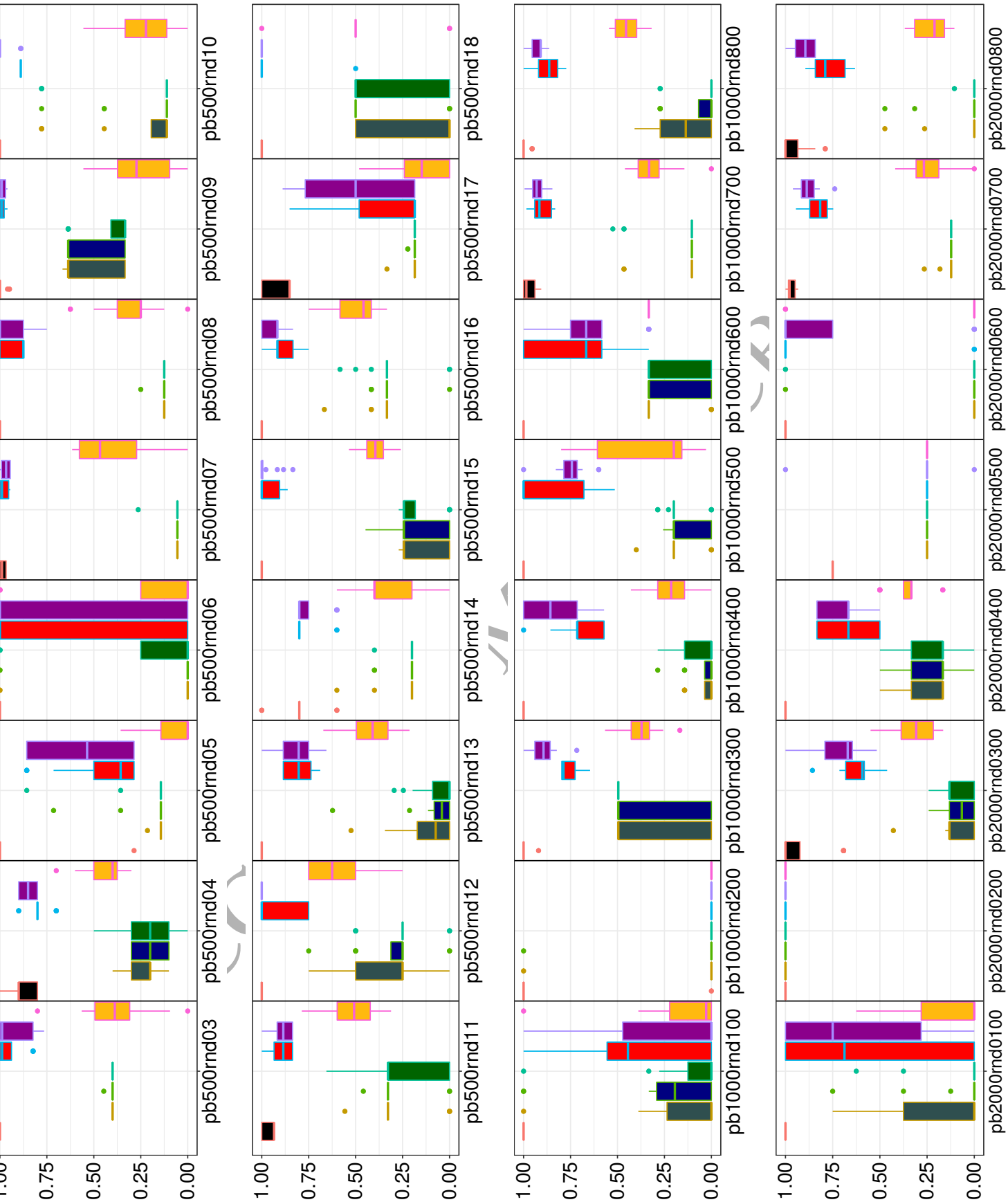


Figure 8: Boxplots of the approaches EA-HH , LH_1, LH_2, LH_3, LH_4, LH_5, and LH_6

7. Minimum weight dominating set problem

In this section, we consider the minimum weight dominating set (MWDS) problem which is also an $\mathcal{NP}$-*hard* [18]. The purpose of considering the MWDS problem in this paper is to show the scalability and generality of the proposed EA-HH approach. Both the SPP and MWDS problems are subset selection problem but have opposite objectives. The SPP has the maximization objective whereas the MWDS problem has the minimization objective. As mentioned in Section 4.2, the low-level heuristics are standard operators and are independent of the nature of the problem. Most of the low-level heuristics do not use the problem domain knowledge in the process of generating a new solution. Therefore, the proposed approach is applied to the MWDS problem to investigate the scalability and generality.

7.1. Problem description

Consider a node weighted undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges. Any pair, say (v, u), of nodes v and u is considered as a pair of adjacent nodes or neighbor iff, there exists an edge between them, i.e., $(v, u) \in \mathcal{E}$. A dominating set $\mathcal{D} \subseteq \mathcal{V}$ is a set of nodes such that each node in $\mathcal{V}$ either belongs to subset $\mathcal{D}$ or is adjacent to at least one node in $\mathcal{D}$, i.e., node in $\mathcal{V} \backslash \mathcal{D}$ has at least one adjacent node in $\mathcal{D}$. Any node that belongs to subset $\mathcal{D}$ is called a dominating node or dominator, and a node not in subset $\mathcal{D}$ is called a non-dominating or dominatee node. The objective of the minimum dominating set problem (MDS) is to find a dominating set with minimum cardinality from all possible dominating sets in $\mathcal{G}$.

In the MWDS problem, a non-negative weight, assigned through a weight function $w : \mathcal{V} \rightarrow \Re^+$, is associated with each node in $\mathcal{V}$, and the objective is to find a dominating set whose sum of weights of nodes is minimum among all the possible dominating sets in $\mathcal{G}$. Both the MDS and MWDS problems have been proven to be $\mathcal{NP}$-*hard* [18]. The object function of the MWDS can be defined as

$\arg\min_{\mathcal{D} \in \mathcal{D}'} \sum_{v \in \mathcal{D}'} w(v)$, where $\mathcal{D}'$ is the set of all possible dominating sets in $\mathcal{G}$.

7.2. The EA-HH approach for the MWDS problem

This section presents the EA-HH approach for the MWDS problem under similar condition as of the SPP. The main components of the EA-HH approach for the MWDS problem are as follows:

7.2.1. *Solution encoding*

As both SPP and MWDS problem are subset selection problem, a subset encoding has been used to represent a solution. The same solution encoding is followed as that used for the SPP.

7.2.2. *Initial solution representation*

As both SPP and MWDS are subset selection problem; therefore, the initial population is generated in the same manner as for the SPP.

Each infeasible solution is passed through the repair operator to make the solution feasible and subsequently, the feasible solution is passed through the improvement operator. The same repair and improvement operators are used as those used in [8].

7.2.3. *Solution generation in generation of the EA-HH approach*

In each generation, similar to the SPP, a new solution is generated by applying the hyper-heuristic. Only LH_1, LH_2, LH_3, LH_5 heuristics are used at the lower level of the hyper-heuristic. As heuristic LH_6 is the problem specific and cannot directly be applied for the MWDS problem. All the above mentioned low-level heuristics are applied in the same manner as for the SPP, except the guided mutation. In the guided mutation, the probability initialization method does not use the problem domain knowledge to initialize the probability vector. Each infeasible solution is passed through the repair to make it feasible and then through the improvement operator to improve the solution quality. The repair and improvement operators are explained in the subsequent section.

7.2.4. *Repair and improvement operators*

Similar to the SPP, after the application of low-level heuristic, each solution is passed through the repair operator to make the solution feasible, and the feasible solution is passed through the improvement operator further to improve the solution quality.

The same repair operator is used as used in [8] to make the infeasible solution feasible. Only the infeasible solution is passed through the repair operator. The repair operator attempts to add uncovered nodes one after another. Uncovered nodes are those nodes that are either not dominating nodes or not the neighbor(s) of at least one dominating node. Iteratively, an uncovered node whose ratio between the sum of weights of uncovered nodes and the weight of the node itself is the highest is chosen.

The improvement operator is used to remove the redundant nodes without violating the feasibility criteria. A redundant node, say v, is a dominating node whose neighborhood nodes

and node itself are covered by at least one dominating node other than node v in the dominating set. Iteratively, a node whose ratio between the weight of node and number of neighborhood nodes among the set of redundant nodes is highest is removed. For a detailed description of the repair and improvement operators, please, refer to Sections 5.5 and 5.6 of [8].

7.3. Computational results for the MWDS problem

This section presents the computational results for the MWDS problem. In [8], two types of instances with different weight ranges are used. For Type I instances, weights of nodes in the network are randomly distributed in the closed interval [20, 70], whereas for Type II instances, weights are randomly distributed in the closed interval [1, $dc(v)^2$], where $dc(v)^2$ is square of the degree of the node v. In this study, instances with network sizes 300, 500, 800, and 1000 and the number of edges varies from 300 to 20000 for both Type I and Type II instances are considered. Ten instances were generated for each combination of the number of nodes ($|\mathcal{V}|$) and the number of edges ($|\mathcal{E}|$). A total of 420 (210$\times$ 2 instances are considered for Type I (21 $\times$ 10) and Type II (21 $\times$ 10)).

The same system and the same parameters are used as those used in [8] to compile the C code for the MWDS problem. Furthermore, the parametric values for the MWDS in EA-HH are listed in Table 2. We have compared the EA-HH approach for the MWDS problem with Raka-ACO [28], HGA [40], ACO-LS [40], ACO-PP-LS [40], EA/G-IR [8], HMA [32], and MSRLS$_o$ [7] for Type I and Type II instances.

In Table 2, it can be observed that all the low-level heuristics are problem independent, except LH_6, and do not use any problem domain knowledge. However, in [28, 40, 8, 32, 7], all of them have problem-dependent operators and they used problem domain knowledge to generate a new solution. The advantage of having a generic operator is that it can be applied on any type of problem in the same domain. However, both the SPP and MWDS problem belong to the domain of subset selection problem, but having opposite objectives. But the same low-level heuristics are used for both the SPP and MWDS problem even though the former one has a maximization objective whereas later one has a minimization objective.

Tables 11 and 12 present the computational results of the state-of-the-art approaches Raka-ACO [28], HGA [40], ACO-LS [40], ACO-PP-LS [40], EA/G-IR [8], HMA [32], and MSRLS$_o$ [7] and the proposed EA-HH approach for the MWDS problem on Type I and Type II instances. These tables present mean value (**Mean**), standard deviation (**SD**) and average total execution

time (**ATET**) for instance groups of ten instances with the same numbers of nodes ($|\mathcal{V}|$) and edges ($|\mathcal{E}|$). However, the SD values for Raka-ACO, HMA and MSRLS$_o$ approaches were not reported in the original paper. Moreover, HMA and MSRLS$_o$ did not report the ATET. Therefore, we did not report the SD and ATET for the respective approaches as well. The reason is that the MSRLS$_o$ approach was run until a fixed time limit of 3600s, which is much larger than that of any other approaches which are reported in these tables.

Out of 42 instance groups, EA-HH is better than HGA on all 42 instance groups. In comparison with ACO-LS, EA-HH is better on 40 instance groups and worse on two instance groups. In comparison with ACO-PP-LS, EA-HH is better on 39 instance groups and worse on 3 instance groups. In comparison with EA/G-IR, EA-HH is better on 30 instance groups and worse on 12 instance groups. EA-HH is better on 28 instance groups than HMA and worse than HMA on 14 instance groups. In comparison with MSRLS$_o$, EA-HH is better on 34 instance groups and worse on 4 instance groups. In [28], Javanovic et al. did not report computational results on the instance groups on the network with 300 nodes for both Type I and Type II. Therefore, we compared the performance on the reported instance groups only. On all 28 instance groups, the EA-HH outperformed the Raka-ACO approach. The average performance also indicates that the EA-HH approach exhibits better performance than all the other state-of-the-art approaches presented in the tables.

In terms of computational time, from Tables 11 and 12, it is evident that EA-HH has taken less computational time than HGA, ACO-LS, ACO-PP-LS, and EA/G-IR. For HMA and MSRLS$_o$, we did not report the computational times because their stopping criteria are different from those of other approaches. However, the EA-HH approach is bit slower than HMA but is computationally more significant than HMA. In the case of MSRLS$_o$, the stopping criterion is fixed at 3600s. Therefore, EA-HH is faster than the MSRLS$_o$ approach.

Table 11: Results of Raka-ACO [28], HGA [40], ACO-LS [40], ACO-PP-LS [40], EA/G-IR [8], HMA [32], MSRLS$_o$ [7] and EA-HH for large Type I instances.

$\|\mathcal{V}\|$	$\|\mathcal{E}\|$	**Raka-ACO**	**HGA**			**ACO-LS**			**ACO-PP-LS**			**EA/G-IR**			**HMA**	**MSRLS$_o$**	**EA-HH**		
			Mean	SD	ATET	Mean	SD	ATET	Mean	SD	ATET	Mean	SD	ATET	Mean	Mean	Mean	SD	ATET
300	300	NA	3255.20	74.13	52.64	3198.50	63.82	22.27	3205.90	70.39	19.10	3213.70	75.81	8.73	3199.30	3202.40	**3193.20**	69.80	5.24
300	500	NA	2509.80	69.21	49.77	2479.20	80.75	18.72	2473.3	84.44	16.30	2474.80	76.95	7.21	2464.40	2468.60	**2457.80**	77.19	4.21
300	750	NA	1933.90	81.23	44.42	1903.30	55.08	16.21	1913.90	64.69	14.30	1896.30	55.74	6.48	1884.60	1897.10	**1878.00**	57.81	3.12
300	1000	NA	1560.10	35.27	40.54	1552.50	32.51	14.27	1555.80	36.19	12.40	1531.00	28.43	5.70	1518.40	1539.20	**1512.50**	29.86	2.57
300	2000	NA	909.60	22.85	26.60	916.80	25.61	10.50	916.50	23.08	9.50	880.10	21.91	4.03	878.70	901.30	**875.90**	24.25	1.43
300	3000	NA	654.90	24.44	22.43	667.80	30.00	9.26	670.70	28.00	8.60	638.20	22.15	3.27	640.90	663.50	**630.30**	16.85	1.09
300	5000	NA	428.30	15.07	17.12	437.40	16.59	7.81	435.90	16.22	7.50	415.70	10.32	2.59	411.70	428.50	**408.30**	12.62	0.81
500	500	5476.30	5498.30	113.45	204.83	5398.30	100.57	89.57	5387.70	99.53	77.50	5380.10	89.37	37.52	5392.10	5389.20	**5379.50**	91.04	20.00
500	1000	4069.80	3798.60	92.08	250.38	3714.80	103.77	100.00	3698.30	85.61	81.00	3695.20	107.47	26.36	**3678.30**	3716.30	3696.60	97.06	15.47
500	2000	2627.50	2338.20	77.75	145.95	2277.60	60.20	60.13	2275.90	65.12	54.40	2264.30	84.53	25.54	**2223.70**	2291.20	2245.50	77.58	8.54
500	5000	1398.50	1122.70	30.96	75.35	1115.30	35.79	33.79	1110.20	41.94	30.10	1083.50	33.27	9.02	1074.20	1138.10	**1070.10**	32.67	3.78
500	10000	825.70	641.10	22.18	43.62	652.80	11.81	25.15	650.90	119.00	24.40	606.80	11.57	6.08	**595.40**	634.50	596.70	14.14	1.82
800	1000	8098.90	8017.70	141.01	841.64	8117.60	162.03	443.92	8068.00	178.60	409.20	7792.20	108.34	129.94	7839.90	**7833.70**	7841.60	110.61	86.34
800	2000	5739.90	5318.70	130.02	576.34	5389.90	151.14	301.57	5389.60	144.43	292.20	5160.70	76.92	102.09	**5100.70**	5224.60	5188.80	67.47	51.26
800	5000	3116.50	2633.40	69.07	346.05	26160	66.49	165.34	2607.90	62.02	148.90	2561.90	37.51	53.019	**2495.70**	2626.90	2535.60	53.63	21.83
800	10000	1923.00	1547.70	45.66	162.32	1525.70	32.40	97.28	1535.30	31.00	95.90	1497.00	33.41	31.17	**1459.80**	1526.20	1460.50	26.93	10.88
1000	1000	10924.40	11095.20	147.38	2193.48	11035.50	174.92	1023.79	11022.90	129.43	922.40	10771.70	122.15	249.82	10863.30	**10766.70**	10809.80	122.69	183.09
1000	5000	4662.70	3996.60	73.73	626.10	4012.00	81.91	341.12	4029.80	85.90	326.50	3876.30	64.70	107.41	**3742.80**	3947.00	3855.10	75.07	74.14
1000	10000	2890.30	2334.70	64.51	380.27	2314.90	64.03	207.39	2306.60	56.03	194.70	2265.10	51.68	63.22	**2193.70**	2283.20	2235.50	47.35	37.48
1000	15000	2164.30	1687.50	28.29	254.64	1656.30	44.23	162.73	1657.40	40.05	150.80	1629.40	30.04	45.86	**1590.90**	-	1613.80	27.01	20.83
1000	20000	1734.30	1337.20	30.97	174.50	1312.80	22.52	142.40	1315.80	24.10	139.20	1299.90	19.32	36.35	**1263.50**	-	1272.60	12.35	12.31
Average		3975.15	2981.88	66.16	310.90	4087.57	67.44	156.82	2963.25	70.75	144.52	2901.61	55.31	45.78	**2881.52**	3077.80	2893.22	54.22	26.96

Table 12: Results of Raka-ACO [28], HGA [40], ACO-LS [40], ACO-PP-LS [40], EA/G-IR [8], HMA [32], MSRLS$_o$ [7] and EA-HH for large Type II instances.

$\|\mathcal{V}\|$	$\|\mathcal{E}\|$	**Raka-ACO**	**HGA**			**ACO-LS**			**ACO-PP-LS**			**EA/G-IR**			**HMA**	**MSRLS$_o$**	**EA-HH**		
			Mean	SD	ATET	Mean	SD	ATET	Mean	SD	ATET	Mean	SD	ATET	Mean	Mean	Mean	SD	ATET
300	300	NA	375.60	24.79	69.70	371.10	23.14	21.06	371.10	23.44	19.20	**370.50**	23.14	9.01	373.90	371.20	372.00	23.46	6.89
300	500	NA	484.20	39.19	72.50	480.80	40.13	20.22	481.20	40.05	18.10	**480.00**	41.24	8.83	484.00	485.80	481.70	41.71	6.96
300	750	NA	623.80	52.76	64.33	621.60	48.76	19.07	618.30	48.90	17.00	**613.80**	52.25	7.57	620.10	634.90	616.90	51.97	5.84
300	1000	NA	751.10	75.91	59.28	744.90	77.80	17.68	743.50	74.20	16.00	**742.20**	72.28	8.96	745.10	755.80	744.50	72.90	5.10
300	2000	NA	1106.70	116.09	55.25	1111.60	114.61	15.31	1107.50	112.03	14.30	**1094.90**	106.64	6.67	1103.60	1118.00	1104.70	111.01	3.36
300	3000	NA	1382.10	125.93	46.13	1422.80	153.78	15.55	1415.30	167.50	13.60	**1359.50**	129.06	5.41	1378.90	1392.50	1380.10	140.79	2.50
300	5000	NA	1686.30	294.44	44.85	1712.10	291.41	12.24	1698.60	300.02	11.70	**1683.60**	294.89	4.02	1692.60	1701.30	1685.20	297.13	1.72
500	500	651.20	632.90	29.54	284.90	627.50	30.06	81.38	627.30	30.13	73.90	**625.80**	30.41	31.06	633.40	627.00	626.30	29.85	24.82
500	1000	1018.10	919.20	41.71	268.61	913.00	35.69	77.31	912.60	36.56	68.50	906.00	42.37	28.27	912.00	934.90	**904.00**	40.47	23.07
500	2000	1871.80	1398.20	131.90	233.33	1384.90	121.03	72.37	1383.90	121.77	65.00	1376.70	116.91	23.41	1394.10	1408.60	**1368.70**	118.50	15.86
500	5000	4299.80	2393.20	222.03	122.30	2459.10	272.38	51.34	2468.80	260.35	51.00	2340.30	210.28	17.36	2388.30	2401.70	**2335.80**	199.70	8.15
500	10000	8543.50	3264.90	4218.00	118.96	3377.90	470.35	37.41	3369.40	482.89	37.60	3216.40	389.63	10.80	3259.60	3261.50	**3215.00**	386.44	5.21
800	1000	1171.20	1128.20	48.22	1091.83	1126.40	51.56	300.80	1125.10	50.79	268.30	1107.90	45.43	132.36	1131.30	1126.20	**1114.40**	47.68	110.70
800	2000	1938.70	1679.20	74.70	927.69	1693.70	80.25	307.11	1697.90	80.26	282.20	1641.70	75.40	111.84	1681.90	1709.90	**1646.40**	74.54	90.06
800	5000	4439.00	3003.60	204.03	525.34	3121.90	227.35	227.41	3120.90	229.21	224.00	2939.30	213.91	68.14	2963.80	2990.40	**2896.40**	198.77	50.68
800	10000	8951.10	4268.10	379.71	387.92	4404.10	380.67	148.32	4447.90	371.23	148.20	4155.10	346.83	40.15	4226.60	4272.50	**4110.80**	326.91	29.76
1000	1000	1289.30	1265.20	30.99	2149.54	1259.30	33.44	574.95	1258.60	34.35	514.20	**1240.80**	30.45	202.08	1270.90	1247.60	1251.80	31.23	221.26
1000	5000	4720.10	3320.10	221.66	1112.49	3411.60	228.22	464.20	3415.10	209.28	461.30	3222.00	196.89	132.94	3317.60	3340.50	**3220.10**	196.64	122.33
1000	10000	9407.70	4947.50	330.77	739.45	5129.10	308.66	323.16	5101.90	306.17	324.40	4798.60	291.65	84.82	4937.90	4935.50	**4757.00**	315.86	67.46
1000	15000	14433.50	6267.60	463.09	617.16	6454.60	445.76	251.67	6470.60	467.53	251.80	5958.10	427.56	61.64	6122.50	NA	**5913.80**	444.10	51.35
1000	20000	19172.60	7088.50	659.71	536.83	7297.40	598.98	212.02	7340.80	604.06	213.80	6775.80	571.69	59.20	6842.90	NA	**6677.80**	574.34	39.68
Average		5850.54	2285.06	370.722	453.73	2339.30	192.10	154.79	2341.73	192.89	147.34	2221.38	176.61	50.22	2261.00	1827.15	***2210.63***^{c}	177.33	42.51

The average value ***2210.63***^{c} indicates that the average of, excluding mean of instances with number of nodes 1000 and the number of edges 15000 and 20000, the mean value of the **EA-HH** is better than the mean value of **MSRLS$_o$**.

Table 13 presents the Wilcoxon-signed-rank test for the EA-HH approach with Raka-ACO, HGA, ACO-LS, ACO-PP-LS, HMA and MSRLS$_o$ for the MWDS problem. Similar to the SPP, a two-tail Wilcoxon-signed-rank test is performed to investigate the significance level of the EA-HH approach with other approaches by setting the significance criterion to 1% (*p-value* $\leqslant$ *0.01*). From Table 13, it can be observed that the EA-HH approach is significantly better than the other approaches.

Similar to the SPP, a mean ranked test is performed over Raka-ACO, HGA, ACO-LS, ACO-PP-LS, HMA, MSRLS$_o$, and the EA-HH approaches and their ranks are 7.76, 5.55, 5.50, 5.05, 2.19, 3.00, 5.05, 1.88, respectively. A smaller value indicates better performance. From the mean ranked test, it is evident that the EA-HH outperformed all the other approaches.

Table 13: Wilcoxon-signed-rank test for the EA-HH with the other approaches for the MWDS problem

Approach	EA-HH vs.						
	N	W^+	W^-	Z	Z_c	*p-value*	*Significance*
Raka-ACO[28]	24	0.00	406.00	T=130	-1.96	0.0000	*yes*
HGA [40]	42	0.00	903.00	-5.645	-1.64	0.0000	*yes*
ACO-LS [40]	42	5.00	898.00	-5.583	-1.64	0.0000	*yes*
ACO-PP-LS [40]	42	6.50	896.50	-5.645	-1.64	0.0000	*yes*
EA/G-IR [8]	42	251.50	651.50	-2.501	-1.64	0.0062	*yes*
HMA [32]	42	252.00	651.00	-2.494	-1.64	0.0063	*yes*
MSRLS$_o$ [7]	38	20.00	721.00	-5.083	-1.64	0.0000	*yes*

8. Conclusions

In this paper, we proposed an evolutionary algorithm based hyper-heuristic (EA-HH) framework for the SPP and MWDS problem. In the EA-HH, at the higher level of hyper-heuristic, an evolutionary algorithm is employed as a search methodology and at the lower level exploratory and exploitative heuristics are used in the heuristics pool. In the evolutionary algorithm, an online-learning mechanism is employed to learn the performance of each heuristic at each generation. The online-learning mechanism was inspired by the UMD algorithm, which at each iteration estimates the performance of each heuristic and stores their performance in the form of a probability vector. Further, this probability vector is used to select a heuristic using the RWS method. Another contribution is the dynamic selection of parameters. Instead of fixing the parameters, the EA-HH allowed making changes in their values with generations. The computational results show the effectiveness of dynamic selection of parametric. The EA-HH was compared with the state-of-the-art approaches for the SPP: GRASP [14], ACO [17], and EA/G [11]. Further, the MWDS problem is used, mainly, to investigate the generality, effectiveness and robustness of the proposed EA-HH approach. In the case of the MWDS problem, [28, 40, 8, 32, 7] are the state-of-the-art-approaches used to compare with the EA-HH approach. This can be evident from the computational results in Tables 7 to 9 as well as Tables 11 and 12 that the same operators and same high-level search heuristic are used to solve the problems with opposite objectives.

As a future work, we intend to explore the capabilities of other search heuristics such as taboo search, genetic programming, artificial bee colony, and genetic algorithm, etc. at the higher level of hyper-heuristics. Further, we will explore the heuristic selection methodology and rigorously analyses the performance for new problems in different domains such as permutation and grouping problems.

Acknowledgements

This research work was supported by a grant from the National Research Foundation (NRF) of Korea, funded by the Korean government (MSIP) (No. 2016R1A2A1A05005306). Authors are also grateful to four anonymous reviewers and the Editor-in-Chief for their valuable comments and suggestions which has helped in improving the quality of this paper.

References

[1] A. Almutairi, E. Özcan, A. Kheiri, W.G. Jackson, Performance of selection hyper-heuristics on the extended hyflex domains, in: Computer and Information Sciences, Springer International Publishing, Cham, 2016, pp. 154–162.

[2] S. Asta, D. Karapetyan, A. Kheiri, E. ÃŰzcan, A.J. Parkes, Combining monte-carlo and hyper-heuristic methods for the multi-mode resource-constrained multi-project scheduling problem, Information Sciences 373 (2016) 476–498.

[3] S. Asta, E. Özcan, A tensor-based selection hyper-heuristic for cross-domain heuristic search, Information Sciences 299 (2015) 412–432.

[4] S. Baluja, Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning, Technical Report, Pittsburgh, PA, USA, 1994.

[5] E.K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, J.R. Woodward, A classification of hyper-heuristic approaches, A classification of hyper-heuristic approaches, Springer US, Boston, MA, 2010, pp. 449–468.

[6] F. Caraffini, F. Neri, M. Epitropakis, Hyperspam: A study on hyper-heuristic coordination strategies in the continuous domain, Information Sciences 477 (2019) 186–202.

[7] D. Chalupa, An order-based algorithm for minimum dominating set with application in graph mining, Information Sciences 426 (2018) 101–116.

[8] S.N. Chaurasia, A. Singh, A hybrid evolutionary algorithm with guided mutation for minimum weight dominating set, Applied Intelligence 43 (2015) 512–529.

[9] S.N. Chaurasia, A. Singh, A hybrid heuristic for dominating tree problem, Soft Computing 20 (2016) 377–397.

[10] S.N. Chaurasia, A. Singh, Hybrid evolutionary approaches for the single machine order acceptance and scheduling problem, Applied Soft Computing Journal 52 (2017) 725–747.

[11] S.N. Chaurasia, S. Sundar, A. Singh, A hybrid evolutionary approach for set packing problem, OPSEARCH 52 (2015) 271–284.

[12] S.S. Choong, L.P. Wong, C.P. Lim, Automatic design of hyper-heuristic based on reinforcement learning, Information Sciences 436-437 (2018) 89–107.

[13] P. Cowling, G. Kendall, E. Soubeiga, A hyperheuristic approach to scheduling a sales summit, in: Practice and Theory of Automated Timetabling III, Springer Berlin Heidelberg, Berlin, Heidelberg, 2001, pp. 176–190.

[14] X. Delorme, X. Gandibleux, J. Rodriguez, GRASP for set packing problems, European Journal of Operational Research 153 (2004) 564–580.

[15] X. Delorme, J. Rodriguez, X. Gandibleux, Heuristics for railway infrastructure saturation, Electronic Notes in Theoretical Computer Science, 2001.

[16] X. Gandibleux, S. Angibaud, X. Delorme, J. Rodriguez, An ant algorithm for measuring and optimizing the capacity of a railway infrastructure, in: In N. Monmarché, F. Guinand, and P. Siarry, editors, Artificial Ants, volume 1, ISTE Ltd and John Wiley & Sons Inc, 2010, pp. 175–203.

[17] X. Gandibleux, X. Delorme, V. T'Kindt, An ant colony optimisation algorithm for the set packing problem, in: ANTS 2004, volume 3172, Berlin, pp. 49–60.

[18] M. Garey, D. Johnson, Computers and intractability: a guide to the theory of NP-Completeness, Freeman, San Francisco, 1979.

[19] F. Glover, Heuristics for integer programming using surrogate constraints, Decision sciences 8 (1977) 156–166.

[20] M. Gondran, M. Minoux, Graphes et algorithmes, Eyrolles, France, 1995.

[21] G. Gottlob, G. Greco, Decomposing combinatorial auctions and set packing problems, J. ACM 60 (2013) 24:1–24:39.

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

[23] M. Gulek, I.H. Toroslu, A dynamic programming algorithm for tree-like weighted set packing problem, Information Sciences 180 (2010) 3974–3979.

[24] M. Hauschild, M. Pelikan, An introduction and survey of estimation of distribution algorithms, Swarm and Evolutionary Computation 1 (2011) 111–128.

[25] O. Heismann, R. Borndörfer, A generalization of odd set inequalities for the set packing problem, in: Operations Research Proceedings 2013, Springer International Publishing, 2014, pp. 193–199.

[26] W.G. Jackson, E. ÃŰzcan, R.I. John, Fuzzy adaptive parameter control of a late acceptance hyper-heuristic, in: 2014 14th UK Workshop on Computational Intelligence (UKCI), pp. 1–8.

[27] W. Jia, C. Zhang, J. Chen, An efficient parameterized algorithm for m-set packing, Journal of Algorithms 50 (2004) 106–117.

[28] R. Jovanovic, M. Tuba, D. Simian, Ant colony optimization applied to minimum weight dominating set problem, in: Proceedings of the 12th WSEAS international conference on Automatic control, modelling and simulation (ACMOS'10), World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, USA, 2010, pp. 322–326.

[29] A. Kheiri, E. Özcan, An iterated multi-stage selection hyper-heuristic, European Journal of Operational Research 250 (2016) 77–90.

[30] S.H. Kim, K.K. Lee, An optimization-based decision support system for ship scheduling, Computers and Industrial Engineering 33 (1997) 689–692.

[31] P. Larraanaga, J.A. Lozano, Estimation of distribution algorithms: A new tool for evolutionary computation, Kluwer Academic Publishers, Norwell, MA, USA, 2001.

[32] G. Lin, W. Zhu, M.M. Ali, An effective hybrid memetic algorithm for the minimum weight dominating set problem, IEEE Transactions on Evolutionary Computation 20 (2016) 892–907.

[33] R. Lusby, J. Larsen, D. Ryan, M. Ehrgott, Routing trains through railway junctions: A new set packing approach, Transportation Science 45 (2011) 228–245.

[34] A. Merel, X. Gandibleux, S. Demassey, A collaborative combination between column generation and ant colony optimization for solving set packing problems, in: MIC 2011: The IX Metaheuristics International Conference, Italy, pp. 25–28.

[35] A. Mingozzi, V. Maniezzo, S. Ricciardelli, L. Bianco, An exact algorithm for the project scheduling with resource constraints based on a new mathematical formulation, Management Science 44 (1998) 714–729.

[36] H. Mühlenbein, The equation for response to selection and its use for prediction, Evolutionary Computation 5 (1997) 303–346.

[37] G. Nemhauser, L. Wolsey, Integer and combinatorial optimization, Wiley-Interscience, New York NY, USA, 1999.

[38] T.D. Nguyen, A fast approximation algorithm for solving the complete set packing problem, European Journal of Operational Research 237 (2014) 62–70.

[39] M.W. Padberg, On the facial structure of set packing polyhedra, Mathematical Programming 5 (1973) 199–215.

[40] A. Potluri, A. Singh, Hybrid metaheuristic algorithms for minimum weight dominating set, Applied Soft Computing 13 (2013) 76–88.

[41] M. Resende, C. Ribeiro, A GRASP with path-relinking for private virtual circuit routing, Networks 41 (2003) 104–114.

[42] M. Rönnqvist, A method for the cutting stock problem with different qualities, European Journal of Operational Research 83 (1995) 57–68.

[43] F. Rossi, S. Smriglio, A set packing model for the ground holding problem in congested networks, European Journal of Operational Research 131 (2001) 400–416.

[44] N.R. Sabar, G. Kendall, Population based monte carlo tree search hyper-heuristic for combinatorial optimization problems, Information Sciences 314 (2015) 225–239.

[45] M. Sviridenko, J. Ward, Large neighborhood local search for the maximum set packing problem, in: Automata, Languages, and Programming, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 792–803.

[46] G. Syswerda, Uniform crossover in genetic algorithms, in: Proceedings of the Third International Conference on Genetic Algorithms, Morgan Kaufmann, 1989, pp. 2–9.

[47] A. Tajima, S. Misono, Using a set packing formulation to solve airline seat allocation/reallocation problems, Journal of the Operations Research Society of Japan 42 (1999) 32–44.

[48] D. Wolpert, W. Macready, No free lunch theorems for optimization, IEEE Transactions on Evolutionary Computation 1 (1997) 67–82.

[49] C. Xu, G. Zhang, The path set packing problem, in: Computing and Combinatorics, Springer International Publishing, Cham, 2018, pp. 305–315.

[50] Q. Zhang, J. Sun, E. Tsang, An evolutionary algorithm with guided mutation for the maximum clique problem, IEEE Transactions on Evolutionary Computation 9 (2005) 192–200.

Declarations of interest

None