OPSEARCH (2020) 57:1164–1204
https://doi.org/10.1007/s12597-020-00463-8

APPLICATION ARTICLE

Two multi-start heuristics for the *k*-traveling salesman problem

Venkatesh Pandiri[1] · **Alok Singh**[1]

Accepted: 4 June 2020 / Published online: 7 July 2020

Abstract

This paper is concerned with the *k*-traveling salesman problem (*k*-TSP), which is a variation of widely studied traveling salesman problem (TSP). Given a set of n cities including a home city and a fixed value k such that $1 < k \leq n$, this problem seeks a tour of minimum length which starts and ends at the home city and visits k cities (including the home city) exactly out of these n cities. Finding a feasible solution to *k*-TSP involves finding a subset of k cities including the home city first, and then a circular permutation representing a tour of these k cities. In this paper, we have proposed two multi-start heuristic approaches for the *k*-TSP. The first approach is based on general variable neighborhood search algorithm (GVNS), whereas the latter approach is a hyper-heuristic (HH) approach. A variable neighborhood descent strategy operating over two neighborhood structures is utilized for doing the local search in the GVNS. As part of the hyper-heuristic, two low level heuristics are considered. To the best of our knowledge, these are the first metaheuristic and hyper-heuristic approaches for the *k*-TSP. To evaluate the performance of our approaches, a set of benchmark instances is created utilizing instances from TSPLIB. Computational results on these benchmark instances show HH approach to be better than GVNS approach.

Keywords *k*-traveling salesman problem · Traveling salesman problem · General variable neighborhood search · Hyper-heuristic · Heuristic

✉ Alok Singh
alokcs@uohyd.ernet.in

Venkatesh Pandiri
venkatesh78.p@gmail.com

[1] School of Computer and Information Sciences, University of Hyderabad, Hyderabad, Telangana 500 046, India

1 Introduction

The k-traveling salesman problem (k-TSP) is a variation of the well known traveling salesman problem (TSP). Given a set of n cities including a home city and a fixed value $1 < k \leq n$, the k-TSP consists in finding a subset of k cities including the home city and a tour visiting each city of this subset exactly once so that this tour has minimum length among all such tours over any subset of k cities that includes the home city. The TSP can be considered as a special case of k-TSP with $k = n$. On the other hand, k-TSP in itself can be considered as a special case of prize collecting traveling salesman problem (PCTSP) [1] where each city is associated with a prize and a penalty. The salesman collects the prize associated with a city in case he visits that city, and incurs the penalty associated with a city in case he do not visit that city. The goal of the PCTSP is to minimize the sumtotal of distance traveled by the salesman and total penalties incurred while collecting a specified minimum aggregate of prize. The quota traveling salesman problem (QTSP) is a special case of the PCTSP, where there is no penalty for not visiting a city, i.e., penalties for all cities can be considered as zero. The k-TSP can be considered as a special case of the QTSP, where the prize associated with each city is 1 and the specified minimum aggregate of prize is k [2]. The k-TSP is $\mathcal{NP}$-hard as it generalizes the TSP whose $\mathcal{NP}$-hardness is well known. This problem finds applications in those situations where there are not enough resources to visit all the cities, like in the design of distribution networks and rural healthcare delivery.

Some constant factor approximation algorithms exist in the literature for the k-TSP [3–8]. The approximation algorithm presented in [7] achieves the best known approximation ratio of 2. However, the k-TSP did not receive as much attention from the researchers as other TSP variants in spite of its potential applications in resource constrained environments. In fact, no metaheuristic approach exists in the literature for the k-TSP. This served as the motivation to develop the metaheuristic approaches presented in this paper.

Solving the k-TSP involves two aspects, viz. subset selection (selecting a subset containing k cities including home city) and permutation (finding the best circular permutation of the k cities belonging to the selected subset). Any solution approach for k-TSP has to deal with both of these aspects in an appropriate manner in order to be effective over a wide range of instances. No matter how good the strategy is for subset selection in an approach for k-TSP, it will not yield a good solution for k-TSP if strategy to deal with permutation aspect is not designed properly. Likewise, an approach using a very good strategy to deal with permutation aspect but a weak strategy for subset selection, will not succeed either. Further, relative importance of these two aspects may vary from one instance of the problem to another, and hence, an approach needs to adapt swiftly as per the characteristic of the instance at hand in order to be efficient. Keeping all these facts in mind, we have developed two multi-start heuristic approaches for k-TSP. Our first approach is a simple but effective general variable neighborhood search (GVNS) algorithm, which incorporates the variable neighborhood descent as local search. This approach utilizes two neighborhood structures one based on subset selection, whereas the other based on permutation. Our second approach is based on hyper-heuristic approach which again incorporates two low level heuristics. The first low level heuristic caters to subset selection aspect, whereas the second low level heuristic caters to permutation aspect. Two versions of hyper-heuristic approach are presented in this paper.

The remainder of this paper is organized in the following manner: Sect. 2 formally defines the k-TSP. Section 3 describes our multi-start general variable neighborhood search based approach for the k-TSP, whereas Sect. 4 describes our multi-start hyper-heuristic approach for the k-TSP. Computational results and their analysis are presented in Sect. 5. Finally, Sect. 6 concludes the paper by listing the contributions of the paper and some directions in which future research can be carried out based on the work reported in this paper.

2 Problem definition

Given a complete, edge-weighted, undirected graph $G = (V, E)$, where $V = \{1, 2, \ldots, n\}$ is the set of n nodes in which the first node '1' represents the home city and the remaining $n-1$ nodes represent other cities, $E = \{(i,j) | i,j \in V\}$ is the set of edges and each edge $(i,j) \in E$ has an associated distance d_{ij}. The k-TSP seeks a minimum length Hamiltonian cycle over all the subgraphs induced by the subsets of V with exactly k nodes including the home city. Throughout this paper, we will use node and city interchangeably. Let V' denotes such a subset of k nodes. We will use binary variable y_i to specify whether a node i belongs to V' ($y_i = 1$) or not ($y_i = 0$), and another binary variable x_{ij} to specify whether an edge (i, j) belongs to the Hamiltonian cycle over V' ($x_{ij} = 1$) or not ($x_{ij} = 0$). With the help of these notational conventions, the k-TSP can be formulated in the following manner:

$$\text{Minimize} \quad \sum_{i \in V} \sum_{j \in V} d_{ij} x_{ij} \tag{1}$$

subject to:

$$\sum_{i \in V} y_i = k, \tag{2}$$

$$\sum_{i \in V} x_{1i} = 1 = \sum_{i \in V} x_{i1}, \tag{3}$$

$$\sum_{(k,i) \in E} x_{ki} + \sum_{(i,j) \in E} x_{ij} = 2y_i \quad \forall i \in V, \tag{4}$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1, \quad \forall S \subset V' \subset V \tag{5}$$

$$x_{ij}, y_i \in \{0, 1\} \quad \forall (i,j) \in E, i \in V. \tag{6}$$

The objective function of the k-TSP is represented by Eq. (1) which minimizes the total length of the cycle. Equation (2) ensures that k cities are exactly visited and Eq. (3) enforces the constraint that the tour should start and end at the first city (i.e., home city). Equation (4) enforces the constraints on the in-degree and the out-degree of the visited nodes. Equation (5) is the sub tour elimination constraint. Equation (6) restricts the values of decision variables x_{ij} and y_i to either 0 or 1.

3 Multi-start general variable neighborhood search approach

Before providing the details of the proposed multi-start general variable neighborhood search approach for the k-TSP, an overview of variable neighborhood search, variable neighborhood descent search and general variable neighborhood search will be provided.

Mladenović and Hansen [9] proposed variable neighborhood search (VNS) which is a metaheuristic that searches different neighborhood structures in a systematic manner. VNS utilizes both deterministic and stochastic changes of the neighborhood structures. The VNS has two phases, viz. local search and shake phases. The local search phase is responsible for exploitation (finding an improved solution from a given solution). On the other hand, the shake phase is responsible for exploration (escaping from locally optimal solution).

Algorithm 1 provides the pseudo-code of basic VNS. Let S_P and $f(S_P)$ represents a solution for a problem under consideration P and its fitness value, and let $\mathcal{N} = \{N_1, N_2, \ldots, N_{i_{max}}\}$ represents a set of i_{max} different neighborhood structures. The VNS begins by generating an initial solution, and setting the neighborhood indicator variable i to 1, then an iterative process ensues. Usually, but not always, this initial solution is generated in a random fashion. During each iteration, a random solution S'_P in the neighborhood N_i of current solution S_P is generated via shake phase. The local search phase takes S'_P as input and make an attempt to find an improved solution S''_P in the neighborhood N_i of S'_P. In case S''_P is found to be better than the current solution S_P, then S''_P replaces S_P as the current solution, and the VNS moves to the first neighborhood structure (i.e., i is reset to 1) in the next iteration, otherwise the VNS continues with the next neighborhood structure (assuming after $N_{i_{max}}$, the next neighborhood structure is N_1) in the next iteration. This process is repeated as long as termination criterion is not met.

Algorithm 1: Pseudo-code of basic VNS

Input: VNS parameters and a problem instance

Output: Best solution obtained by VNS

```
S_P ← Generate_Initial_Solution();
while termination criterion is not met do
    i ← 1;
    while i ≤ i_max do
        S'_P ← Shake(S_P, N_i);
        S''_P ← Local_Search(S'_P, N_i);
        if f(S''_P) is better than f(S_P) then
            S_P ← S''_P;
            i ← 1;
        else
            i ← i + 1;
return S_P;
```

Algorithm 2: Pseudo-code of VND

Input: VND parameters and a problem instance

Output: Best solution obtained by VND

```
S_P ← Generate_Initial_Solution();
repeat
    i ← 1;
    while i ≤ i_max do
        S'_P ← Local_Search(S_P, N_i);
        if f(S'_P) is better than f(S_P) then
            S_P ← S'_P;
            i ← 1;
        else
            i ← i + 1;
until no improvement exists in any of i_max neighborhoods;
return S_P;
```

The variable neighborhood descent (VND) [9] is a special case of the basic VNS, where there is no shake phase, and as a result different neighborhood structures are explored in a deterministic manner. Further, the termination criterion is non-existence of a solution better than the current solution in any of the i_{max} neighborhoods. Algorithm 2 provides the pseudo-code of VND. Clearly, VND returns a solution that is locally optimal with respect to all i_{max} neighborhoods. The use of VND for local search is quite common due to the fact that the chances of obtaining a good solution is high by using it to explore multiple neighborhood structures in a systematic manner in comparison to methods utilizing a single neighborhood structure. When VND is used as a local search, it starts with a solution that is passed to it as input instead of an initial (random) solution.

The general variable neighborhood search [10] is a variation of VNS [9], that makes use of VND for the local search phase. The general variable neighborhood search has been applied successfully to solve numerous combinatorial optimization problems.

3.1 Proposed multi-start general variable neighborhood search

Inspired by the success of the general variable neighborhood search in solving several variations of the TSP [11–14], we have developed a multi-start general variable neighborhood search approach for the *k*-TSP which incorporates a VND strategy as local search. This VND strategy explores the different neighborhood structures by following the first improvement strategy [10]. Algorithm 3 provides the pseudo-code of our multi-start GVNS approach. Hereafter, this approach will be referred to as GVNS. GVNS being a multi-start approach, restarts N_{rst} number of times. The description about the salient features of our GVNS approach can be found in the following subsections. It is to be noted that a preliminary version of this GVNS approach for *k*-TSP has been presented in [15].

3.1.1 Solution encoding and fitness

We have represented a solution by the linear permutation of *k* cities constituting the tour where home city always occupies the first position. Actually, a tour is a circular permutation and *k* linear permutations corresponds to a single circular permutation comprising *k* cities. Hence, there is a redundancy when a linear permutation is used to represent a circular permutation. By fixing the first position permanently for the home city, this redundancy got eliminated. It is to be noted that no component of GVNS can modify home city or its position.

The objective function (Eq. 1) itself is used as the fitness function, i.e., the fitness of a solution is the total distance traveled by the salesman. As *k*-TSP is a minimization problem, a solution having a lower value of the fitness function is regarded as more fit than a solution having a higher value.

3.1.2 Initial solution generation

The initial solution is generated in an iterative manner. The procedure begins by inserting the home city at the first position in the tour and then during each iteration an unvisited city is chosen randomly and inserted into the tour at some random position. This procedure repeats as long as feasibility condition remains unsatisfied, i.e., k cities have not been visited.

3.1.3 Variable neighborhood descent (VND)

VND is used as a local search very often due to its ability to explore different neighborhood structures systematically. For VND to be effective, the neighborhood structures have to be defined as per the characteristics of the problem at hand. We have defined two neighborhood structures, viz. N_1 and N_2 for the k-TSP after giving due consideration to its characteristics. The N_1 deals with the subset selection characteristic, whereas the N_2 deals with the permutation characteristic. These two neighborhoods are described below:

1. *Neighborhood* N_1: The first neighborhood N_1 is based on exchanging a visited city with an unvisited city. In this neighborhood, to create a new solution S' in the neighborhood of an existing solution S, a visited city is removed from S and an unvisited city is added at the best possible position. By best possible position, we mean a position that yields a solution of least cost once the city is added to that position. To find the best possible position, we have to explore all the $k-1$ positions.
2. *Neighborhood* N_2: The second neighborhood N_2 is based on swapping the positions of two visited cities in the tour. In this neighborhood, to create a new solution S' in the neighborhood of an existing solution S, two visited cities swap their positions.

Owing to the fact that the neighborhood structures at the beginning are explored more often than the latter ones, the performance of VND is influenced by the order in which various neighborhood structures are explored. We explore N_1 first and then N_2 (In fact, we have named them so based on this decision only). As mentioned previously, the first improvement strategy is used by the VND. In any of our two neighborhoods, the moment VND finds a solution better than the current solution, the current solution is immediately replaced with this new better solution, and the VND moves to the neighborhood N_1 of this new current solution. VND continues until both N_1 & N_2 are completely explored without finding any improved solution. The current solution at this juncture is locally optimum with respect to both N_1 & N_2 and is returned as the solution found by VND.

As mentioned already, our two neighborhoods are explored according to a first improvement strategy. Once, an improved solution is encountered in the neighborhood, current solution will be replaced with the improved solution and the search process begins exploring neighborhood N_1 of this newly improved solution. In the worst case of exploring neighborhood N_1 as per the exchange move, all the $k-1$ cities (excluding the home city) in the tour may have to be tried one after the other for exchange with each of the $n-k$ unvisited cities. For identifying the best position for these $n-k$ cities, all the $k-1$ positions (excluding the position 1 reserved for home city) in the tour need to be tried for possible insertion. Therefore, exploring the neighborhood N_1 requires $(k-1) \times (n-k) \times (k-1)$ operations which has the complexity of $\mathcal{O}(nk^2)$. Likewise, in the worst case of exploring neighborhood N_2 as per the swap move, each of the $k-1$ cities in tour may have to be tried one after the other for swap with other $k-2$ cities in the tour. Therefore, exploring the neighborhood N_2 has the complexity of $\mathcal{O}((k-1) \times (k-2)) = \mathcal{O}(k^2)$.

Algorithm 3: Pseudo-code of GVNS Approach for k-TSP

```
Input: GVNS parameters and a k-TSP instance
Output: Best solution obtained by GVNS
best ← ∞;
for j ← 1 to N_rst do
    S_{k-TSP} ← Generate_Initial_Solution();
    if f(S_{k-TSP}) < f(best) then
        best ← S_{k-TSP};
    i ← 1;
    while termination criterion is not met do
        S'_{k-TSP} ← Shake(S_{k-TSP}, N_i);
        S''_{k-TSP} ← VND(S'_{k-TSP});
        if f(S''_{k-TSP}) < f(S_{k-TSP}) then
            S_{k-TSP} ← S''_{k-TSP};
            i ← 1;
        else if i < i_max then
            i ← i + 1;
        else
            i ← 1;
    if f(S_{k-TSP}) < f(best) then
        best ← S_{k-TSP};
return best;
```

4 Multi-start hyper-heuristic approach for the *k*-TSP

In this section, first we will provide an overview of hyper-heuristics and then the details of proposed multi-start hyper-heuristic approach for the k-TSP will be presented.

Now a days, hyper-heuristics are getting much focus from researchers because of their ability to swiftly adapt as per the problem instance under consideration, thereby yielding high quality solutions for a wide range of instances of a problem [16]. The term *hyper-heuristic* was first used in a technical report by Denzinger et al. [17] as a strategy to combine a range of artificial intelligence methods for automated theorem proving, and does not provide any definition of hyper-heuristics. However, the basic idea of automating the design and/or selection of heuristics is proposed in early 1960s by Fisher [18] and Crowston et al. [19]. In Cowling et al. [20], hyper-heuristics is described as the heuristics to choose the heuristics in the context of combinatorial optimization. According to Burke et al. [16], the hyper-heuristics can be described as high level strategies that handle a pool of low-level heuristics and work either by choosing a heuristic or producing a new heuristic from the components of available heuristics at each step in the search process and utilizing the heuristic chosen/produced. A hyper-heuristic and a metaheuristic differ fundamentally. A hyper-heuristic works in the search space of heuristics, whereas a metaheuristic directly works in the search space of solutions to the problem under consideration. A hyper-heuristic tries to find the most appropriate heuristic to solve the problem under consideration in the search space of available heuristics, whereas a metaheuristic tries to find the best solution in the search space of solutions to the problem under consideration [21]. The motivation of developing hyper-heuristics arises from the fact that relative performance of different heuristics may not be same for all the instances of a problem, and even for the same instance, the performance of an individual heuristics may differ in different stages of the search process. Therefore, one may get better solutions if several heuristics are utilized in an appropriate manner. Here it is pertinent to mention that a low level heuristic within a hyper-heuristic can be a metaheuristic or a metaheuristic may employ a hyper-heuristic for local search/neighborhood search.

Based on their purpose, hyper-heuristics can be of two types.

- Selective hyper-heuristics: methodologies for choosing/selecting from available low-level heuristics.
- Generative hyper-heuristics: methodologies for producing new heuristics using elements of available low-level heuristics.

We have utilized a selective hyper-heuristic and Fig. 1 illustrates the general framework of such hyper-heuristics, where domain barrier acts as insulator between high level search strategy and low level heuristics. The high level search strategy selects and applies the low level heuristic by considering only the domain independent information. However, the performance of a hyper-heuristic can definitely be influenced by what domain-specific heuristics are available as low level heuristics.

The abundant literature on hyper-heuristics proves its effectiveness in solving combinatorial optimization problems, particularly when dealing with a wide range of instances having differing characteristics as it raises the level of generality. An excellent survey on hyper-heuristics and their applications can be found in [16].

4.1 Proposed hyper-heuristic approach

The hyper-heuristics individually and in hybridization with other metaheuristics have already been successfully applied for solving several variations of the TSP, e.g. [23–26]. Motivated by the success of these approaches, we have developed a multi-start hyper-heuristic approach for k-TSP, where two low level heuristics are used.

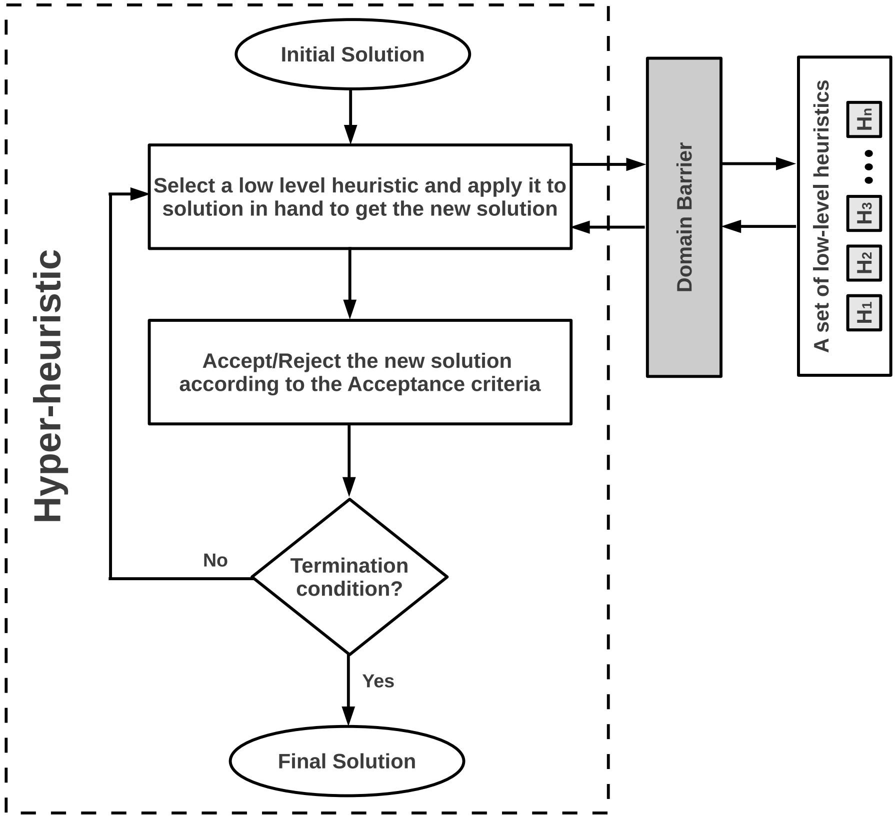


Fig. 1 Framework of selective hyper-heuristic [20, 22]

The number of restarts is governed by parameter N_{rst}. Hereafter, this approach will be referred to as HH.

Following subsections describe the salient features of our HH approach for the *k*-TSP.

4.1.1 Solution encoding and fitness

Solution encoding and fitness function is same as used for GVNS (Sect. 3.1.1). Further, no component of HH can change home city or its position.

4.1.2 Generation of initial solutions

Each initial solution is generated in the same manner as described in Sect. 3.1.2 for GVNS.

4.1.3 Generation of neighboring solutions by using low level heuristics

An effective neighboring solution generation procedure has to take into account all the characteristics of the problem at hand and also has to maintain a proper balance among the different characteristics regarding the importance given to each of them. The hyper-heuristic generates a solution in the neighborhood of the current solution using one of the low level heuristics. Our hyper-heuristic employs two low-level heuristics H_1 and H_2 to tackle the subset selection and permutation characteristics respectively. These two heuristics are discussed below:

1. H_1: This heuristic deals with the subset selection characteristic of the k-TSP. This heuristic is a ruin-recreate heuristic which partially ruins the tour and then recreates it. Each city in the tour is removed with probability ρ_r. All such removed cities are added to the set of unvisited cities which already contains all those cities which are not part of the tour. A city, which increases the cost of the tour by the least amount when inserted at its best position, is chosen from this set of unvisited cities. By best position of a city, we mean a position that yields the least increase in the cost of the tour after the city is inserted at that position in comparison to all other positions. For determining this city, all possible combinations of unvisited cities and insertion positions in the tour need to be checked. The city so determined is inserted into the tour at its best position. This procedure repeats till the feasibility condition is met, i.e., exactly k cities have been visited. Notice that, in this heuristic, there may be a change in the constituent cities of the tour and their respective positions too.
2. H_2: This heuristic tackles the permutation characteristic of the k-TSP. This heuristic is also a ruin-recreate heuristic which partially ruins the tour and then recreates it. Each city in the tour is removed with a probability ρ_r. All such removed cities are added to a set which is initially empty, and then, one-by-one, a city is chosen randomly from this set and inserted into the tour at its best possible position. This process continues until all the removed cities are inserted back into the tour. Notice that, in this heuristic, there is no change in the constituent cities, but there may be a change in their respective positions in the tour.

In the heuristic H_1, according to probability ρ_r, approximately $\rho_r \times k$ cities may be removed from the tour and added to the unvisited cities as part of the ruin procedure. The tour is made feasible again by following the recreate procedure which works in an iterative manner by inserting one city at a time to the tour. To identify the best city each time, all the unvisited cities (approximately $n - k + \rho_r \times k$ cities) need to be checked for insertion in all the available positions (approximately $k - 1 - \rho_r \times k$ positions) in the tour. Therefore, H_1 has the complexity of $\mathcal{O}(\rho_r \times k \times (n - k + \rho_r \times k) \times (k - 1 - \rho_r \times k)) = \mathcal{O}((n-k)k^2) = \mathcal{O}(nk^2)$. In the heuristic H_2, only the removed cities are inserted back into the tour at their best possible position. Hence, H_2 has the complexity of $\mathcal{O}(k^2)$.

Algorithm 4 provides the pseudo-code for generating a neighboring solution, where the values 1 and 2 respectively corresponds to H_1 and H_2. Basically, *Create_Neighbor(S, j)* takes as input a solution S & a parameter j indicating the choice of the heuristic and returns a neighboring solution S' by applying the chosen heuristic on S.

Algorithm 4: Pseudo-code for generating a neighboring solution

```
Input: A solution S
Output: A neighboring solution S'
function Create_Neighbor(S, j)
begin
if j == 1 then
    foreach city c in the tour as per their order do            } Procedure for applying H1.
        Generate a random number r such that 0 ≤ r ≤ 1;
        if r < ρr then
            Add c to the set of unvisited cities;
        else
            Copy c to the tour in S';
    while S' contains less than k cities do
        Insert an unvisited city into the tour of S' as per heuristic H1;
    return S';
else if j == 2 then
    foreach city c in the tour as per their order do            } Procedure for applying H2.
        Generate a random number r such that 0 ≤ r ≤ 1;
        if r < ρr then
            Add c to a set of unassigned cities;
        else
            Copy c to the tour in S';
    foreach city c in the set of unassigned cities in some random order do
        Insert c into the tour of S' as per heuristic H2 ;
    return S';
end function
```

4.1.4 Other features: selection mechanism, acceptance criteria

The selection mechanism plays a vital role in hyper-heuristic yielding a good solutions. There are many selection mechanisms available in the literature. Out of these, we have examined the hyper-heuristic with random and greedy selection mechanisms. In random selection mechanism, a low level heuristic is selected at random and used to return a solution at each step, whereas in greedy selection mechanism, all the low level heuristics are used at each step and the best solution among all the solutions obtained through these low level heuristics is returned. The heuristic whose solution is returned is deemed to be selected by greedy selection mechanism at that step. Motivation of choosing only these two mechanisms is due to the fact that the number of low-level heuristics are less (i.e., only 2) in our hyper-heuristic, and other mechanisms can be beneficial only when there are large number of low-level heuristics [16]. The two versions of HH with random and greedy selection mechanisms will be referred to as HH_RAND and HH_GREEDY respectively. In each iteration of HH_RAND, one of the two heuristics is used. Hence the complexity of an iteration is equal to the complexity of the heuristic used. The complexity of each iteration of HH_GREEDY is $\mathcal{O}(nk^2)$ as both heuristics are used in an iteration and H_1 has higher complexity.

There are many acceptance criterias available in the literature [16]. Out of these, we examined our hyper-heuristic with AA (all acceptance), OI (only improvement) acceptance criterias. Out of these two, only improvement (OI) criteria obtained the better results, and hence, the results with this criteria only are reported in this paper.

Algorithm 5 provides the pseudo-code of our HH approach, where N_{rst} is the number of times the hyper-heuristic is restarted. *Selection_Mechanism*(S, $\mathbb{S}_{LH}$) is a function that takes as input a solution S and a set $\mathbb{S}_{LH}$ of low level heuristics and returns a solution as per selection mechanism by making use of *Create_Neighbor()* function one or more times as the case may be.

Algorithm 5: Pseudo-code of HH approach for k-TSP

```
Input: HH approach parameters and a k-TSP instance
Output: Best solution obtained by HH
best ← ∞;
for j ← 1 to N_rst do
    S ← Generate_Initial_Solution();
    if f(S) < f(best) then
        best ← S;
    while termination criterion is not met do
        S' ← Selection_Mechanism(S, 𝕊_LH)
        if f(S') < f(S) then
            S ← S';
        if f(S) < f(best) then
            best ← S;
return best;
```

5 Computational results

Since our approaches, viz. GVNS, HH_RAND and HH_GREEDY are the first heuristic approaches for k-TSP, no benchmark instances exist for the k-TSP. As a result, we have to utilize the fresh instances for evaluating the performance of our approaches. Our benchmark instances for the k-TSP are derived from the instances publicly available in TSPLIB[1]. We have taken 76 instances from TSPLIB. These

[1] http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html

instances are in $n \times n$ distance matrix format and contain cities in the range from 14 to 783. The first city is always taken to be home city in these instances. With each of these 76 instances, we have considered three different scenarios, each having a specific value for k as per the value of n. These three scenarios are as follows:

1. **Small** – Small k value: $k = \lfloor \frac{1}{4}n \rfloor$
2. **Medium** – Medium k value: $k = \lfloor \frac{1}{2}n \rfloor$
3. **Large** – Large k value: $k = \lfloor \frac{3}{4}n \rfloor$

This leads to a total of 228 test cases for the k-TSP.

Our approaches have been implemented in C and executed on a Linux based 3.10 GHz Core-i5-2400 system with 4 GB RAM. All the approaches, viz. GVNS, HH_RAND and HH_GREEDY have been executed on each benchmark instance ten independent times. In the GVNS approach, number of restarts N_{rst} is set to 100, and the number of neighborhoods i_{max} is set to 2. In both HH_RAND and HH_GREEDY, number of restarts N_{rst} is again set to 100 like GVNS, and the probability ρ_r of a city to be removed from the tour is set to 0.05. For our approaches, we choose two termination criterias with short and long time to compare our approaches from the perspective of convergence behavior. The chosen termination criterias are

1. **Short Run(SR)**–Termination after a time of $0.05 \times n$ seconds
2. **Long Run(LR)**–Termination after a time of $0.2 \times n$ seconds

As the termination criteria is according to the time and our approaches are multi-start approaches, each execution of an approach after a fresh start is allowed for time $\frac{T}{N_{rst}}$, where T is total time allowed for the approach.

We have divided the results into six groups, each corresponding to a particular combination of a scenario and a termination criteria. These six groups are (Small, SR), (Medium, SR), (Large, SR), (Small, LR), (Medium, LR), and (Large, LR). All the groups have the same 76 TSPLIB instances, but the scenario and/or termination criteria vary from one group to another. Detailed instance-by-instance results can be found in Tables 4, 5, 6, 7, 8, 9 of "Appendix I". Here, we have compared the relative performance of our approaches on each of the six groups, and, overall, in terms of number of instances on which an approach obtained better, same or worse solution in comparison to other approaches. This performance comparison has been done according to the best and the average solution quality. The results are summarised in Table 1. This table presents the relative performance of our approaches in terms of the number of instances on which the algorithm on the left side (HH_RAND/HH_GREEDY) obtained better ('<'), same ('=') or worse solution ('>') than the algorithm on the upper side (GVNS/HH_GREEDY). Further, the overall performance of our approaches by considering all three scenarios and two termination criterias is reported in the last two rows named as 'Overall'. This table clearly shows that both HH_RAND and HH_GREEDY performed better than GVNS in terms of best and average solution quality both in all three scenarios under both the termination conditions. Further, the relative performance of HH_RAND and HH_GREEDY improve

Table 1 Performance comparison summary

Group		Best Solution Quality						Average Solution Quality					
		GVNS			HH_GREEDY			GVNS			HH_GREEDY		
		<	=	>	<	=	>	<	=	>	<	=	>
(Small, SR)	HH_RAND	40	33	3	28	36	12	49	20	7	40	21	15
	HH_GREEDY	36	33	7	–	–	–	46	21	9	–	–	–
(Medium, SR)	HH_RAND	54	19	3	32	24	20	62	11	3	55	13	8
	HH_GREEDY	54	20	2	–	–	–	61	11	4	–	–	–
(Large, SR)	HH_RAND	57	15	4	42	15	19	65	6	5	59	6	11
	HH_GREEDY	57	14	5	–	–	–	68	6	2	–	–	–
(Small, LR)	HH_RAND	43	31	2	30	37	9	49	20	7	38	20	18
	HH_GREEDY	38	33	5	–	–	–	50	21	5	–	–	–
(Medium, LR)	HH_RAND	53	19	4	35	22	19	63	11	2	49	12	15
	HH_GREEDY	54	20	2	–	–	–	63	11	2	–	–	–
(Large, LR)	HH_RAND	59	15	2	39	16	21	65	6	5	57	6	13
	HH_GREEDY	60	15	1	–	–	–	70	6	0	–	–	–
Overall	HH_RAND	306	132	18	206	150	100	353	74	29	298	78	80
	HH_GREEDY	299	135	22	–	–	–	358	76	22	–	–	–

as we move from small to large scenario. However, there is not much difference in relative performance under two different termination conditions. When it comes to the comparison between HH_RAND and HH_GREEDY, the former performed better than the latter.

The poor performance of the GVNS can be attributed to higher complexity of exploring the neighborhood N_1 which is used more often than N_2. Further, these two neighborhoods are explored multiple times in an iteration of GVNS. On the other hand, HH_RAND uses either H_1 or H_2 and HH_GREEDY uses both H_1 and H_2 in an iteration. Moreover, H_1 and H_2 can perform much more exploration than a single move in N_1/N_2. So when GVNS is executed for the same amount of time as HH_RAND and HH_GREEDY, it is not able to explore as much search space as explored by the latter two approaches leading to its poor performance. In a likewise manner, the superior performance of HH_RAND over HH_GREEDY can be explained. HH_GREEDY applies both H_1 and H_2 on the current solution, whereas HH_RAND applies only one of these two heuristics randomly. As a result, single iteration of HH_GREEDY requires more time than HH_RAND. As a result, HH_GREEDY gets lesser number of iterations in comparison to HH_RAND, when both HH_RAND over HH_GREEDY are executed for the same amount of time. In this situation, HH_GREEDY can outperform HH_RAND only when the gain of using both the heuristics in an iteration surpasses the loss HH_GREEDY incurs due to the lesser number of iterations. This is clearly not the case with k-TSP.

To understand the variation in the composition of the tour as per the value of k, we have taken instance eil51 and plotted the best solution found by our approaches

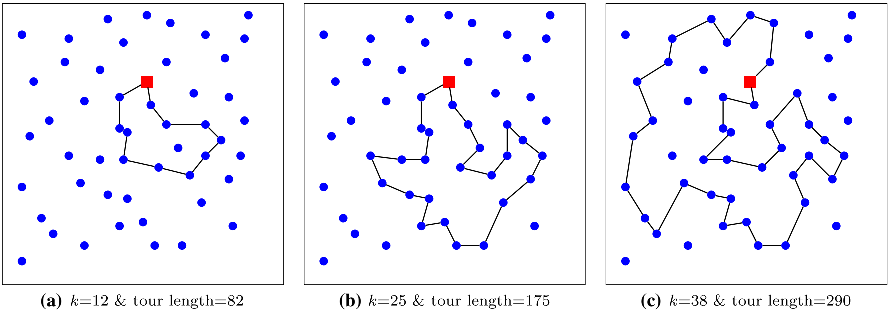


Fig. 2 Best solution found by our approaches on instance eil51 for different k values

on this instance under each of the three different scenarios corresponding to k values 12, 25 and 38. Please note that all our approaches found the same best solution for this instance under all three scenarios. Figure 2 presents these plots. In these plots, first city (i.e., home city) where the salesman has to start and end his tour is represented as red colored square, whereas remaining cities are represented as blue circles. These plots clearly show that the composition of the tour changes with the value of k. Obviously, the tour length will increase with increase in value of k.

To check whether there are significant differences among the performances of our approaches, we have used the two-tailed Wilcoxon signed rank test [27]. To perform this test, we have set the significance criteria to 5% (i.e. p-value ≤ 0.05) and made use of the calculator available online[2]. As part of this test, the difference between the normalized values of *Average* obtained by HH_GREEDY/HH_RAND and the compared approach is ranked. Tables 2 and 3 present the results of this test under SR and LR termination conditions respectively. In these tables, the column named *NWT/Total* reports the number of instances without tied values out of the total number of instances used in comparison. The column named R^+ reports the sum of ranks for the instances where the approach on the top of the table (HH_GREEDY/HH_RAND) performs better than its contender mentioned on the left side of the table, whereas the column R^- reports the sum of ranks for the instances where the approach on the top of the table (HH_GREEDY/HH_RAND) performs worse than its contender mentioned on the left side of the table. As the number of instances without tie exceeds thirty ($NWT > 30$) in all the cases, we have used the test statistic Z. The resulting Z value is compared with the critical value Z_{Cri} as per the Wilcoxon signed rank test [27]. A Z value not exceeding Z_{Cri} ($Z \leq Z_{Cri}$) indicates a significant difference between the performance of the two approaches being compared, or else the difference is insignificant. Conclusions that can be derived from these two tables are identical. Both show that HH_RAND is significant with respect to both GVNS & HH_GREEDY, and HH_GREEDY is significant with respect to GVNS.

We have also studied the convergence behavior of our approaches, and which can be found in "Appendix II".

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

Table 2 Results of Wilcoxon signed rank test between our approaches with SR termination criteria

	HH_RAND vs...						HH_GREEDY vs...					
	NWT/*Total*	R^+	R^-	Z	Z_{Cri}	Significant	*NWT*/*Total*	R^+	R^-	Z	Z_{Cri}	Significant
GVNS	191/228	18016	320	-11.566	-1.960	yes	190/228	17628	517	-11.272	-1.960	yes
HH_GREEDY	188/228	16226	540	-9.829	-1.960	yes						

Table 3 Results of Wilcoxon signed rank test between our approaches with LR termination criteria

	HH_RAND vs...						HH_GREEDY vs...					
	NWT/Total	R^+	R^-	Z	Z_{Cri}	Significant	*NWT/Total*	R^+	R^-	Z	Z_{Cri}	Significant
GVNS	191/228	18028	308	− 11.582	− 1.960	Yes	190/228	17871	274	− 11.592	− 1.960	Yes
HH_GREEDY	190/228	15446	2699	− 8.397	− 1.960	Yes						

6 Conclusions

In this paper, we have proposed two approaches, viz. GVNS and HH for the k-TSP based on general variable neighborhood search and hyper-heuristic respectively. The GVNS approach makes use of two neighborhood structures comprising exchange (N_1) & swap (N_2) operations and utilizes variable neighborhood descent as its principal component. These two neighborhood structures essentially tackle both the characteristics of the k-TSP, i.e., subset selection and permutation. Likewise, hyper-heuristic also incorporates two heuristics (H_1 and H_2) as low level heuristics to handle subset selection and permutation. Two versions of hyper-heuristic, viz. HH_RAND and HH_GREEDY are proposed based on two different selection mechanisms. To evaluate the performance of the various approaches proposed (viz. GVNS, HH_RAND and HH_GREEDY), various k-TSP test instances are derived from the publicly available instances in TSPLIB. All the details regarding how these instances are derived from TSPLIB instances have been provided in the paper to facilitate reuse of these instances. Computational results on these test instances show that both HH_RAND and HH_GREEDY performed better than GVNS. As far as comparison between HH_RAND and HH_GREEDY is concerned, the former performed better than the latter.

As our approaches are the first heuristic approaches for k-TSP, these approaches will be used as the baseline approaches for evaluating the performance of future heuristic approaches for this problem. Approaches analogous to our approaches can be developed for other problems also where there is a need to explore different neighborhoods and/or heuristics as per the characteristics of the problem. In future, we intend to investigate the reinforcement learning based hyper-heuristics for different variants of traveling salesman problem. Interested researchers can develop additional problem specific strategies to improve the performance of our approaches.

Acknowledgements The authors would like to express their sincere gratitude to two anonymous reviewers and the editor for their valuable comments and suggestions which helped in improving the quality of this manuscript. The first author is grateful to the Council of Scientific and Industrial Research (CSIR), Government of India for supporting his research through a Senior Research Fellowship. The second author acknowledges the support of the research Grant No. MTR/2017/000391 received under MATRICS scheme from Science and Engineering Research Board (SERB), Government of India.

Appendix I

This appendix presents the results obtained by our approaches on each TSPLIB instance under each of the six groups. Therefore, there are six tables, each corresponding to a group. Tables 4, 5 and 6 correspond to groups (Small, SR), (Medium, SR) and (Large, SR) respectively. These three tables report the performance of GVNS, HH_RAND and HH_GREEDY under short run. On the other hand, tables 7, 8 and 9 correspond to groups (Small, LR), (Medium, LR) and (Large, LR) respectively, and report the performance of our approaches under long run. In all these tables, the first column lists the instance names. The numerical values at the end of an instance name specifies the number of cities in the corresponding instance.

Table 4 Results of various approaches on each instance under small scenario with SR termination criteria

Instance	k	GVNS			HH_RAND			HH_GREEDY		
		Best	Worst	Average	Best	Worst	Average	Best	Worst	Average
a280	70	691	749	720.60	**686**	**718**	**703.00**	693	734	716.30
ali535	133	22648	34030	26606.60	13147	**14540**	13771.50	**12406**	15259	**13614.00**
att48	12	**1925**	**1925**	**1925.00**	**1925**	**1925**	**1925.00**	**1925**	**1925**	**1925.00**
att532	133	5373	6113	5661.60	**4014**	**4388**	**4234.60**	4024	4523	4271.90
bayg29	7	**332**	**332**	**332.00**	**332**	**332**	**332.00**	**332**	**332**	**332.00**
bays29	7	**400**	**400**	**400.00**	**400**	**400**	**400.00**	**400**	**400**	**400.00**
berlin52	13	**679**	**679**	**679.00**	**679**	**679**	**679.00**	**679**	**679**	**679.00**
bier127	31	**10619**	11029	10804.60	10687	**11014**	**10792.00**	10692	11028	10801.90
brazil58	14	**4965**	5030	4983.70	**4965**	**4965**	**4965.00**	**4965**	**4965**	**4965.00**
brg180	45	650	710	680.00	540	**560**	**551.00**	**520**	580	554.00
burma14	3	**359**	**359**	**359.00**	**359**	**359**	**359.00**	**359**	**359**	**359.00**
ch130	32	1149	1351	1261.10	**1130**	1291	**1226.30**	1167	**1290**	1238.60
ch150	37	1318	1350	1330.20	**1276**	**1336**	1316.90	**1276**	1359	**1312.40**
d198	49	5085	5257	5189.10	**5027**	**5102**	**5058.40**	5069	5147	5102.40
d493	123	10665	11769	11226.70	**9399**	**9713**	**9509.50**	9451	9814	9644.50
d657	164	18371	24327	20230.40	**12808**	**13623**	**13250.00**	13142	14110	13469.10
dantzig42	10	**145**	**145**	**145.00**	**145**	**145**	**145.00**	**145**	**145**	**145.00**
eil101	25	**107**	**108**	**107.30**	**107**	109	107.60	**107**	109	**107.30**
eil51	12	**82**	**82**	**82.00**	**82**	**82**	**82.00**	**82**	**82**	**82.00**
eil76	19	**102**	**102**	**102.00**	**102**	**102**	**102.00**	**102**	**102**	**102.00**
fl417	104	2304	4656	2795.10	**2258**	**2267**	**2260.70**	**2258**	2283	2264.50
fri26	6	**243**	**243**	**243.00**	**243**	**243**	**243.00**	**243**	**243**	**243.00**
gil262	65	593	633	612.60	**540**	**580**	**565.20**	555	590	575.20
gr120	30	**1308**	1362	1314.20	**1308**	**1318**	**1312.60**	**1308**	1385	1318.50
gr137	34	17802	18065	17996.50	**17399**	**17999**	**17780.20**	17510	18155	17874.80
gr17	4	**234**	**234**	**234.00**	**234**	**234**	**234.00**	**234**	**234**	**234.00**
gr202	50	8175	8573	8421.70	8175	8426	8332.30	**8142**	**8364**	**8263.40**
gr21	5	**324**	**324**	**324.00**	**324**	**324**	**324.00**	**324**	**324**	**324.00**
gr229	57	20604	24168	21836.10	**18589**	**19489**	**19062.20**	19129	19946	19438.80
gr24	6	**264**	**264**	**264.00**	**264**	**264**	**264.00**	**264**	**264**	**264.00**
gr431	107	16084	17874	17042.20	**14959**	**15966**	**15542.80**	15303	16233	15816.80
gr48	12	**874**	**874**	**874.00**	**874**	**874**	**874.00**	**874**	**874**	**874.00**
gr666	166	54939	98321	71424.80	**28464**	**30988**	29394.80	28630	31235	**29340.20**
gr96	24	10465	**10465**	**10465.00**	**10460**	10561	10474.00	**10460**	10786	10529.20
hk48	12	**2827**	**2827**	**2827.00**	**2827**	**2827**	**2827.00**	**2827**	**2827**	**2827.00**
kroA100	25	4998	5104	5023.10	**4970**	**5061**	**5011.80**	4998	5111	5030.20
kroA150	37	5725	**6182**	5936.10	**5690**	6454	5938.60	**5690**	6231	**5907.20**
kroA200	50	6438	7137	6775.60	**6220**	**6576**	**6393.50**	6272	6747	6493.90
kroB100	25	4353	4788	4588.80	**4305**	4684	**4473.70**	**4305**	**4605**	4536.60
kroB150	37	6410	**6854**	6588.00	**6071**	6938	**6468.20**	6482	6855	6666.60

Table 4 (continued)

Instance	k	GVNS			HH_RAND			HH_GREEDY		
		Best	Worst	Average	Best	Worst	Average	Best	Worst	Average
kroB200	50	**6388**	7625	7076.80	6630	**7060**	6884.30	6414	7280	**6827.00**
kroC100	25	**4964**	5346	5048.60	**4964**	5103	4988.70	**4964**	**4982**	**4966.50**
kroD100	25	**4762**	5062	**4855.60**	4787	**5014**	4905.60	**4762**	5029	4874.00
kroE100	25	**3905**	3977	3918.70	**3905**	3984	3916.80	**3905**	**3935**	**3914.50**
lin105	26	**2606**	2751	2632.90	**2606**	**2680**	**2613.40**	**2606**	**2680**	**2613.40**
lin318	79	9133	10402	9654.90	8936	**9433**	**9175.60**	**8901**	10136	9463.10
p654	163	13671	26915	18722.80	**7348**	**8162**	**7834.00**	**7348**	8173	8030.50
pa561	140	609	710	639.10	**532**	570	**547.70**	536	**569**	552.20
pcb442	110	11994	12664	12206.70	11254	**11861**	**11558.30**	**11247**	12136	11734.50
pr107	26	**8443**	9311	8590.80	**8443**	**8449**	**8444.50**	**8443**	8729	8472.20
pr124	31	14952	**14952**	14952.00	**14640**	**14952**	**14920.80**	14952	**14952**	14952.00
pr136	34	21174	**23108**	**22186.60**	**21116**	26162	22862.80	21289	24118	22821.80
pr144	36	**14538**	**16119**	**14710.50**	**14538**	17196	15312.90	**14538**	16815	15327.30
pr152	38	23373	**24412**	**23895.70**	23373	24456	23988.40	**23195**	25027	24069.60
pr226	56	**20033**	27165	24288.50	**20033**	**25416**	**22968.40**	21814	26450	24669.20
pr264	66	10188	15034	11532.90	**9472**	**10512**	**9924.00**	9672	11073	10290.50
pr299	74	11877	13258	12554.00	11513	**11835**	**11653.60**	**11449**	12792	12164.50
pr439	109	22438	24316	23516.90	**20736**	**21704**	**21175.60**	20984	22266	21468.20
pr76	19	**23450**	**23450**	**23450.00**	**23450**	**23450**	**23450.00**	**23450**	**23450**	**23450.00**
rat195	48	592	618	603.30	**557**	594	**579.20**	568	**586**	580.40
rat575	143	1885	2156	1994.10	1612	**1696**	**1661.80**	**1599**	1710	1665.90
rat783	195	3939	6776	4466.50	**2343**	**2572**	**2478.90**	2533	2977	2690.00
rat99	24	**284**	291	285.70	**284**	**287**	285.20	**284**	**287**	**284.50**
rd100	25	**1438**	1613	1517.80	**1438**	**1521**	1472.30	**1438**	1545	**1462.70**
rd400	100	3750	4084	3931.00	**3383**	**3681**	**3526.30**	3458	3733	3597.00
si175	43	4968	5246	5107.40	4878	**4932**	**4901.40**	**4875**	5034	4942.00
si535	133	11777	13804	12116.70	11271	**11820**	**11460.40**	**11231**	12280	11543.60
st70	17	**120**	**125**	123.50	**120**	**125**	123.50	**120**	**125**	**123.40**
swiss42	10	**192**	**192**	**192.00**	**192**	**192**	**192.00**	**192**	**192**	**192.00**
ts225	56	**28828**	**28828**	**28828.00**	**28828**	**28828**	**28828.00**	**28828**	**28828**	**28828.00**
tsp225	56	957	1049	997.40	**923**	**970**	**952.50**	939	1009	970.10
u159	39	9176	9629	9354.30	**8983**	9623	9262.30	9085	**9332**	**9198.70**
u574	143	11188	12850	11794.80	**8384**	**9183**	**8747.50**	8711	9303	9016.40
u724	181	19865	34745	22750.10	**11293**	**12282**	**11802.20**	12423	14590	13555.80
ulysses16	4	**935**	**935**	**935.00**	**935**	**935**	**935.00**	**935**	**935**	**935.00**
ulysses22	5	**747**	**747**	**747.00**	**747**	**747**	**747.00**	**747**	**747**	**747.00**

The second column (k) reports the number of cities (including the home city) that the salesman has to visit. The columns *Best*, *Worst* & *Average* under an approach reports the best, worst and average solution quality obtained over ten independent

Table 5 Results of various approaches on each instance under medium scenario with SR termination criteria

Instance	k	GVNS			HH_RAND			HH_GREEDY		
		Best	Worst	Average	Best	Worst	Average	Best	Worst	Average
a280	140	1446	1575	1508.80	**1358**	**1451**	**1397.70**	1373	1462	1409.90
ali535	267	107094	256356	132364.09	**52272**	75825	56897.30	53331	**65386**	**56496.10**
att48	24	**3603**	**3603**	**3603.00**	**3603**	**3603**	**3603.00**	**3603**	**3603**	**3603.00**
att532	266	17045	36396	20907.50	10766	12388	**11172.60**	**10654**	**12112**	11272.30
bayg29	14	**626**	**626**	**626.00**	**626**	**626**	**626.00**	**626**	**626**	**626.00**
bays29	14	**733**	**733**	**733.00**	**733**	**733**	**733.00**	**733**	**733**	**733.00**
berlin52	26	**1874**	**1928**	1883.70	**1874**	**1928**	**1879.40**	**1874**	1992	1895.80
bier127	63	27519	28758	28119.00	26377	**27617**	**27030.10**	**26145**	28091	27127.20
brazil58	29	8001	8170	8060.60	7993	**8077**	**8028.10**	**7978**	8132	**8028.10**
brg180	90	1310	1500	1432.00	**1110**	**1170**	**1135.00**	**1110**	1190	1158.00
burma14	7	**1272**	**1272**	**1272.00**	**1272**	**1272**	**1272.00**	**1272**	**1272**	**1272.00**
ch130	65	2594	2915	2772.30	**2408**	**2570**	**2506.60**	2463	2646	2548.50
ch150	75	2927	3158	3067.30	**2793**	**2929**	**2883.30**	2821	3039	2917.30
d198	99	7297	7582	7400.50	7080	**7155**	**7133.00**	**7058**	7270	7181.40
d493	246	21354	35272	24460.20	15041	**15615**	15268.40	**14651**	15640	**15221.00**
d657	328	42682	137294	55356.60	**34359**	**53018**	**38065.90**	37322	70533	42231.90
dantzig42	21	**260**	**260**	**260.00**	**260**	**260**	**260.00**	**260**	**260**	**260.00**
eil101	50	234	246	241.40	228	**237**	**232.30**	**227**	243	234.60
eil51	25	**175**	185	180.00	**175**	187	180.40	**175**	**184**	**178.80**
eil76	38	218	227	223.50	219	229	222.80	**217**	**226**	**221.60**
fl417	208	9821	21810	11752.50	6908	**7551**	7231.90	**6662**	7719	**7202.70**
fri26	13	**414**	**414**	**414.00**	**414**	**414**	**414.00**	**414**	**414**	**414.00**
gil262	131	1163	1260	1222.80	1044	**1134**	**1106.00**	**1042**	1176	1125.40
gr120	60	2736	3048	2892.10	**2690**	2889	**2767.10**	2694	**2884**	2782.30

Table 5 (continued)

Instance	*k*	GVNS			HH_RAND			HH_GREEDY		
		Best	Worst	Average	Best	Worst	Average	Best	Worst	Average
gr137	68	29920	32460	31354.10	**29491**	**30557**	**29779.10**	29541	32365	30564.10
gr17	8	**517**	**517**	**517.00**	**517**	**517**	**517.00**	**517**	**517**	**517.00**
gr202	101	14509	15476	14988.80	14313	**14944**	**14596.30**	**14221**	15056	14656.70
gr21	10	**918**	**918**	**918.00**	**918**	**918**	**918.00**	**918**	**918**	**918.00**
gr229	114	45181	48567	47082.30	**41518**	**43585**	**42233.50**	42403	45229	43489.80
gr24	12	**504**	**504**	**504.00**	**504**	**504**	**504.00**	**504**	**504**	**504.00**
gr431	215	66497	92715	72582.30	**38134**	**41630**	**39686.70**	38469	43318	40311.60
gr48	24	**1819**	1836	1820.70	**1819**	1836	1822.40	**1819**	**1819**	**1819.00**
gr666	333	221489	810842	298109.91	**128728**	**190163**	**148593.80**	149752	302646	184097.20
gr96	48	20765	22069	21437.60	**20688**	**20881**	**20766.90**	20733	21617	21083.80
hk48	24	**4701**	4759	4712.30	**4701**	4759	4710.20	**4701**	**4735**	**4707.40**
kroA100	50	9572	10280	9885.00	**9184**	**9736**	**9369.90**	**9184**	10176	9666.80
kroA150	75	12866	13704	13374.90	**11783**	**12515**	**12150.20**	11812	13350	12662.80
kroA200	100	14631	15852	15139.60	12945	**13824**	**13475.90**	**12850**	14484	13585.50
kroB100	50	10026	10974	10452.60	**9096**	**9797**	**9485.60**	9150	10286	9912.80
kroB150	75	12276	14361	13278.90	11703	**12066**	**11861.10**	**11535**	12531	12038.90
kroB200	100	14297	15955	15271.80	**13080**	**14369**	**13775.50**	13434	14808	14284.50
kroC100	50	9668	10288	9986.00	**9457**	**10027**	**9702.40**	9709	10216	9884.10
kroD100	50	8962	9582	9223.20	**8719**	9134	**8870.20**	**8719**	**9103**	8884.60
kroE100	50	9283	10093	9804.20	9130	**9452**	**9259.30**	**9102**	9724	9452.40
lin105	52	5880	5954	5899.80	**5848**	**5883**	**5863.10**	**5848**	5954	5885.20
lin318	159	20453	25916	22657.50	**18600**	20346	**19462.40**	19114	**20322**	19786.80
p654	327	33482	256641	59420.50	**22241**	**73518**	**30505.20**	25556	94437	34445.80

Table 5 (continued)

Instance	k	GVNS			HH_RAND			HH_GREEDY		
		Best	Worst	Average	Best	Worst	Average	Best	Worst	Average
pa561	280	1743	3178	1963.30	1285	**1407**	**1326.50**	**1232**	1501	1348.80
pcb442	221	31481	44188	34710.10	25214	**26075**	**25609.40**	**25149**	26379	25663.90
pr107	53	29652	31418	30722.90	18052	**29261**	**21452.50**	**18028**	29927	27145.80
pr124	62	**22998**	25088	23584.70	**22998**	**22998**	**22998.00**	**22998**	23321	23037.20
pr136	68	48496	52908	50164.50	47147	**47981**	**47663.60**	**47016**	48757	47875.20
pr144	72	29464	32664	31845.20	**28402**	**32321**	**30364.40**	29297	32674	30776.80
pr152	76	37881	47105	**41807.80**	37928	**46495**	41887.00	**36637**	47630	44106.90
pr226	113	39638	42326	40438.60	**38941**	**40461**	**39597.80**	39496	41838	40768.80
pr264	132	32000	36703	34025.30	**27898**	**29623**	**29189.20**	28699	32174	30294.50
pr299	149	26696	29615	27820.20	**23694**	**24100**	**23964.80**	23855	25679	24668.90
pr439	219	56023	91406	63483.30	**40440**	45601	**42148.30**	41011	**44208**	42267.10
pr76	38	41254	42786	41816.10	41258	**41976**	**41516.30**	**41248**	42472	41849.90
rat195	97	1185	1288	1240.80	**1159**	**1184**	**1171.80**	1161	1213	1185.50
rat575	287	5232	10894	5963.60	**3672**	**4242**	**3886.20**	3943	4586	4148.70
rat783	391	8682	41474	12405.30	**7511**	**18160**	**8928.20**	8091	24225	10010.50
rat99	49	577	599	588.10	**574**	**589**	**580.80**	575	590	582.70
rd100	50	3192	3371	3268.60	**3168**	**3236**	**3200.40**	3192	3322	3234.20
rd400	200	8839	12104	9610.70	7556	**7819**	**7683.20**	**7487**	7943	7731.70
si175	87	10500	11042	10750.40	**10188**	**10487**	**10320.20**	10244	10522	10404.10
si535	267	23554	32045	24735.10	**23039**	**26630**	**23525.70**	23101	27112	23609.40
st70	35	**260**	**278**	267.20	**260**	279	265.90	**260**	280	**263.00**
swiss42	21	**458**	**458**	**458.00**	**458**	**458**	**458.00**	**458**	**458**	**458.00**
ts225	112	**56828**	58257	57589.00	**56828**	**57656**	**57229.90**	**56828**	**57656**	57312.70

Table 5 (continued)

Instance	k	GVNS			HH_RAND			HH_GREEDY		
		Best	Worst	Average	Best	Worst	Average	Best	Worst	Average
tsp225	112	1867	2044	1943.70	**1766**	**1849**	**1820.00**	1783	1877	1835.40
u159	79	18608	19524	19023.80	**18401**	**18762**	**18550.10**	**18401**	18939	18703.70
u574	287	28540	75322	35589.70	**20544**	**25252**	**22068.90**	22698	28472	24273.60
u724	362	40096	174467	57592.40	**36221**	**85596**	**42480.70**	40110	104609	47298.50
ulysses16	8	**1685**	**1685**	**1685.00**	**1685**	**1685**	**1685.00**	**1685**	**1685**	**1685.00**
ulysses22	11	**1902**	1903	1902.20	**1902**	**1902**	**1902.00**	**1902**	**1902**	**1902.00**

Table 6 Results of various approaches on each instance under large scenario with SR termination criteria

Instance	k	GVNS			HH_RAND			HH_GREEDY		
		Best	Worst	Average	Best	Worst	Average	Best	Worst	Average
a280	210	2268	2749	2386.60	2102	**2201**	**2139.90**	**2094**	2254	2167.70
ali535	401	196679	717736	270305.09	**118366**	**217688**	**144713.59**	136714	239308	151024.00
att48	36	**6563**	6656	**6583.10**	**6563**	6693	6586.30	**6563**	**6636**	6584.80
att532	399	31837	94931	39125.70	**22442**	**29921**	**24166.70**	22556	36159	25062.20
bayg29	21	**999**	**999**	**999.00**	**999**	**999**	**999.00**	**999**	**999**	**999.00**
bays29	21	**1194**	**1204**	**1196.40**	**1194**	**1204**	1197.80	**1194**	**1204**	1196.80
berlin52	39	**4174**	4436	4350.10	4251	4458	4342.80	**4174**	**4409**	**4289.90**
bier127	95	**51113**	57192	53949.80	51565	**53103**	**52269.80**	51593	54638	52822.80
brazil58	43	**11614**	11964	11785.40	**11614**	**11619**	**11616.10**	**11614**	11699	11623.30
brg180	135	2070	2310	2186.00	1650	**1750**	**1695.00**	**1620**	1820	1721.00
burma14	10	**1642**	**1642**	**1642.00**	**1642**	**1642**	**1642.00**	**1642**	**1642**	**1642.00**
ch130	97	4186	4635	4418.50	4012	**4212**	**4112.10**	**4003**	4421	4210.70
ch150	112	4814	5288	5001.40	4565	4843	4677.90	**4503**	**4842**	**4676.20**
d198	148	**9622**	11450	10300.10	9627	**10115**	**9861.70**	9758	10358	9971.00
d493	369	31965	78152	38732.00	**24533**	**31294**	**26597.20**	25635	31763	26990.10
d657	492	61969	297626	87099.60	55571	**131286**	**64648.20**	**54983**	184501	71055.90
dantzig42	31	**427**	**449**	435.10	**427**	459	432.40	**427**	455	**430.70**
eil101	75	403	430	418.50	**396**	**409**	**403.70**	**396**	418	406.90
eil51	38	290	300	294.80	290	**299**	295.20	**289**	300	**292.90**
eil76	57	348	366	358.70	341	**356**	**347.90**	**339**	363	350.80
fl417	312	12024	41153	16353.30	8816	13523	**9714.70**	**8810**	**13431**	10007.60
fri26	19	**601**	**601**	**601.00**	**601**	**601**	**601.00**	**601**	**601**	**601.00**
gil262	196	1835	2065	1960.60	**1695**	**1800**	**1748.00**	1717	1829	1773.00
gr120	90	4688	5027	4828.60	**4424**	**4622**	**4530.40**	4525	4770	4675.70

Table 6 (continued)

Instance	k	GVNS			HH_RAND			HH_GREEDY		
		Best	Worst	Average	Best	Worst	Average	Best	Worst	Average
gr137	102	44356	52121	48566.10	**44220**	**45256**	**44751.00**	44382	47195	45816.80
gr17	12	**951**	**951**	**951.00**	**951**	**951**	**951.00**	**951**	**951**	**951.00**
gr202	151	22651	24576	23681.50	**22119**	**23100**	**22712.60**	22530	23388	22979.00
gr21	15	**1501**	**1501**	**1501.00**	**1501**	**1501**	**1501.00**	**1501**	**1501**	**1501.00**
gr229	171	73882	83324	77747.30	**69201**	**73782**	**70951.70**	70598	74436	72796.40
gr24	18	**844**	**844**	**844.00**	**844**	**844**	**844.00**	**844**	**844**	**844.00**
gr431	323	123736	243300	142645.80	**87161**	**102599**	**90941.30**	87748	103074	90947.30
gr48	36	3113	3261	3178.60	3113	**3231**	3159.40	**3104**	3380	**3144.90**
gr666	499	344171	1734123	498885.31	**264813**	**722902**	**335465.19**	275516	954405	365561.41
gr96	72	32425	35322	33450.30	**31437**	**32452**	**31798.20**	31706	32746	32017.80
hk48	36	7311	7677	7422.60	7326	7559	7442.30	**7278**	**7386**	**7318.70**
kroA100	75	15418	17074	16558.20	**14492**	**15586**	**14917.90**	14775	16542	15444.60
kroA150	112	20019	21807	20799.70	**18629**	**19592**	**19017.80**	19176	20717	19679.40
kroA200	150	22354	24847	23937.70	21056	**22196**	**21563.50**	**20723**	22617	21834.10
kroB100	75	15857	17040	16456.70	**14831**	**15416**	**15029.20**	14880	16045	15470.00
kroB150	112	19312	21896	20549.40	18103	**19018**	**18427.50**	**17729**	19749	18709.50
kroB200	150	22895	24442	23885.60	21241	**22346**	**21765.70**	**21043**	22965	21833.00
kroC100	75	15144	17257	15990.40	**14412**	**15052**	**14711.40**	14581	16423	15468.80
kroD100	75	15053	16798	15795.30	**14382**	**15088**	**14803.50**	14454	15968	15240.70
kroE100	75	15250	17118	16202.60	**14776**	**15605**	**15123.00**	15060	16272	15609.70
lin105	78	9405	9796	9650.70	**9058**	**9499**	**9206.30**	9161	9986	9406.10
lin318	238	34430	43832	36058.30	31370	**34198**	**32202.50**	**30682**	34447	32321.50
p654	490	41141	578099	98659.90	**30845**	**180175**	**50757.90**	34879	302242	64742.90

Table 6 (continued)

Instance	k	GVNS			HH_RAND			HH_GREEDY		
		Best	Worst	Average	Best	Worst	Average	Best	Worst	Average
pa561	420	2944	8298	3535.90	**2205**	**3141**	**2405.70**	2288	3231	2478.10
pcb442	331	50388	92044	55843.70	40037	45301	42130.40	**40012**	**43452**	**41496.20**
pr107	80	39574	40944	40270.60	36843	**38060**	**37386.20**	**36468**	39045	37798.40
pr124	93	40230	43145	41802.20	39381	**39599**	**39533.50**	**39237**	41298	39974.90
pr136	102	73061	80892	77248.70	**69939**	**72538**	**71350.90**	71737	74652	73310.70
pr144	108	45204	54047	48876.10	**42721**	**46123**	**44607.20**	44266	49345	46872.50
pr152	114	60924	69307	64608.10	**58097**	**62759**	**60589.30**	60829	64174	62321.60
pr226	169	51519	64822	57824.90	**49198**	61479	**53353.00**	49732	**61193**	54024.20
pr264	198	44120	68691	49413.40	**39130**	**45863**	**41945.90**	41061	47907	44298.30
pr299	224	42458	51252	44269.00	**36657**	**39024**	**37847.50**	36977	41521	38599.80
pr439	329	97027	194132	114449.20	**69371**	**86472**	**75268.70**	72548	90071	76568.90
pr76	57	66833	70073	68378.40	64990	**66566**	**65593.50**	**64694**	66988	65642.70
rat195	146	1852	1981	1917.90	**1782**	**1856**	**1810.90**	1783	1888	1827.00
rat575	431	7984	29090	10308.30	**6896**	**11120**	**7451.90**	7176	14586	8085.90
rat783	587	11660	77251	18571.30	**10561**	**39014**	**13791.10**	11371	52236	15729.80
rat99	74	912	952	940.80	**876**	**913**	**894.70**	888	948	918.90
rd100	75	5197	5836	5552.50	**5096**	**5577**	**5287.30**	5226	5618	5488.10
rd400	300	14507	22552	15634.30	**11796**	**13400**	**12277.90**	11973	14055	12521.10
si175	131	15998	17139	16352.30	**15708**	**15978**	**15871.30**	15720	16068	15938.10
si535	401	36651	56842	39029.00	**35417**	**45057**	**36630.60**	35538	45353	36684.00
st70	52	444	468	456.00	**428**	**451**	**440.20**	**428**	465	446.60
swiss42	31	**760**	782	770.70	**760**	**774**	**762.50**	**760**	788	764.10
ts225	168	86484	93538	89286.20	**85656**	**88741**	**87418.20**	86535	89828	88558.40

Table 6 (continued)

Instance	k	GVNS			HH_RAND			HH_GREEDY		
		Best	Worst	Average	Best	Worst	Average	Best	Worst	Average
tsp225	168	2987	3236	3070.30	**2780**	2936	**2875.80**	2813	**2932**	2877.80
u159	119	27890	31140	29311.50	**27621**	**29816**	**28556.40**	27790	29891	29101.20
u574	430	44488	180915	60303.20	**36238**	**62734**	**40817.20**	39009	85203	45108.60
u724	543	55269	340356	86381.70	**52148**	**166699**	**65839.60**	52708	212251	71306.40
ulysses16	12	**3183**	**3184**	3183.70	**3183**	**3184**	3183.40	**3183**	**3184**	**3183.30**
ulysses22	16	**2941**	**2942**	2941.70	**2941**	**2942**	2941.90	**2941**	**2942**	**2941.20**

Table 7 Results of various approaches on each instance under small scenario with LR termination criteria

Instance	k	GVNS			HH_RAND			HH_GREEDY		
		Best	Worst	Average	Best	Worst	Average	Best	Worst	Average
a280	70	687	742	717.50	**670**	**713**	**698.40**	683	736	711.10
ali535	133	18832	28075	22382.20	**12602**	**13612**	**13220.30**	12980	15030	13486.80
att48	12	**1925**	**1925**	**1925.00**	**1925**	**1925**	**1925.00**	**1925**	**1925**	**1925.00**
att532	133	4596	4873	4699.50	**3980**	**4258**	**4132.90**	4093	4392	4236.90
bayg29	7	**332**	**332**	**332.00**	**332**	**332**	**332.00**	**332**	**332**	**332.00**
bays29	7	**400**	**400**	**400.00**	**400**	**400**	**400.00**	**400**	**400**	**400.00**
berlin52	13	**679**	**679**	**679.00**	**679**	**679**	**679.00**	**679**	**679**	**679.00**
bier127	31	**10619**	11029	10804.60	10687	11014	10792.00	10692	**10813**	**10745.80**
brazil58	14	**4965**	5030	4983.70	**4965**	**4965**	**4965.00**	**4965**	**4965**	**4965.00**
brg180	45	650	710	686.00	540	580	554.00	**530**	**560**	**546.00**
burma14	3	**359**	**359**	**359.00**	**359**	**359**	**359.00**	**359**	**359**	**359.00**
ch130	32	1149	1351	1261.10	**1130**	**1291**	1226.30	**1130**	1296	**1219.70**
ch150	37	1318	1350	1330.20	**1276**	**1336**	1316.90	**1276**	1386	**1310.50**
d198	49	5080	5244	5169.10	**5028**	**5120**	**5066.40**	5037	5130	5075.40
d493	123	10032	10868	10404.90	9411	**9576**	**9489.60**	**9394**	9767	9567.40
d657	164	14029	14919	14424.60	**12299**	**12751**	**12554.10**	12562	13607	12942.80
dantzig42	10	**145**	**145**	**145.00**	**145**	**145**	**145.00**	**145**	**145**	**145.00**
eil101	25	**107**	**108**	**107.30**	**107**	109	107.60	**107**	109	**107.30**
eil51	12	**82**	**82**	**82.00**	**82**	**82**	**82.00**	**82**	**82**	**82.00**
eil76	19	**102**	**102**	**102.00**	**102**	**102**	**102.00**	**102**	**102**	**102.00**
fl417	104	2285	2493	2324.90	**2257**	**2269**	**2259.40**	2258	2283	2262.20
fri26	6	**243**	**243**	**243.00**	**243**	**243**	**243.00**	**243**	**243**	**243.00**
gil262	65	574	641	606.70	**545**	**570**	**561.00**	553	579	564.70
gr120	30	**1308**	1362	1314.20	**1308**	**1318**	**1312.60**	**1308**	1386	1323.40
gr137	34	17802	18065	17996.50	**17399**	**17999**	**17780.20**	17445	18065	17813.20
gr17	4	**234**	**234**	**234.00**	**234**	**234**	**234.00**	**234**	**234**	**234.00**
gr202	50	8191	8564	8422.40	**8142**	**8404**	8301.40	**8142**	8428	**8283.50**
gr21	5	**324**	**324**	**324.00**	**324**	**324**	**324.00**	**324**	**324**	**324.00**
gr229	57	20252	24811	21926.40	**18555**	19805	**19102.80**	18970	**19600**	19195.30
gr24	6	**264**	**264**	**264.00**	**264**	**264**	**264.00**	**264**	**264**	**264.00**
gr431	107	15556	16227	15960.10	**14857**	**15612**	**15374.30**	15350	16221	15819.40
gr48	12	**874**	**874**	**874.00**	**874**	**874**	**874.00**	**874**	**874**	**874.00**
gr666	166	29083	32399	31037.20	**27358**	**28430**	**28045.50**	27820	29347	28660.90
gr96	24	10465	**10465**	**10465.00**	**10460**	10561	10474.00	**10460**	10786	10529.20
hk48	12	**2827**	**2827**	**2827.00**	**2827**	**2827**	**2827.00**	**2827**	**2827**	**2827.00**
kroA100	25	4998	5104	5023.10	**4970**	**5061**	**5011.80**	4998	5104	5017.20
kroA150	37	5725	6182	5936.10	**5690**	6454	5938.60	**5690**	**6154**	**5826.70**
kroA200	50	6438	7155	6780.70	**6202**	**6496**	**6358.00**	6421	6663	6503.10
kroB100	25	4353	4788	4588.80	**4305**	**4684**	**4473.70**	4501	4788	4574.50
kroB150	37	6410	6854	6588.00	6071	6938	6468.20	**5812**	**6781**	**6426.20**

Table 7 (continued)

Instance	k	GVNS			HH_RAND			HH_GREEDY		
		Best	Worst	Average	Best	Worst	Average	Best	Worst	Average
kroB200	50	6414	7543	7051.00	**6370**	**7034**	**6711.20**	6440	7219	6811.80
kroC100	25	**4964**	5346	5048.60	**4964**	**5103**	**4988.70**	**4964**	5227	4991.50
kroD100	25	**4762**	5062	4855.60	4787	5014	4905.60	**4762**	**4980**	**4843.70**
kroE100	25	**3905**	3977	3918.70	**3905**	3984	3916.80	**3905**	**3935**	**3911.00**
lin105	26	**2606**	2751	2632.90	**2606**	2680	2613.40	**2606**	**2606**	**2606.00**
lin318	79	9108	10190	9624.10	**8912**	**9626**	**9162.50**	9219	9813	9429.30
p654	163	8152	8328	8255.70	7348	**8128**	**7439.20**	**7341**	8173	7633.80
pa561	140	575	658	607.70	**523**	**563**	**540.20**	526	568	547.40
pcb442	110	11628	12267	12000.80	**11099**	**11716**	**11457.00**	11539	12054	11782.70
pr107	26	**8443**	9311	8590.80	**8443**	**8449**	**8444.50**	**8443**	8729	8472.20
pr124	31	14952	**14952**	14952.00	**14640**	**14952**	**14920.80**	14952	**14952**	14952.00
pr136	34	21174	**23108**	**22186.60**	**21116**	26162	22862.80	21174	24564	22398.40
pr144	36	14538	**16119**	**14710.50**	14538	17196	15410.30	**14327**	16363	15483.40
pr152	38	**23373**	**24412**	**23895.70**	**23373**	24544	24001.70	23700	25078	24298.30
pr226	56	21202	27255	24446.10	**20033**	26595	**23665.00**	20125	**25902**	24008.30
pr264	66	10864	15034	11990.60	9432	**10432**	9931.20	**9232**	11032	**9819.20**
pr299	74	11773	13272	12205.60	**11392**	**11916**	**11611.50**	11675	12622	12055.20
pr439	109	22243	23926	22911.30	**20874**	**21267**	**21064.90**	20987	21728	21355.50
pr76	19	**23450**	**23450**	**23450.00**	**23450**	**23450**	**23450.00**	**23450**	**23450**	**23450.00**
rat195	48	593	618	603.50	**557**	**586**	580.80	559	590	**573.90**
rat575	143	1694	1817	1758.80	1591	**1663**	**1622.30**	**1589**	1691	1640.80
rat783	195	2453	2746	2577.60	**2192**	**2274**	**2222.90**	**2192**	2283	2234.60
rat99	24	**284**	291	285.70	**284**	**287**	285.20	**284**	**287**	**284.50**
rd100	25	**1438**	1613	1517.80	**1438**	**1521**	**1472.30**	**1438**	1545	1488.30
rd400	100	3762	4013	3884.90	**3362**	**3506**	**3439.90**	3486	3769	3596.80
si175	43	4968	5246	5107.40	**4881**	**4947**	**4906.00**	4906	5010	4945.30
si535	133	11688	13780	12051.00	**11208**	**11815**	**11374.40**	11347	12280	11575.00
st70	17	**120**	**125**	123.50	**120**	**125**	123.50	**120**	**125**	**123.40**
swiss42	10	**192**	**192**	**192.00**	**192**	**192**	**192.00**	**192**	**192**	**192.00**
ts225	56	**28828**	**28828**	**28828.00**	**28828**	**28828**	**28828.00**	**28828**	**28828**	**28828.00**
tsp225	56	977	1028	999.10	**915**	**964**	942.10	**915**	973	**939.70**
u159	39	9176	9629	9354.30	**8983**	9623	9262.30	**8983**	**9412**	**9188.60**
u574	143	9146	9602	9364.20	**8310**	**8725**	**8520.20**	8453	9194	8870.50
u724	181	11737	13354	12395.80	10176	**10733**	**10362.50**	**10071**	11162	10578.20
ulysses16	4	**935**	**935**	**935.00**	**935**	**935**	**935.00**	**935**	**935**	**935.00**
ulysses22	5	**747**	**747**	**747.00**	**747**	**747**	**747.00**	**747**	**747**	**747.00**

runs respectively by that approach. The best values are reported in bold font for ease of identification. Please note that we have also reported the worst solution obtained by various approaches over ten runs here. The knowledge about the worst solution

Table 8 Results of various approaches on each instance under medium scenario with LR termination criteria

Instance	k	GVNS			HH_RAND			HH_GREEDY		
		Best	Worst	Average	Best	Worst	Average	Best	Worst	Average
a280	140	1376	1510	1443.60	**1314**	**1417**	**1363.90**	1362	1436	1400.10
ali535	267	59255	107878	72798.10	46457	53987	**49143.40**	**46195**	**52936**	49680.30
att48	24	**3603**	**3603**	**3603.00**	**3603**	**3603**	**3603.00**	**3603**	**3603**	**3603.00**
att532	266	11285	13411	12027.20	**9991**	**10800**	**10396.80**	10285	10924	10466.30
bayg29	14	**626**	**626**	**626.00**	**626**	**626**	**626.00**	**626**	**626**	**626.00**
bays29	14	**733**	**733**	**733.00**	**733**	**733**	**733.00**	**733**	**733**	**733.00**
berlin52	26	**1874**	1928	1883.70	**1874**	1928	1879.40	**1874**	**1917**	**1878.30**
bier127	63	27404	28361	27942.20	**26062**	27335	26841.20	26102	**27329**	**26745.50**
brazil58	29	8001	8170	8060.60	7993	8077	8028.10	**7978**	**8065**	**8005.80**
brg180	90	1370	1470	1412.00	**1080**	**1120**	**1103.00**	1110	1160	1130.00
burma14	7	**1272**	**1272**	**1272.00**	**1272**	**1272**	**1272.00**	**1272**	**1272**	**1272.00**
ch130	65	2483	2920	2742.70	**2423**	**2564**	**2512.60**	2544	2657	2588.20
ch150	75	2977	3158	3077.00	**2761**	**2948**	**2851.20**	2856	2968	2914.30
d198	99	7270	7517	7396.70	**7073**	**7198**	**7124.30**	7126	7318	7209.60
d493	246	15997	17188	16514.90	**14223**	**14803**	**14568.80**	14356	15119	14674.80
d657	328	32740	54078	37002.30	24462	27495	**25991.30**	**24424**	**26937**	26093.40
dantzig42	21	**260**	**260**	**260.00**	**260**	**260**	**260.00**	**260**	**260**	**260.00**
eil101	50	240	245	242.20	228	**238**	234.00	**227**	239	**232.80**
eil51	25	**175**	185	180.00	**175**	187	180.40	**175**	**184**	**178.80**
eil76	38	218	227	223.50	219	229	222.80	**216**	**226**	**218.70**
fl417	208	7404	8593	7848.30	**6404**	**6955**	**6597.00**	6483	7157	6853.40
fri26	13	**414**	**414**	**414.00**	**414**	**414**	**414.00**	**414**	**414**	**414.00**
gil262	131	1141	1290	1224.20	1037	**1101**	**1078.20**	**1034**	1136	1095.60
gr120	60	2826	3001	2891.00	2713	2871	**2767.70**	**2710**	**2843**	2772.40
gr137	68	30456	32460	31654.50	**29363**	**30127**	**29700.00**	29673	31016	29936.60
gr17	8	**517**	**517**	**517.00**	**517**	**517**	**517.00**	**517**	**517**	**517.00**
gr202	101	14333	15415	14983.00	**14182**	**14670**	**14399.40**	14199	14958	14646.80
gr21	10	**918**	**918**	**918.00**	**918**	**918**	**918.00**	**918**	**918**	**918.00**
gr229	114	45078	49105	47002.90	**41005**	**42574**	**41642.70**	41242	44401	42704.70
gr24	12	**504**	**504**	**504.00**	**504**	**504**	**504.00**	**504**	**504**	**504.00**
gr431	215	41906	45594	43969.60	**37071**	**39334**	**37973.30**	38207	40976	38820.40
gr48	24	**1819**	1836	1820.70	**1819**	1836	1822.40	**1819**	**1819**	**1819.00**
gr666	333	140432	253851	160908.09	93994	107511	98839.90	**89448**	**107124**	**95744.70**
gr96	48	20807	22097	21431.10	**20688**	**20937**	**20756.80**	**20688**	21930	21095.20
hk48	24	**4701**	4759	4712.30	**4701**	4759	4710.20	**4701**	**4735**	**4707.40**
kroA100	50	9335	10417	10035.50	**9184**	**9525**	**9218.10**	**9184**	9949	9565.90
kroA150	75	12166	14020	13228.10	**11625**	**12379**	**12015.30**	11936	13270	12621.80
kroA200	100	14213	15717	15060.50	**12753**	**13668**	**13235.30**	13011	14069	13595.80
kroB100	50	9244	11025	10291.70	9312	**9787**	9577.40	**9096**	9980	**9537.00**
kroB150	75	12751	14361	13368.80	**11642**	**12132**	**11880.40**	11910	12412	12102.50

Table 8 (continued)

Instance	*k*	GVNS			HH_RAND			HH_GREEDY		
		Best	Worst	Average	Best	Worst	Average	Best	Worst	Average
kroB200	100	14004	15955	14971.90	**13178**	**14114**	**13704.90**	13222	14637	14030.90
kroC100	50	9668	10288	9986.00	9493	10049	9726.20	**9457**	**9923**	**9654.40**
kroD100	50	8962	9582	9223.20	**8719**	9150	8853.50	**8719**	**8969**	**8805.50**
kroE100	50	9345	10141	9781.70	9132	**9631**	**9303.70**	**9102**	9792	9386.40
lin105	52	5857	5921	5880.50	**5848**	**5919**	**5873.20**	5857	5995	5909.80
p654	327	27188	73557	34053.80	19380	**21433**	**20473.00**	**19332**	22443	20649.30
pa561	280	1429	1768	1497.50	**1211**	**1300**	**1237.30**	1224	1327	1262.40
pcb442	221	25899	27504	27015.00	24536	**25147**	**24921.30**	**24451**	26191	25208.20
pr107	53	29652	31206	30625.80	**18028**	**29798**	**23769.60**	18165	30212	28257.10
pr124	62	**22998**	24923	23890.80	**22998**	**22998**	**22998.00**	**22998**	23321	23037.20
pr136	68	49652	52064	50590.10	**46890**	**47931**	**47316.90**	47016	51128	48389.10
pr144	72	30810	33628	32062.30	**28402**	**31557**	**30063.60**	29380	32167	31107.10
pr152	76	38369	46768	42781.70	**37336**	**45171**	**40178.30**	38355	46485	41500.30
pr226	113	**38718**	40662	39691.60	38914	**39830**	**39292.40**	39231	43349	40238.70
pr264	132	31251	36187	33866.70	**27711**	**29598**	**28317.30**	28247	29903	28856.50
pr299	149	24843	27832	25756.10	**23475**	**24544**	**23849.00**	23636	25119	24498.30
pr439	219	45318	49432	46788.20	38874	**41581**	**39778.20**	**38313**	43040	40781.30
pr76	38	41254	42786	41816.10	41258	**41976**	**41516.30**	**41248**	42465	41699.30
rat195	97	1185	1288	1238.80	**1140**	**1165**	**1152.80**	1144	1208	1171.60
rat575	287	3853	4512	4052.60	**3349**	**3526**	**3444.20**	3431	3604	3489.00
rat783	391	7258	17344	8431.00	**5007**	**5580**	**5174.90**	5132	5858	5304.60
rat99	49	577	599	588.80	576	591	581.00	**574**	**589**	**579.40**
rd100	50	3192	3371	3268.60	**3168**	**3236**	**3196.80**	3190	3260	3210.30
rd400	200	7951	8542	8192.20	**7013**	**7555**	**7287.70**	7428	7799	7577.10
si175	87	10467	11042	10700.80	10265	**10436**	**10319.10**	**10208**	10556	10382.70
si535	267	23460	30386	24433.90	22940	**25599**	**23316.70**	**22896**	25926	23360.90
st70	35	**260**	278	267.20	**260**	279	265.90	**260**	**273**	**262.60**
swiss42	21	**458**	**458**	**458.00**	**458**	**458**	**458.00**	**458**	**458**	**458.00**
ts225	112	**56828**	**57656**	57436.90	**56828**	**57656**	**57229.90**	**56828**	57949	57271.30
tsp225	112	1904	2035	1964.00	**1729**	**1852**	**1788.10**	1818	1926	1868.10
u159	79	18491	19617	19031.20	18401	**18762**	**18516.70**	**18399**	18955	18580.10
u574	287	20940	26302	22489.60	**17355**	**19033**	**18214.10**	17553	19438	18249.20
u724	362	33596	71937	39033.60	**21802**	**25214**	23468.30	22138	25262	**23227.40**
ulysses16	8	**1685**	**1685**	**1685.00**	**1685**	**1685**	**1685.00**	**1685**	**1685**	**1685.00**
ulysses22	11	**1902**	1903	1902.20	**1902**	**1902**	**1902.00**	**1902**	**1902**	**1902.00**

obtained aids in ascertaining the robustness of an approach in comparison to other approaches. As can be seen from these tables, even the worst solution obtained by

Table 9 Results of various approaches on each instance under large scenario with LR termination criteria

Instance	k	GVNS			HH_RAND			HH_GREEDY		
		Best	Worst	Average	Best	Worst	Average	Best	Worst	Average
a280	210	2149	2452	2269.60	2066	**2185**	**2116.90**	**2043**	2199	2135.60
ali535	401	144825	291826	171311.59	113209	147140	120854.20	**108369**	**140219**	**118111.20**
att48	36	**6563**	6656	6583.10	**6563**	6693	6586.30	**6563**	**6605**	**6575.90**
att532	399	23148	34207	25235.50	19245	**21196**	**19729.40**	**18858**	21441	19747.60
bayg29	21	**999**	**999**	**999.00**	**999**	**999**	**999.00**	**999**	**999**	**999.00**
bays29	21	**1194**	1204	1196.40	**1194**	1204	1197.80	**1194**	**1194**	**1194.00**
berlin52	39	**4174**	4436	4350.10	4251	4458	4342.80	**4174**	**4385**	**4323.80**
bier127	95	52853	57192	54586.10	**50324**	**53100**	**51530.10**	51427	53888	52671.70
brazil58	43	**11614**	11964	11785.40	**11614**	**11619**	11615.60	**11614**	**11619**	**11614.70**
brg180	135	2030	2310	2163.00	**1600**	**1710**	**1663.00**	1650	1780	1693.00
burma14	10	**1642**	**1642**	**1642.00**	**1642**	**1642**	**1642.00**	**1642**	**1642**	**1642.00**
ch130	97	4156	4628	4412.20	**3907**	4176	**4068.50**	3956	**4166**	4089.40
ch150	112	4860	5288	4994.90	**4499**	**4705**	**4599.60**	4625	4812	4716.00
d198	148	9819	11440	10493.80	9428	**9713**	**9586.10**	**9386**	10358	9796.30
d493	369	26223	31451	27158.30	23483	**25003**	**23868.10**	**23380**	26149	24260.50
d657	492	52098	130389	61837.70	**41410**	**48611**	**42767.90**	41976	49857	43707.80
dantzig42	31	**427**	449	435.10	**427**	459	432.40	**427**	**427**	**427.00**
eil101	75	409	432	419.20	397	**408**	**402.70**	**389**	419	403.30
eil51	38	**290**	300	294.80	**290**	299	295.20	**290**	**296**	**293.30**
eil76	57	351	367	359.00	**336**	**356**	**346.10**	339	359	347.80
fl417	312	9690	16049	11174.40	8337	**8998**	**8603.10**	**8242**	11504	9027.80
fri26	19	**601**	**601**	**601.00**	**601**	**601**	**601.00**	**601**	**601**	**601.00**
gil262	196	1817	1999	1896.00	**1672**	**1759**	**1717.80**	1703	1829	1751.10
gr120	90	4585	5027	4763.90	4420	**4598**	**4528.20**	**4380**	4770	4589.00

Table 9 (continued)

Instance	k	GVNS			HH_RAND			HH_GREEDY		
		Best	Worst	Average	Best	Worst	Average	Best	Worst	Average
gr137	102	46737	50307	48199.10	44182	**45781**	**44665.50**	**43912**	47350	45554.80
gr17	12	**951**	**951**	**951.00**	**951**	**951**	**951.00**	**951**	**951**	**951.00**
gr202	151	22701	24537	23731.80	**21563**	**22761**	**22444.20**	21954	23633	22854.10
gr21	15	**1501**	**1501**	**1501.00**	**1501**	**1501**	**1501.00**	**1501**	**1501**	**1501.00**
gr229	171	72938	81767	75845.00	**67848**	**72217**	**70154.60**	68553	75220	71644.00
gr24	18	**844**	**844**	**844.00**	**844**	**844**	**844.00**	**844**	**844**	**844.00**
gr431	323	93079	107716	97093.70	81904	**87997**	**84147.10**	**81144**	89321	84736.20
gr48	36	3113	3261	3178.60	3113	**3231**	3159.40	**3104**	**3231**	**3135.00**
gr666	499	273618	763260	341435.41	**194897**	**253287**	**208981.80**	204846	259577	215325.91
gr96	72	32498	35322	33822.20	**31437**	**32452**	**31707.80**	31526	34413	32144.10
hk48	36	7311	7677	7422.60	7326	**7559**	7442.30	**7278**	7561	**7317.70**
kroA100	75	15461	16943	16280.90	14592	**14969**	**14747.30**	**14500**	16437	15267.10
kroA150	112	19192	21807	20779.40	**18210**	**19202**	**18724.50**	18917	20031	19344.40
kroA200	150	22926	25030	23673.70	20794	**21736**	**21323.40**	**20740**	22159	21542.40
kroB100	75	15846	17040	16344.20	**14744**	**15412**	**15045.60**	14799	15775	15312.50
kroB150	112	19506	21896	20560.30	17695	**18717**	**18140.40**	**17501**	19749	18705.70
kroB200	150	22964	24418	23548.40	20673	**21541**	**21191.40**	**20508**	23134	21695.30
kroC100	75	15536	17257	16162.50	14201	**15093**	**14615.30**	**14067**	15770	14922.90
kroD100	75	15211	16798	15943.00	**14171**	**15145**	**14691.00**	14288	15710	15139.00
kroE100	75	15452	17118	16102.00	**14640**	**15605**	**15154.10**	14843	15956	15217.90
lin105	78	9178	9982	9523.10	**9034**	**9310**	**9206.20**	**9034**	9408	9267.10
lin318	238	32100	37616	33558.10	**29829**	**31398**	**30479.10**	30625	32702	31419.10
p654	490	34821	198432	53572.60	**24514**	44335	**29129.30**	25671	**42799**	29650.40

Table 9 (continued)

Instance	k	GVNS			HH_RAND			HH_GREEDY		
		Best	Worst	Average	Best	Worst	Average	Best	Worst	Average
pa561	420	2361	3538	2548.40	**2031**	**2419**	**2147.40**	2067	2458	2152.20
pcb442	331	41889	46635	42991.40	**37941**	**40196**	**38976.70**	38419	40523	39116.90
pr107	80	39770	41113	40531.90	**36627**	**37745**	**37072.50**	36661	39331	38251.40
pr124	93	40471	43934	41953.00	**39174**	**39772**	**39562.40**	39528	42758	40375.60
pr136	102	75922	80389	77741.50	**69690**	**72784**	**71071.10**	69905	75594	72198.40
pr144	108	46889	51287	48861.80	**41452**	**48138**	**43983.30**	42164	49900	46620.10
pr152	114	61758	69307	64513.60	**57431**	**60424**	**59372.00**	58114	65974	62302.90
pr226	169	49552	60248	56582.70	**47516**	**54238**	**50928.10**	50315	61193	54532.20
pr264	198	43189	60377	48234.90	**39503**	**41739**	**40366.70**	40896	44796	42967.90
pr299	224	38188	43058	40204.00	**35942**	**38155**	**37030.40**	36957	39966	37949.40
pr439	329	74268	97896	78410.10	**64497**	**77142**	**68425.10**	66365	77489	70468.70
pr76	57	66186	69717	67805.60	64251	66734	65505.40	**64142**	**66117**	**64878.40**
rat195	146	1891	1983	1929.10	**1753**	**1828**	**1780.40**	1783	1844	1817.20
rat575	431	6595	11320	7231.10	**5452**	**5942**	**5591.30**	5498	6018	5610.30
rat783	587	10561	36820	13406.80	**8487**	**12529**	**9152.30**	8820	15063	9685.50
rat99	74	928	960	942.00	**861**	**904**	**884.50**	866	927	894.50
rd100	75	5221	5819	5523.20	**5094**	**5494**	**5264.50**	5192	5503	5325.20
rd400	300	12224	14484	12864.20	**11326**	12423	**11671.80**	11398	**12231**	11765.10
si175	131	16074	17136	16426.20	**15625**	**15915**	**15821.20**	15756	16016	15889.00
si535	401	35926	49376	37418.00	35141	42336	**35974.00**	**35114**	**42018**	35993.00
st70	52	444	468	456.00	**428**	**448**	**436.60**	**428**	451	437.70
swiss42	31	**760**	782	770.70	**760**	**774**	**762.50**	**760**	784	763.30
ts225	168	86898	90490	88150.60	**85656**	**87783**	**86888.40**	86070	89191	87537.20

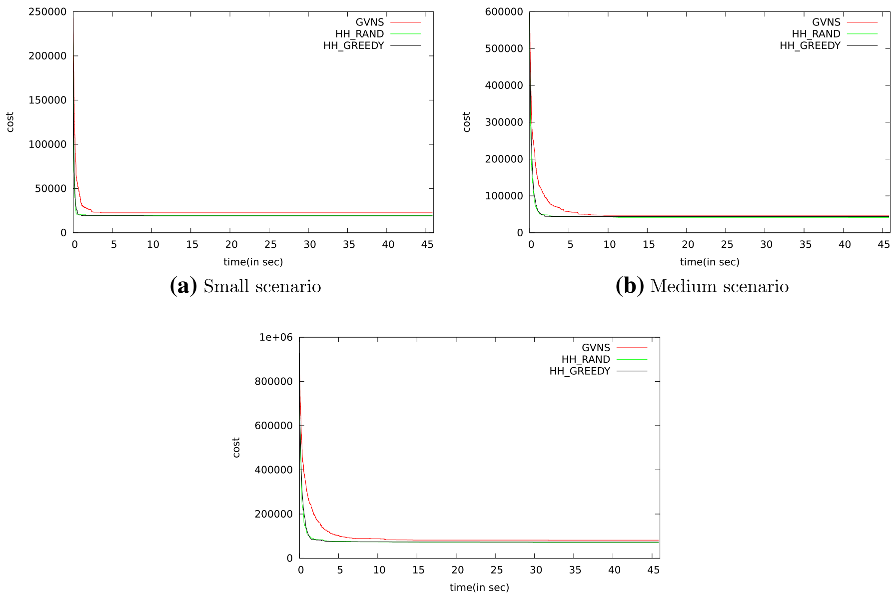
Table 9 (continued)

Instance	k	GVNS			HH_RAND			HH_GREEDY		
		Best	Worst	Average	Best	Worst	Average	Best	Worst	Average
tsp225	168	2934	3236	3067.70	**2665**	**2888**	**2773.60**	2800	2934	2855.50
u159	119	29181	30761	29968.30	**27413**	**28672**	**27927.30**	27817	30354	28991.80
u574	430	35032	64203	39064.40	28950	**31892**	30120.80	**28376**	32163	**30045.00**
u724	543	49460	155156	62157.80	**37730**	**48702**	**40120.60**	39467	54504	42298.80
ulysses16	12	**3183**	**3184**	3183.70	**3183**	**3184**	3183.40	**3183**	**3184**	**3183.30**
ulysses22	16	**2941**	**2942**	2941.70	**2941**	**2942**	2941.90	**2941**	**2942**	**2941.20**

HH_RAND is better than the worst solution obtained by other two approaches in most of the cases.

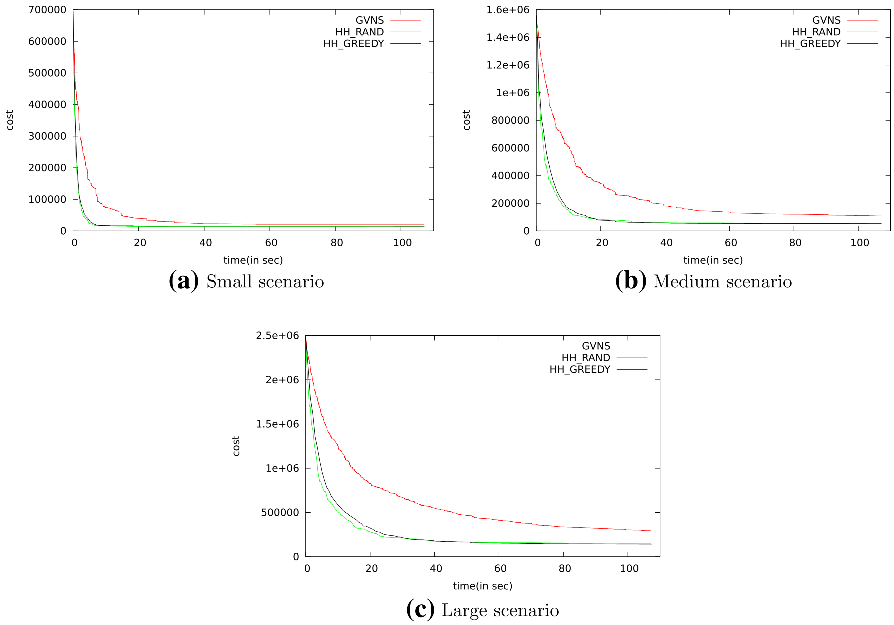
Appendix II

This appendix presents the convergence behavior of our approaches. For studying the convergence behavior, three instances of different sizes, namely *gr*229, *ali*535 and *p*654, have been considered. Figures 3, 4 and 5 plot the convergence behavior of our three approaches, viz. GVNS, HH_RAND and HH_GREEDY respectively for the instances *gr*229, *ali*535 and *p*654 under small, medium and large scenarios. The convergence graphs depict that both HH_RAND and HH_GREEDY converges faster than the GVNS. When it comes to the comparison between HH_RAND and HH_GREEDY based on their convergence behavior, there is only a minute difference in their convergence behavior except for the large scenario of *p*654, where HH_RAND clearly converges faster than HH_GREEDY.



(a) Small scenario **(b)** Medium scenario **(c)** Large scenario

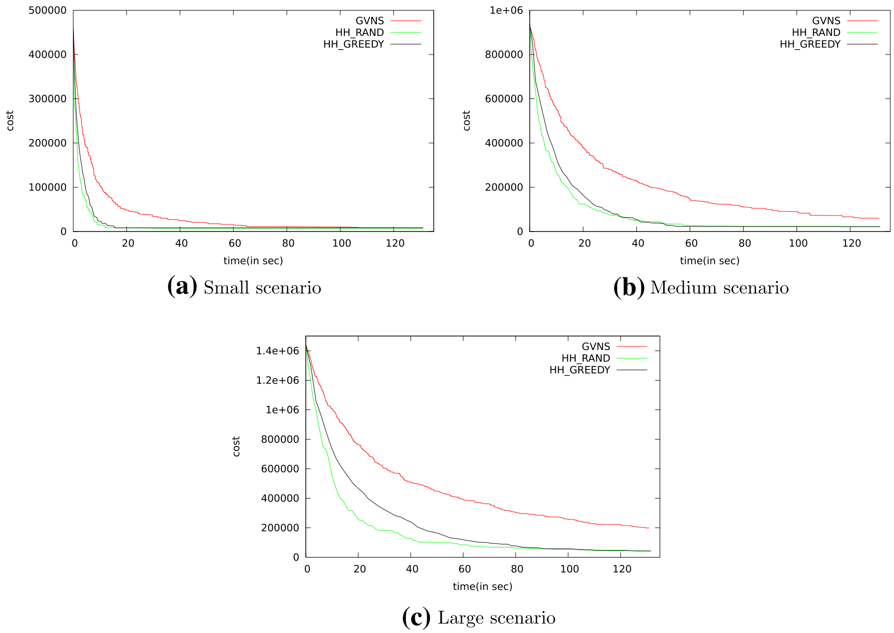
Fig. 3 Convergence behavior on instance gr229 under different scenarios



(a) Small scenario **(b)** Medium scenario

(c) Large scenario

Fig. 4 Convergence behavior on instance ali535 under different scenarios



(a) Small scenario **(b)** Medium scenario

(c) Large scenario

Fig. 5 Convergence behavior on instance p654 under different scenarios

References

1. Balas, E.: The prize collecting traveling salesman problem. Networks **19**(6), 621–636 (1989)
2. Ausiello, G., Bonifaci, V., Leonardi, S., Marchetti-Spaccamela, A.: Prize-collecting traveling salesman and related problems. Handbook Approx. Algorithms Metaheuristics **40**, 1–13 (2007)
3. Blum, A., Ravi, R., Vempala, S.: A constant-factor approximation algorithm for the k MST problem. In: Proceedings of the 28th Annual ACM Symposium on Theory of Computing, pp. 442–448. ACM (1996)
4. Garg, N.: A 3-approximation for the minimum tree spanning k vertices. In: Proceedings of 37th Annual Symposium on Foundations of Computer Science, pp. 302–309. IEEE, (1996)
5. Arora, S., Karakostas, G.: A 2+ ε approximation algorithm for the k-MST problem. In: Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 754–759. Society for Industrial and Applied Mathematics, (2000)
6. Chaudhuri, K., Godfrey, B., Rao, S., Talwar, K.: Paths, trees, and minimum latency tours. In: Proceedings of 44th Annual IEEE Symposium on Foundations of Computer Science, pp. 36–45. IEEE, (2003)
7. Garg, N.: Saving an epsilon: a 2-approximation for the k-MST problem in graphs. In: Proceedings of the 37th Annual ACM Symposium on Theory of Computing, pp. 396–402. ACM, (2005)
8. Blum, A., Chawla, S., Karger, D.R., Lane, T., Meyerson, A., Minkoff, M.: Approximation algorithms for orienteering and discounted-reward TSP. SIAM J. Comput. **37**(2), 653–670 (2007)
9. Mladenović, N., Hansen, P.: Variable neighborhood search. Comput. Oper. Res. **24**(11), 1097–1100 (1997)
10. Hansen, P., Mladenović, N., Moreno Pérez, J.A.: Variable neighbourhood search: methods and applications. Ann. Oper. Res. **175**(1), 367–407 (2010)
11. Mladenović, N., Urošević, D., Hanafi, S., Ilić, A.: A general variable neighborhood search for the one-commodity pickup-and-delivery travelling salesman problem. Eur. J. Oper. Res. **220**(1), 270–285 (2012)
12. Mladenović, N., Todosijević, R., Urošević, D.: Two level general variable neighborhood search for attractive traveling salesman problem. Comput. Oper. Res. **52**, 341–348 (2014)
13. Mladenović, N., Todosijević, R., Urošević, D.: An efficient general variable neighborhood search for large travelling salesman problem with time windows. Yugoslav J. Oper. Res. **23**(1), 19–30 (2013)
14. Todosijević, R., Mjirda, A., Mladenović, M., Hanafi, S., Gendron, B.: A general variable neighborhood search variants for the travelling salesman problem with draft limits. Optim. Lett. **11**(6), 1047–1056 (2017)
15. Venkatesh, P., Srivastava, G., Singh, A.: A general variable neighborhood search algorithm for the k-traveling salesman problem. In: Proceedings of the 8th International Conference on Advances in Computing & Communications (ICACC-2018), Procedia Computer Science, vol. 143, pp. 189–196. Elsevier, (2018)
16. Burke, E.K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Qu, R.: Hyper-heuristics: a survey of the state of the art. J. Oper. Res. Soc. **64**(12), 1695–1724 (2013)
17. Denzinger, J., Fuchs, M., Fuchs, M.: High performance ATP systems by combining several AI methods. In: Proceedings of the 15th International Joint Conference on Artifical Intelligence, vol. 1, pp. 102–107, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc (1997)
18. Fisher, H.: Probabilistic learning combinations of local job-shop scheduling rules. In: Industrial scheduling, pp. 225–251, (1963)
19. Crowston, W.B., Glover, F., Thompson, G.L., Trawick, J.D.: Probabilistic and parametric learning combinations of local job shop scheduling rules. Technical report, Research Memorandum, No. 117, GSIA, Carnegie Mellon University, Pittsburgh, (1963)
20. Cowling, P., Kendall, G., Soubeiga, E.: A hyperheuristic approach to scheduling a sales summit. In: International Conference on the Practice and Theory of Automated Timetabling, pp. 176–190. Springer, (2000)
21. Chakhlevitch, K., Cowling, P.: Hyperheuristics: recent developments. In: Adaptive and multilevel metaheuristics, pp. 3–29. Springer, (2008)
22. Drake, J.H., Özcan, E., Burke, E.K.: An improved choice function heuristic selection for cross domain heuristic search. In: International Conference on Parallel Problem Solving from Nature, pp. 307–316. Springer, (2012)

23. Kendall, G., Li, J.: Competitive travelling salesmen problem: a hyper-heuristic approach. J. Oper. Res. Soc. **64**(2), 208–216 (2013)
24. Pandiri, V., Singh, A.: A hyper-heuristic based artificial bee colony algorithm for k-interconnected multi-depot multi-traveling salesman problem. Inf. Sci. **463**, 261–281 (2018)
25. Zalilah Abd Aziz: Ant colony hyper-heuristics for travelling salesman problem. Procedia Comput. Sci. **76**, 534–538 (2015)
26. Choong, S.S., Wong, L.-P., Lim, C.P.: An artificial bee colony algorithm with a modified choice function for the traveling salesman problem. Swarm Evol. Comput. **44**, 622–635 (2019)
27. Wilcoxon, F., Katti, S.K., Wilcox, R.A.: Critical values and probability levels for the wilcoxon rank sum test and the wilcoxon signed rank test. Sel. Tab. Math. Stat. **1**, 171–259 (1970)