Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 143 (2018) 189–196

Procedia Computer Science

www.elsevier.com/locate/procedia

8th International Conference on Advances in Computing and Communication (ICACC-2018)

A general variable neighborhood search algorithm for the k-traveling salesman problem

Pandiri Venkatesh, Gaurav Srivastava, Alok Singh

School of Computer & Information Sciences, University of Hyderabad, Hyderabad - 500 046, Telangana, India.

Abstract

This paper addresses a variant of the traveling salesman problem, i.e., k-traveling salesman problem (k-TSP). Given a set of n cities and a fixed value $1 < k \leq n$, the k-TSP is to find a minimum length tour by visiting exactly k of the n cities. The k-TSP is a combination of both subset selection and permutation characteristics. In this paper, we have proposed a general variable neighborhood search algorithm for the k-TSP. A variable neighborhood descent consisting of two neighborhood structures is used as local search in our approach. To the best of the authors knowledge, this is the first metaheuristic approach for the k-TSP. Moreover, to present the computational experiments, a set of benchmark instances is generated by using the standard TSPLIB.

Keywords: k-traveling salesman problem; general variable neighborhood search; heuristic; variable neighborhood descent

1. INTRODUCTION

The k-traveling salesman problem (k-TSP) is a variant of the traveling salesman problem (TSP), and a special case of the prize collecting traveling salesman problem (PCTSP). The PCTSP was introduced by Balas *et al.*[3]. In this problem, a salesman gets a prize by visiting a city and pays a penalty by not visiting a city. The objective of the PCTSP is to minimize the total distance and total penalty while collecting the given amount of prize. If the associated penalties of all cities are zero, then the PCTSP becomes a quota traveling salesman problem (QTSP). The k-TSP is a special case of the QTSP, where all the prizes are unitary and the quota is k. k-TSP, being a special case of the PCTSP, is also NP-Hard[6, 2]. The k-TSP consists in finding a subset of k cities and arranging them in order to minimize the total distance traveled by the salesman. The important applications of this problem arise in the design of distributed networks and rural healthcare delivery.

E-mail address: venkatesh78.p@gmail.com, gauravsrignp@gmail.com, *alokcs@uohyd.ernet.in (Alok Singh).

10.1016/j.procs.2018.10.375

There are several constant factor approximation schemes known for the k-TSP[5, 7, 1, 8, 4]. The best known approximation ratio, i.e., a 2-approximation scheme for the k-TSP is given in [8]. However, there are no metaheuristic approaches available for the k-TSP to the best of the authors knowledge. Compared to other variants of TSP, the k-TSP did not get that much attention from the researchers. Thus, still it needs to be explored by the researchers from different fields. In this paper we have proposed a simple and efficient general variable neighborhood search (GVNS) algorithm for the k-TSP by incorporating the variable neighborhood descent.

The remainder of this paper is organized as follows: Section 2 formally defines the k-TSP. The proposed GVNS approach for the k-TSP is described in Section 3. Section 4 presents the computational results on the test instances. Finally, Section 5 concludes the paper by summarizing the contributions made and suggests directions for future research.

2. PROBLEM DEFINITION

Given a complete undirected graph $G = (V, E)$, where $V = \{1, 2, \ldots, n\}$ is the set of cities, $E = \{(i, j)|i, j \in V\}$ is the set of edges and a distance d_{ij} is associated with each edge $(i, j) \in E$. The objective of the k-TSP is to find a minimum length Hamiltonian cycle among all the Hamiltonian cycles over subgraphs induced by the subsets of k cities. We will denote such a subset of k vertices by V'. By introducing binary variable y_i to indicate whether city i is part of the subset ($y_i = 1$) or not ($y_i = 0$), and another binary variable x_{ij} to indicate whether edge (i, j) is part of Hamiltonian cycle ($x_{ij} = 1$) or not ($x_{ij} = 0$), the mathematical formulation for the k-TSP can be represented as follows:

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

subject to:

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

$$\sum_{(i,j) \in E} x_{ij} + \sum_{(j,i) \in E} x_{ji} = 2y_i \quad \forall i \in V, \tag{3}$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \le |S| - 1, \quad \forall S \subset V' \subset V \tag{4}$$

$$x_{ij}, y_i \in \{0, 1\} \quad \forall (i, j) \in E, i \in V. \tag{5}$$

Equation1 represents the objective function for the k-TSP and it minimizes the total distance. 2 enforces the constraint of k-TSP, i.e., visiting exactly k cities. Equation3 satisfies the constraints of indegree and outdegree of the visited cities. Equation4 represents the sub tour elimination constraint. 5 enforces the binary nature of the decision variables x_{ij} and y_i.

3. General variable neighborhood search algorithm for the k-TSP

In this section, we give a brief introduction about the variable neighborhood search, and variable neighborhood descent followed by the details of proposed GVNS approach for the k-TSP.

Variable neighborhood search (VNS) is a metaheuristic proposed by Mladenović and Hansen[10] based on the systematic search of different neighborhood structures. The VNS is composed of two phases, viz., local search and shake phases. The local search phase will do the job of exploitation (finds local optimum), whereas the shake phase will do the job of exploration (escapes from local optimum).

The pseudo code of basic VNS is given in Algorithm1. Let S be a solution and $F(S)$ be its fitness value, and let $\mathcal{N} = \{N_1, N_2, \dots, N_p\}$ be a set of p different neighborhood structures. The VNS algorithm starts by generating an initial(random) solution, and then the algorithm generates a random solution S' in the p^{th} neighborhood of S by using shake phase. The S' is used as input to the local search phase which tries to get an improved solution S'' in the p^{th} neighborhood of S'. If S'' is better than the current solution S, then the S will be replaced with S'' and the algorithm continues with the first neighborhood (i.e., $p = 1$), otherwise, the algorithm moves to the next neighborhood structure. The termination condition can be maximum time, maximum number of iterations, or some other quality measurements. The basic VNS can be perceived as a method which combines both deterministic and stochastic changes of the neighborhood.

Algorithm 1: Pseudo code of basic VNS

```
Input: Set of parameters for VNS
Output: Best solution found
S ← Generate_Initial_Solution();
while Termination condition not satisfied do
    p ← 1;
    while p ≤ p_max do
        S' ← Shake(S, N_p);
        S'' ← Local_Search(S', N_p);
        if F(S'') < F(S) then
            S ← S'';
            p ← 1;
        else
            p ← p+1;
return best solution S;
```

Algorithm 2: Pseudo code of VND

```
Input: Set of parameters for VND
Output: Best solution found
S ← Generate_Initial_Solution();
repeat
    p ← 1;
    while p ≤ p_max do
        S' ← Local_Search(S, N_p);
        if F(S') < F(S) then
            S ← S';
            p ← 1;
        else
            p ← p+1;
until there is no improvement with respect to all
      p_max neighborhoods;
return best solution S;
```

The variable neighborhood descent (VND)[10] is a simple variation of the basic VNS, which follows a deterministic way to change the neighborhood. The pseudo code of VND is given in Algorithm2. The VND starts by an initial solution S and p=1. The search starts by exploring the neighborhood $N_1(S)$ until there is no improvement possible. And then, the search continues in the neighborhood $N_2(S)$. If an improved solution is found in the neighborhood $N_2(S)$, then the VND goes back to the $N_1(S)$ to explore the neighborhood of this newly improved solution until there is no improvement possible. Otherwise, VND continues with $N_3(S)$, and so on. The final solution obtained by the VND is a local optimum with respect to all p_{max} neighborhoods. The VND is very often used as a local search method, as the chance of getting a good solution is high by using it than with a single neighborhood structure.

3.1. Proposed general variable neighborhood search (GVNS)

The general variable neighborhood search (GVNS)[9] is a variant of VNS[10] which uses the VND as the local search. The GVNS has been successfully applied to solve multiple variants of the TSP[13, 12, 11, 14]. The proposed VND explores the different neighborhood structures systematically by using the first improvement strategy[9]. The pseudo code of the proposed GVNS approach is given in Algorithm3. Notice that the proposed GVNS in Algorithm3 is similar to VNS in Algorithm1, where local search phase is replaced by VND and the termination condition is the maximum number of iterations. In fact, the GVNS can be seen as a multi start algorithm that uses shake phase and VND as its local search procedures. The components of the GVNS for k-TSP are discussed in following subsections.

3.1.1. Initial solution generation and shake phase:

The initial solution generation procedure starts by selecting a city uniformly at random and then an iterative process ensues. During each iteration a city which is not visited is selected uniformly at random and inserted into the best position in the salesman's tour. This procedure is repeated until the feasibility condition is satisfied, i.e., exactly k cities has to be visited. The initial solution S generated is used as input to the shake phase. During the shake phase, a random solution S' is generated in the p^{th} neighborhood of S.

3.1.2. Variable neighborhood descent (VND):

It is very common to use the VND as a local search in many algorithms as it systematically explores different neighborhood structures. The neighborhood structures can be designed according to the characteristics of a problem. Keeping this in mind, two neighborhood structures N_1 and N_2 are proposed for the problem under consideration. The N_1 addresses the characteristic of subset selection, whereas the N_2 addresses the characteristic of permutation.

The first neighborhood N_1 of a solution S uses the operation of exchange to form a new solution S', which means removing a city from the salesman's tour and adding an unvisited city in to the tour at best possible position. In detail, given a pair of cities $i \in V'$, $j \in V - V'$, and an edge $\{a, b\}$ of the tour, remove i from V' and add j to V' so that one edge $\{a, b\}$ becomes two edges $\{a, j\}$ and $\{j, b\}$. The second neighborhood N_2 of a solution S uses the operation of swap to form a new solution S'. That is, given a pair of cities i, j in the tour, i is moved to the place of j, and j is moved to the place of i.

The performance of VND also depends on the order in which the neighborhoods are explored. It is due to the fact that the initial neighborhood structures may be explored more than the final ones. We followed a deterministic order of exploring N_1 first and then N_2. The proposed VND is based on the first improvement strategy. In both the neighborhoods N_1 and N_2, if the VND encounters a better solution than the current solution, then it is immediately replaced by new better solution and the search starts again from the neighborhood N_1. This process continues until it traverse both the neighborhoods and there is no improved solution found. Finally, the solution returned by VND is local optimum with respect to both the neighborhoods.

Algorithm 3: Pseudo code of GVNS for k-TSP

```
Input: Set of parameters for GVNS
Output: Best solution found
while iter ≤ iter_max do
    S ← Generate_Initial_Solution();
    p ← 1;
    while p ≤ p_max do
        S' ← Shake(S, N_p);
        S'' ← VND(S');
        if F(S'') < F(S) then
            S ← S'';
            p ← 1;
        else
            p ← p + 1;
return bestsolution S;
```

4. Computational results

As our approach GVNS is the first metaheuristic approach, there were no benchmark instances available in the literature for the k-TSP. Hence, it is inevitable to generate the test instances. We generated our own

k-TSP test instances, which are actually derived from the TSPLIB[1]. These instances have cities ranging from 14 to 200, and have a $n \times n$ distance matrix format. Our approach GVNS is executed on each test instance ten times independently, each time with a different random seed.

Our approach GVNS is implemented in C and executed on a Linux based 3.10 GHz Core-i5 system with 4 GB RAM. In all our experiments with the GVNS, we used the following parameters – GVNS is iterated 100 times, i.e., $iter_{max}$=100, and only 2 neighborhoods are used, i.e., p_{max}=2. For comparison between these neighborhoods, the GVNS is executed individually with neighborhood N_1 (i.e., GVNS(N_1)), neighborhood N_2 (i.e., GVNS(N_2)) and both neighborhoods N_1 and N_2 (i.e., GVNS($N_1 + N_2$)).

Table 1. Results of GVNS approaches on instances with $k = \lfloor \frac{n}{4} \rfloor$

Instance	k	**GVNS(N_1)**			**GVNS(N_2)**			**GVNS($N_1 + N_2$)**		
		Best	Worst	Average	Best	Worst	Average	Best	Worst	Average
att48	12	**1925**	**1925**	**1925.00**	2514	2662	2528.80	**1925**	**1925**	**1925.00**
bayg29	7	**350**	**350**	**350.00**	**350**	357	350.70	**350**	**350**	**350.00**
bays29	7	**418**	**418**	**418.00**	**418**	**418**	**418.00**	**418**	**418**	**418.00**
berlin52	13	686	707	688.10	742	763	744.10	**679**	**679**	**679.00**
bier127	31	11477	11941	11666.90	11994	13258	12120.40	**10747**	**10770**	**10749.30**
brazil58	14	6496	6500	6496.40	6496	6500	6496.40	**4965**	**4965**	**4965.00**
brg180	45	530	530	530.00	550	590	554.00	**510**	**520**	**517.00**
burma14	3	**359**	**359**	**359.00**	**359**	**359**	**359.00**	**359**	**359**	**359.00**
ch130	32	1319	1415	1328.60	1373	1598	1395.50	**1130**	**1130**	**1130.00**
ch150	37	1354	1444	1363.00	1483	1491	1483.80	**1276**	**1276**	**1276.00**
d198	49	5211	5236	5213.50	5250	5275	5252.50	**5002**	**5127**	**5090.90**
dantzig42	10	**145**	158	146.30	155	156	155.10	**145**	**145**	**145.00**
eil101	25	149	149	149.00	155	155	155.00	**107**	**108**	**107.50**
eil51	12	83	83	83.00	83	83	83.00	**82**	**82**	**82.00**
eil76	19	119	120	119.60	129	130	129.10	**102**	**117**	**106.90**
fri26	6	**243**	**243**	**243.00**	**243**	**243**	**243.00**	**243**	**243**	**243.00**
gr120	30	1411	1417	1411.60	1411	1417	1411.60	**1396**	**1396**	**1396.00**
gr137	34	18742	18764	18744.20	18742	18764	18744.20	**17509**	**17509**	**17509.00**
gr17	4	**234**	**234**	**234.00**	245	245	245.00	**234**	**234**	**234.00**
gr21	5	**324**	**324**	**324.00**	**324**	**324**	**324.00**	**324**	**324**	**324.00**
gr24	6	**264**	**264**	**264.00**	**264**	**264**	**264.00**	**264**	**264**	**264.00**
gr48	12	**874**	**874**	**874.00**	984	984	984.00	**874**	**874**	**874.00**
gr96	24	11383	12031	11447.80	11383	12031	11447.80	**10821**	**11041**	**10856.30**
hk48	12	2850	2850	2850.00	3304	3304	3304.00	**2827**	**2827**	**2827.00**
kroA100	25	5318	5341	5320.30	6226	6335	6236.90	**5050**	**5050**	**5050.00**
kroA150	37	6742	6756	6750.80	7286	7806	7338.00	**6295**	**6648**	**6586.80**
kroA200	50	7998	8273	8025.50	7998	8273	8025.50	**6826**	**6961**	**6906.20**
kroB100	25	**4605**	**4605**	**4605.00**	4930	4938	4930.80	**4605**	**4605**	**4605.00**
kroB150	37	7479	7573	7488.40	7802	7896	7811.40	**6120**	**6180**	**6150.70**
kroB200	50	8175	8381	8195.60	8175	8381	8195.60	**6100**	**6539**	**6485.90**
kroC100	25	6248	6248	6248.00	6577	6577	6577.00	**4967**	**5363**	**5125.40**
kroD100	25	5350	5485	5363.50	5495	5579	5503.40	**4762**	**4787**	**4784.50**
kroE100	25	**3905**	4014	3915.90	4569	4722	4584.30	**3905**	**3910**	**3905.50**
lin105	26	2779	2803	2781.40	2881	2905	2883.40	**2606**	**2606**	**2606.00**
pr107	26	8615	8705	8624.00	9266	9582	9297.60	**8443**	**8443**	**8443.00**
pr124	31	14516	14639	14528.30	15618	17016	15757.80	**14325**	**14325**	**14325.00**
pr136	34	24315	25528	24436.30	24315	25528	24436.30	**21367**	**23857**	**23352.00**
pr144	36	14437	14437	14437.00	14437	14437	14437.00	**14327**	**14327**	**14327.00**
pr152	38	**20029**	**20029**	**20029.00**	**20029**	**20029**	**20029.00**	**20029**	**20029**	**20029.00**
pr76	19	27179	27179	27179.00	28464	28464	28464.00	**23450**	**23450**	**23450.00**
rat195	48	584	598	586.70	648	705	653.70	**565**	**575**	**570.50**
rat99	24	302	302	302.00	305	305	305.00	**291**	**291**	**291.00**
rd100	25	1586	1607	1588.10	1675	1696	1677.10	**1438**	**1500**	**1460.00**
si175	43	5136	5240	5179.50	5695	5719	5697.40	**4959**	**5096**	**5062.70**
st70	17	**120**	**120**	**120.00**	**120**	**120**	**120.00**	**120**	**120**	**120.00**
swiss42	10	**192**	**192**	**192.00**	**192**	**192**	**192.00**	**192**	**192**	**192.00**
u159	39	9392	9459	9398.70	9534	9601	9540.70	**9085**	**9085**	**9085.00**
ulysses16	4	**935**	**935**	**935.00**	**935**	**935**	**935.00**	**935**	**935**	**935.00**
ulysses22	5	**747**	**747**	**747.00**	970	970	970.00	**747**	**747**	**747.00**

Tables 1, 2 and 3 report the performance of the aforementioned approaches GVNS(N_1), GVNS(N_2) and GVNS($N_1 + N_2$). These tables are only differs by k value, which is set as $k = \lfloor \frac{n}{4} \rfloor$, $k = \lfloor \frac{n}{2} \rfloor$, $k = \lfloor \frac{3 * n}{4} \rfloor$ respectively for table1, 2 and 3. Please note that the n is the total number of cities in the instance and the salesman has to visit k cities by including the depot(i.e., first city). In all these tables, the first column represents the name of the instance with total number of cities in the end. The second column (k) shows the number of cities that needs to be visited by the salesman. The columns (Best, Worst & Average) reports the

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

Table 2. Results of GVNS approaches on instances with $k = \lfloor \frac{n}{2} \rfloor$

Instance	k	**GVNS(N_1)**			**GVNS(N_2)**			**GVNS($N_1 + N_2$)**		
		Best	Worst	Average	Best	Worst	Average	Best	Worst	Average
att48	24	3865	3865	3865.00	4943	5404	4989.10	**3603**	**3603**	**3603.00**
bayg29	14	643	643	643.00	686	744	691.80	**626**	**626**	**626.00**
bays29	14	784	784	784.00	826	826	826.00	**733**	**733**	**733.00**
berlin52	26	2163	2163	2163.00	2269	2269	2269.00	**2006**	**2041**	**2037.50**
bier127	63	26835	27935	26945.00	28900	31153	29125.30	**26107**	**26338**	**26133.30**
brazil58	29	8235	8703	8281.80	11868	12222	11903.40	**8062**	**8235**	**8092.10**
brg180	90	1030	1050	1043.00	1100	1180	1108.00	**1010**	**1030**	**1022.00**
burma14	7	1298	1298	1298.00	1366	1366	1366.00	**1272**	**1272**	**1272.00**
ch130	65	2930	2952	2932.20	3145	3464	3176.90	**2576**	**2653**	**2632.40**
ch150	75	3140	3148	3140.80	3543	3650	3553.70	**2935**	**3029**	**3002.40**
d198	99	7378	7458	7386.00	7543	7623	7551.00	**7086**	**7149**	**7130.40**
dantzig42	21	284	285	284.10	296	310	297.40	**260**	**260**	**260.00**
eil101	50	242	252	247.00	276	281	276.50	**234**	**239**	**236.10**
eil51	25	198	201	198.30	198	201	198.30	**175**	**175**	**175.00**
eil76	38	233	233	233.00	252	254	252.20	**219**	**222**	**221.40**
fri26	13	446	446	446.00	489	489	489.00	**414**	**414**	**414.00**
gr120	60	3102	3233	3160.60	3502	3537	3505.50	**2917**	**2919**	**2918.40**
gr137	68	31933	32012	31940.90	33930	34011	33938.10	**30897**	**31784**	**31401.20**
gr17	8	**517**	**517**	**517.00**	**517**	**517**	**517.00**	517	517	517.00
gr21	10	982	999	983.70	**918**	**918**	**918.00**	918	918	918.00
gr24	12	512	512	512.00	512	512	512.00	**504**	**504**	**504.00**
gr48	24	**1925**	1986	1931.10	2034	2095	2040.10	**1925**	**1925**	**1925.00**
gr96	48	23653	24297	23717.40	23844	25609	24020.50	**22027**	**22196**	**22094.60**
hk48	24	**4759**	**4759**	**4759.00**	5409	5526	5420.70	**4759**	**4759**	**4759.00**
kroA100	50	11775	11893	11786.80	12646	12754	12656.80	**10204**	**10208**	**10206.40**
kroA150	75	12834	12834	12834.00	15061	15666	15121.50	**12722**	**12762**	**12758.00**
kroA200	100	15435	15732	15464.70	16159	16620	16205.10	**14379**	**14542**	**14445.80**
kroB100	50	11694	12238	11748.40	11694	12238	11748.40	**9917**	**10328**	**10227.20**
kroB150	75	13676	14192	13727.60	15329	15645	15360.60	**12040**	**12350**	**12098.00**
kroB200	100	15713	16106	15752.30	16593	17275	16661.20	**13113**	**14051**	**13643.10**
kroC100	50	12991	13181	13010.00	12991	13181	13010.00	**9729**	**9820**	**9790.10**
kroD100	50	11498	11626	11510.80	11498	11626	11510.80	**9614**	**9705**	**9623.10**
kroE100	50	10597	10950	10632.30	11430	11980	11485.00	**10053**	**10065**	**10061.40**
lin105	52	6130	6360	6153.00	6130	6360	6153.00	**5920**	**5944**	**5922.40**
pr107	53	19131	19775	19195.40	19425	20271	19509.60	**18028**	**18028**	**18028.00**
pr124	62	25038	25038	25038.00	27488	29355	27674.70	**22998**	**24431**	**23908.10**
pr136	68	50303	52668	50539.50	50303	52668	50539.50	**47909**	**47919**	**47910.00**
pr144	72	29283	34914	29846.10	29283	34914	29846.10	**28964**	**34644**	**29532.00**
pr152	76	43976	44433	44021.70	44094	45211	44205.70	**41641**	**43403**	**41852.00**
pr76	38	44675	44849	44692.40	51378	52764	51516.60	**41813**	**41970**	**41922.90**
rat195	97	1176	1217	1195.10	1264	1342	1271.80	**1146**	**1153**	**1149.30**
rat99	49	622	622	622.00	638	657	639.90	**574**	**585**	**582.30**
rd100	50	4012	4033	4014.10	4052	4073	4054.10	**3392**	**3554**	**3420.20**
si175	87	10369	10374	10369.50	10835	10879	10839.40	**10283**	**10342**	**10302.10**
st70	35	**302**	311	302.90	302	311	302.90	**302**	**306**	**302.40**
swiss42	21	469	515	473.60	469	515	473.60	**458**	**458**	**458.00**
u159	79	18841	19850	19058.60	22879	23033	22894.40	**18491**	**18556**	**18522.60**
ulysses16	8	2210	2232	2212.20	2362	2362	2362.00	**1685**	**1685**	**1685.00**
ulysses22	11	2489	2489	2489.00	2498	2518	2500.00	**1902**	**1958**	**1908.30**

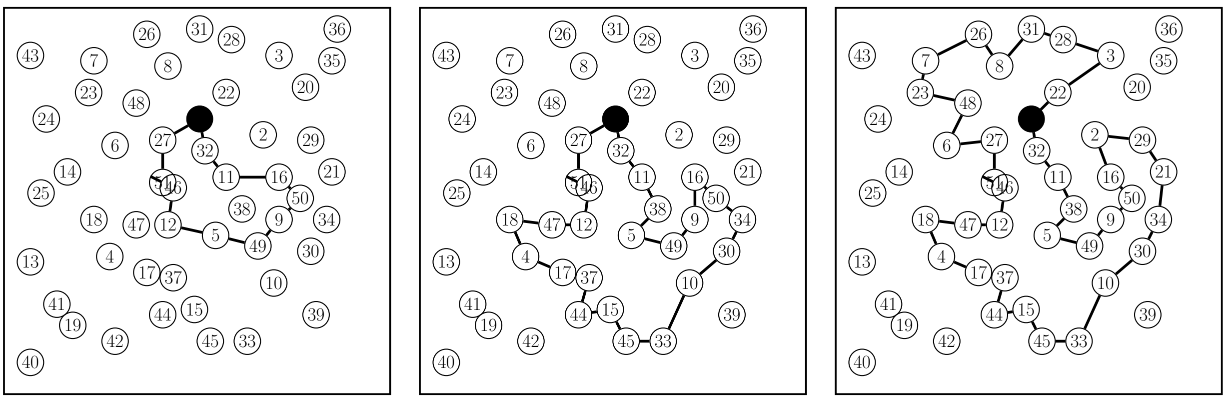
best, worst and average costs over ten independent runs, respectively. The best values are reported in bold for ease of identification. These tables clearly show that GVNS($N_1 + N_2$) performed better than GVNS(N_1) and GVNS(N_2). When it comes to the comparison between two individual neighborhoods, GVNS(N_1) performed better than GVNS(N_2). Thus, it shows the importance of choosing a proper subset of cities to visit by a salesman rather than order of cities to visit. Figure1 plots the solutions obtained by our GVNS approach for the instance eil51 with different k values(12, 25, 38). In all the plots of this figure, the cities are shown as circles having corresponding city numbers, and the first city(i.e., depot) where the salesman has to start and end is shown as black colored circle. This figure depicts the variation in tour and distance according to the k value.

5. Conclusions

In this paper, we have proposed a simple and efficient general variable neighborhood search algorithm for the k-TSP. The main component of the proposed GVNS approach is the variable neighborhood descent which uses two neighborhood structures based on exchange (N_1) and swap (N_2) operations. These two neighborhood structures effectively handle both the characteristics of the k-TSP, i.e., subset selection and permutation of the cities. To evaluate the proposed GVNS approach with different neighborhood structures,

Table 3. Results of GVNS approaches on instances with $k = \lfloor \frac{3*n}{4} \rfloor$

Instance	k	GVNS(N_1)			GVNS(N_2)			GVNS($N_1 + N_2$)		
		Best	Worst	Average	Best	Worst	Average	Best	Worst	Average
att48	36	6714	6801	6722.70	6863	7444	6921.10	**6563**	**6563**	**6563.00**
bayg29	21	1028	1028	1028.00	1055	1113	1060.80	**999**	**999**	**999.00**
bays29	21	1204	1204	1204.00	1246	1246	1246.00	**1194**	**1194**	**1194.00**
berlin52	39	4555	4642	4563.70	4860	4947	4868.70	**4213**	**4441**	**4237.50**
bier127	95	54291	56616	54523.50	54490	56815	54722.50	**50284**	**50903**	**50544.00**
brazil58	43	**11614**	12082	11660.80	14963	15747	15041.40	**11614**	**11849**	**11637.50**
brg180	135	1610	1630	1618.00	1660	1780	1672.00	**1510**	**1540**	**1524.00**
burma14	10	1693	1754	1699.10	1643	1656	1644.30	**1642**	**1642**	**1642.00**
ch130	97	4403	4770	4439.70	4719	5139	4761.00	**4213**	**4260**	**4227.60**
ch150	112	5255	5370	5266.50	5311	5426	5322.50	**4637**	**4690**	**4656.10**
d198	148	9795	10037	9819.20	10045	10269	10067.40	**9363**	**9483**	**9436.50**
dantzig42	31	478	496	479.80	478	496	479.80	**442**	**457**	**443.50**
eil101	75	424	426	425.80	458	467	458.90	**406**	**408**	**406.40**
eil51	38	309	309	309.00	316	327	317.10	**287**	**287**	**287.00**
eil76	57	372	373	372.10	382	387	382.50	**342**	**355**	**351.00**
fri26	19	682	682	682.00	689	689	689.00	**601**	**601**	**601.00**
gr120	90	4727	4794	4748.10	5096	5195	5105.90	**4501**	**4536**	**4520.20**
gr137	102	52699	53282	52757.30	52699	53282	52757.30	**47465**	**48623**	**47943.50**
gr17	12	**951**	**951**	**951.00**	**951**	**951**	**951.00**	951	951	951.00
gr21	15	1565	1582	1566.70	1565	1582	1566.70	**1501**	**1501**	**1501.00**
gr24	18	852	852	852.00	852	852	852.00	**844**	**844**	**844.00**
gr48	36	3548	3627	3555.90	3548	3627	3555.90	**3333**	**3352**	**3337.40**
gr96	72	41504	43679	41721.50	41504	43679	41721.50	**31717**	**32965**	**32353.10**
hk48	36	7631	7883	7656.20	7963	8165	7983.20	**7400**	**7411**	**7409.90**
kroA100	75	16288	16436	16315.30	18256	18580	18288.40	**15740**	**15901**	**15772.20**
kroA150	112	20951	21457	21001.60	21792	22475	21860.30	**18809**	**19223**	**19144.40**
kroA200	150	23898	25053	24013.50	24262	25677	24403.50	**20135**	**20469**	**20289.60**
kroB100	75	16535	17282	16708.00	18663	19238	18720.50	**15346**	**15493**	**15467.80**
kroB150	112	21876	22550	21943.40	21876	22550	21943.40	**17349**	**17672**	**17554.80**
kroB200	150	25058	26124	25164.60	25058	26124	25164.60	**20459**	**21272**	**20981.30**
kroC100	75	17531	17739	17551.80	18465	19047	18523.20	**14871**	**16738**	**16002.40**
kroD100	75	17079	17243	17095.40	17721	17942	17743.10	**15630**	**15793**	**15729.50**
kroE100	75	16169	16604	16402.70	17771	18506	17844.50	**15179**	**15236**	**15214.10**
lin105	78	9444	9844	9484.00	9469	9877	9509.80	**8999**	**9034**	**9008.90**
pr107	80	40731	41687	40826.60	40959	42699	41133.00	**38579**	**39930**	**39128.80**
pr124	93	39978	40846	40388.80	41102	43281	41319.90	**39203**	**39423**	**39236.80**
pr136	102	78807	82446	79170.90	78807	82446	79170.90	**70790**	**74732**	**71782.40**
pr144	108	48403	54366	48999.30	48878	54823	49472.50	**44657**	**50147**	**45206.00**
pr152	114	59070	59527	59115.70	59926	61777	60111.10	**56727**	**57075**	**56761.80**
pr76	57	67800	68492	67869.20	70925	77449	71577.40	**64990**	**65199**	**65018.90**
rat195	146	1787	1815	1793.50	1890	2064	1907.40	**1713**	**1718**	**1715.80**
rat99	74	947	994	951.70	947	994	951.70	**870**	**870**	**870.00**
rd100	75	6247	6268	6249.10	6247	6268	6249.10	**5175**	**6136**	**5980.50**
si175	131	15907	16058	16005.40	16367	16614	16391.70	**15723**	**15765**	**15749.50**
st70	52	482	487	482.50	494	510	495.60	**477**	**486**	**480.10**
swiss42	31	773	918	787.50	773	918	787.50	**760**	**760**	**760.00**
u159	119	31296	31895	31355.90	31296	31895	31355.90	**27612**	**28182**	**28062.70**
ulysses16	12	3184	3264	3192.00	3184	3264	3195.20	**3183**	**3183**	**3183.00**
ulysses22	16	3110	3124	3111.40	3240	3254	3241.40	**2968**	**2968**	**2968.00**



(a) k=12 & *distance*=82 (b) k=25 & *distance*=175 (c) k=38 & *distance*=287

Fig. 1. Solutions of instance eil51 obtained by GVNS with different k values

test instances with various sizes are generated for the k-TSP. Computational results on these test instances show that the GVNS with both neighborhoods(i.e., GVNS($N_1 + N_2$)) performs better than the two GVNS approaches with individual neighborhoods (i.e., GVNS(N_1) & GVNS(N_2)). Our GVNS approach being the first metaheuristic approach will serve as the baseline for future metaheuristic approaches for the k-TSP. Similar approaches can be developed for other related problems where different neighborhoods need to be explored. As a future work, we intend to investigate the possibility of developing a population based metaheuristic approach by utilizing the components of GVNS to solve the profitable tour problem.

Acknowledgements

The first and second authors acknowledge the financial support received from the Council of Scientific & Industrial Research (CSIR), Government of India in the form of a Senior Research Fellowship.

References

[1] S. Arora, G. Karakostas, A 2+ ε approximation algorithm for the k-mst problem, in: Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms, Society for Industrial and Applied Mathematics, 2000, pp. 754–759.
[2] G. Ausiello, V. Bonifaci, S. Leonardi, A. Marchetti-Spaccamela, Prize-collecting traveling salesman and related problems, Handbook of Approximation Algorithms and Metaheuristics 40 (2007) 1–13.
[3] E. Balas, The prize collecting traveling salesman problem, Networks 19 (6) (1989) 621–636.
[4] A. Blum, S. Chawla, D. R. Karger, T. Lane, A. Meyerson, M. Minkoff, Approximation algorithms for orienteering and discounted-reward tsp, SIAM Journal on Computing 37 (2) (2007) 653–670.
[5] A. Blum, R. Ravi, S. Vempala, A constant-factor approximation algorithm for the k mst problem, in: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing, ACM, 1996, pp. 442–448.
[6] M. R. Garey, D. S. Johnson, Computers and intractability: a guide to the theory of NP-completeness, W. H. Freeman, San Francisco, 1979.
[7] N. Garg, A 3-approximation for the minimum tree spanning k vertices, in: Foundations of Computer Science, 1996. Proceedings., 37th Annual Symposium on, IEEE, 1996, pp. 302–309.
[8] N. Garg, Saving an epsilon: a 2-approximation for the k-mst problem in graphs, in: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing, ACM, 2005, pp. 396–402.
[9] P. Hansen, N. Mladenović, J. A. M. Pérez, Variable neighbourhood search: methods and applications, Annals of Operations Research 175 (1) (2010) 367–407.
[10] N. Mladenović, P. Hansen, Variable neighborhood search, Computers & operations research 24 (11) (1997) 1097–1100.
[11] N. Mladenović, R. Todosijević, D. Urošević, An efficient general variable neighborhood search for large travelling salesman problem with time windows, Yugoslav Journal of Operations Research 23 (1) (2013) 19–30.
[12] N. Mladenović, R. Todosijević, D. Urošević, Two level general variable neighborhood search for attractive traveling salesman problem, Computers & Operations Research 52 (2014) 341–348.
[13] N. Mladenović, D. Urošević, A. Ilić, et al., A general variable neighborhood search for the one-commodity pickup-and-delivery travelling salesman problem, European Journal of Operational Research 220 (1) (2012) 270–285.
[14] R. Todosijević, A. Mjirda, M. Mladenović, S. Hanafi, B. Gendron, A general variable neighborhood search variants for the travelling salesman problem with draft limits, Optimization Letters 11 (6) (2017) 1047–1056.