Decision Science Letters 11 (2022) 473–484

Contents lists available at GrowingScience

Decision Science Letters

homepage: www.GrowingScience.com/dsl

An efficient hybrid genetic algorithm for solving truncated travelling salesman problem

S. Purusotham[a], T. Jayanth Kumar[b*], T. Vimala[b] and K.J. Ghanshyam[b]

[a]*Department of Mathematics, School of Advanced Sciences, VIT, Vellore-632014, Tamil Nadu, India*
[b]*Department of Mathematics, Faculty of Engineering and Technology, Jain (Deemed-to-be University), Kanakapura, Bangalore-562112, Karnataka, India*

C H R O N I C L E

Article history:
Received February 12, 2022
Received in revised format:
June 18, 2022
Accepted June 24 2022
Available online
June 24, 2022

Keywords:
Truncated travelling salesman problem
Genetic algorithm
Mutation strategies

A B S T R A C T

This paper considers a practical truncated traveling salesman problem (TTSP), in which the salesman is only required to cover a subset of out of given cities (rather than covering all the given cities as in conventional travelling salesman problem (TSP)) with minimal traversal distance. Thus, every feasible solution tour contains exactly cities including the starting city. However, extensive research on TSP has been received and various efficient solution techniques including exact, heuristic, and metaheuristic algorithms are devoted, a very limited attention has been given to TTSP models because of its solution structure. The TTSP model comprises two types of problems including city selection i.e. as a salesman's trip need not include all the cities, the challenge is to identify which combination of cities are to be visited and which sequence of cities will constitute minimal traversal distance. A hybrid genetic algorithm (GA) comprising sophisticated mutation operators is developed to tackle this problem efficiently. Comparative computational findings suggest that the proposed GA has capability to outperform existing approaches in terms of TTSP results. In addition, the proposed GA report improved results and will serve as a basis for forthcoming TTSP studies.

1. Introduction

One of the most well known and widely studied NP-hard combinatorial optimization problems is the travelling salesman problem (TSP) (Applegate et al., 2006; Stodola et al., 2022). The TSP involves determining the minimum closed tour to cover all the given cities and finds several real-world applications that arises in logistics distribution (Elgesem et al., 2018; Baniasadi et al., 2020), shortest path network (Fatthi et al., 2018), warehouse order picking (Madani et al., 2021), vehicle routing (Demir et al., 2019), dairy beverage delivery (Palhares and Araújo, 2018) etc. Furthermore, the foraging behaviour of birds, animals, and insects, which is governed by power-law functions, can also be better understood using the TSP models (Stanley and Buldyrev, 2001). Due to its diversified applications, several practical models have been addressed by transforming the traditional TSP structure. According to Bhavani and Murthy (2006), selection of $k(<n)$ out of n cities is known as truncation and there exists n_{C_k} feasible solutions for the TSP. However, the problem becomes more difficult when higher values of n and k. Thus, exact algorithms will not work for higher dimensional problems and it is inevitable to develop an efficient heuristic or metaheuristic algorithm. In this study, a real-world variant of TSP and an exceptional case of the prize collection TSP (PCTSP) called truncated Travelling salesman problem (TTSP) is addressed. The present study mainly focuses on the developments of TTSP. The TTSP received little attention from the researchers as compared to TSP and its other variants. The TTSP is also known as *k*-TSP in some studies (Venkatesh et al., 2018). Saksena and

* Corresponding author.
E-mail address: jayanth@jainuniversity.ac.in (T. Jayanth Kumar)

doi: 10.5267/dsl.2022.6.003

Kumar (1966), Ibaraki (1973), and Laporte et al. (1984) are few investigated TTSP with an added constraint that the salesman must visit a subset of k cities.

The objective of TTSP is to identify a set of k out of n cities as well as to find the permutation of k cities that optimizes the total distance covered by the salesman. The TTSP finds several applications in routing scenarios and rural health care delivery. For instance, logistic distribution of goods delivery, the distribution begins in the home/starting/depot city, visits only a restricted number of cities and concludes the tour at the depot city. Fig. 1 and Fig. 2 demonstrate the difference between TSP and TTSP, respectively. In Fig.1, the salesman starts from the home city, travels through the remaining 7 cities only once and return to the home city with the minimal total travel distance. Although, the salesman in Fig. 2 begins from the depot city, not necessary to visit all the given 8 cities but required to visit only $k = 5(< n)$ cities.

In the literature, some solution techniques such as heuristics and exact approaches are presented for solving TTSP and its variants. Gensch (1978) presented TTSP as a time restriction based industrial scheduling problem. In this model, the salesperson must select a subset of cities to be covered with shortest distance within the time constraint. The Lagrangian relaxation based branch and bound method has been devoted as a solution to this problem, and it is capable of addressing problems of higher dimensions. Mittenthal and Noon (1992) suggested an efficient heuristic approach that combines an insertion and deletion strategies for solving TTSP with an added constraint and its computational results demonstrate its usefulness. Westerlund et al. (2006) studied TTSP and an efficient heuristic decomposition based column generation approach has suggested to tackle it. Giardini and Kalmár-Nagy (2011) has applied TTSP to multiagent planning scenarios that are solved through a hybrid Genetic algorithm (GA) and its performance is demonstrated with computational results. According to Stetsyuk (2016), the task of determining the *k*-node least cycle is more challenging than determining the shortest Hamiltonian cycle with *n*-nodes. Venkatesh et al. (2018) has developed the first metaheuristic namely general variable neighbourhood search algorithm (GVNS) that integrates two neighbourhood strategies such as exchange and swap processes to solve *k*-TSP effectively. Pandiri and Singh, (2020) have developed two multi-start heuristic algorithms for solving *k*-TSP. More recently, Singamsetty et al., (2021) has investigated a variant of TTSP called Open TTSP and developed an efficient hybrid genetic algorithm. Several researchers have shown that the Genetic algorithm (GA) is more proficient in dealing with TSP and its variants than any other metaheuristic technique (Bahaabadi et al., 2012; Singh et al., 2018; Ha et al., 2020; Sharma and Jain, 2021).

In addition to the aforementioned studies, the truncation aspect has been considered in various optimization contexts including merchant sub tour problem (Verweij and Aardal (2003)), truncated M-TSP (Bhavani and Murthy, 2006), spanning networks (Kumar & Purusotham, 2017; Adasme & Dehghan Firoozabadi, 2020), wireless sensor networks (Adasme et al., 2018), bicriteria TTSP with time threshold (Kumar Thenepalle and Singamsetty, 2018), green communication in underground mines (Xu et al., 2018), profitable tour problem (Dasari et al., 2021), open TTSP (Singamsetty et al., 2021), LPG delivery problem (Singamsetty & Thenepalle, 2021), *k*-cardinality unbalanced assignment problem (Prakash et al., 2022). With the motivation from the above-cited works, the present study address the TTSP and develop a simple and efficient metaheuristic algorithm called the nearest neighbourhood based Genetic algorithm (GA) with complex mutation strategies that provide the best results. The GA is the first evolutionary technique for the TTSP, to the author's best information.

The following is how the rest of the paper is organised: The definition and formulation of the TTSP will be provided in the next section. The GA and its operators will be described in Section 3. The computational findings are demonstrated in Section 4. Section 5 accomplishes with a conclusion and a description of the scope of future work.

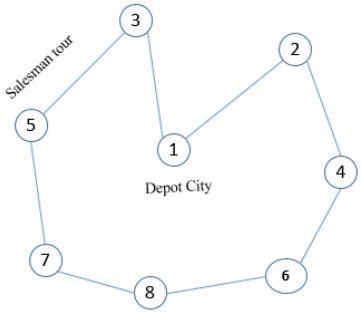


Fig. 1. An arbitrary 8 city TSP solution

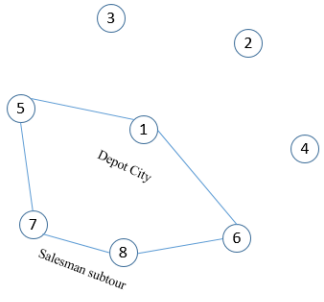


Fig. 2. An arbitrary 5 city TTSP solution

2. Truncated Travelling Salesman Problem

The following notations are used in the present model

Notations	Description
$G=G(N,E)$	An undirected complete edge-weighted graph
$N=\{1,2,...,n\}$	Node set with n cities/nodes
$E=\{(i,j)/i,j\in V;\ i\neq j\}$	Edge set with n^2-n edges
$x_{ij}\in\{0,1\},\forall(i,j)\in E$	A binary variable
$D=\left[d_{ij}\right]_{n\times n}$	A symmetric distance matrix defined over E
$d_{ij}\ ;(d_{ij}=d_{ji};d_{ii}=\infty,d_{ij}>0)$	Distance between the cities i and j
k	Truncation parameter, the cities that the salesman required to cover (i.e. k out of n)
$S(\mid S\mid=k,where\ \ S\subseteq V,k\leq n)$	A subset with k cardinality
$y_i\in\{0,1\},\ i\in V$	Binary variable related to visited cities

The travelling salesman problem (TSP) over G is the problem of determining a Hamiltonian cycle of length n with shortest distance. The present study is mainly dedicated on solving symmetric truncated travelling salesman problem (TTSP), which is a variant of classical TSP. The TTSP can be mathematically formulated over $G=G(N,E)$ defined on a symmetric distance matrix $D=\left[d_{ij}\right]_{n\times n}$ with n cities and n^2-n edges. Each edge of G indicate the path between two cities and it is assigned with positive distance. The salesman cover exactly k out of n cities, starting and ending at home city and each city is to be covered by salesman exactly once. On covering k cities, there exists $\binom{n}{k}\times(k-1)!$ possible feasible solutions. It is note that throughout this study, starting city is assumed the city 1. The TTSP aims to determine an optimal tour of covering k out of n cities with minimal traversal distance. A sub tour of length k is called a tour/feasible tour/ feasible solution and a tour of length that is less than k is called a sub tour/illegitimate tour. The binary variable x_{ij} takes 1 when the salesman covers j^{th} city from i^{th} city, and $x_{ij}=0$ otherwise. Here y_j, the binary variable that assumes 1 when the salesman visits city j and 0 otherwise. The mathematical model of TTSP is given as follows:

The salesman traversal distance can be minimized using the following objective function

$$\text{Minimize } Z=\sum_{i=1}^{n}\sum_{j=1}^{n}d_{ij}\,x_{ij} \tag{1}$$

Constraint set (2-4) is imposed to ensure the salesman begin and conclude his tour at the starting city (i.e. city 1) exactly once and traverse a city and leave from it at most once.

$$\sum_{j=2}^{n}x_{1j}=\sum_{i=2}^{n}x_{i1}=1 \tag{2}$$

$$\sum_{i=1}^{n}x_{ij}\leq 1;\ \forall\ j\in V\ \&\ i\neq j \tag{3}$$

$$\sum_{j=1}^{n}x_{ij}\leq 1;\ \forall\ i\in V\ \&\ i\neq j \tag{4}$$

To ensure that any feasible solution includes k edges, Constraint (5) is enforced. However, it does not promise in giving feasible solutions with those k edges. For example, if $n=8\ \&k=4$, and $(1,3),(3,2),(6,5),(5,1)$ be four edges. This set of edges cannot construct a feasible tour of length 4. Moreover, the cities contained in $S=\{1,3,2,5,6\}$ do not match with the desired cardinality $\mid S\mid=k\neq 4$. The resulting tour must be a cycle that begins and concludes at the starting city. Thus, this constraint alone is not enough to judge the feasible tour.

$$\sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij} = k \tag{5}$$

Degree of each city in the tour should be equal to two. To preserve this, Constraint (6) is presented and it assures that whenever the salesman visits a city, he must depart from that city.

$$\sum_{i=1}^{n} x_{ij} - \sum_{p=1}^{n} x_{jp} = 0;\ \forall\ j \in V \tag{6}$$

However, Constraint (6) cannot control the formation of illegitimate tours with length smaller than k . This can be validated with the example: if $k = 5$, the two illegitimate tours $2-3-2$ and $5-4-6-5$ involves 5 cities and 5 edges that can optimize the TTSP. Hence, Constraint (7) is introduced to avoid the formation of illegitimate tours of length less than $k-1.$

$$x_{1p_1} + x_{p_1p_2} + x_{p_2p_3} + ... + x_{p_{i-1}p_i} + x_{p_i1} \le k-1;\ p_1, p_2 ... p_i \in V / \{1\} \tag{7}$$

To ensure that any feasible tour should include exactly k cities, Constraint (8) is imposed. Finally, the binary variables x_{ij} and y_i are specified in Constraint (9).

$$\sum_{i=1}^{n} y_i = k \tag{8}$$

$$x_{ij} \in \{0,1\}\ \&\ y_i \in \{0,1\} \tag{9}$$

3. Genetic algorithm

The classical Genetic algorithm (GA), which is most extensively used metaheuristic in evolutionary computation (Goldenberg, 1989), is provided first, followed by a detailed discussion of the proposed algorithm. The survival of the fittest approach was first presented by Holland in 1975. The GA, by its nature, begins with a pool of preliminary solutions known as the initial population/ chromosomes where all genetic data is kept. Each number on the chromosome is considered as a gene. In addition, a fitness value is calculated to assess chromosome effectiveness. Each time, based on their fitness values two best parent chromosomes are randomly chosen from the population. The crossover operation is then applied over those two chromosomes that results in two new chromosomes for the next generation. If the newly found chromosomes have higher fitness values, then they will replace the old ones. The newly generated chromosomes are then subjected to a mutation operation to preserve the population's variability. Repeat the selection, crossover, and mutation operations to produce numerous novel chromosomes until the size of the population matches that of the old. The updated population is then used to begin the iteration. Because superior chromosomes are more likely to be designated for crossover, and the newer chromosomes produced are more likely to transfer the properties of their parent chromosomes. Until the predefined conditions are fulfilled, the search process will continue for several generations. This complete procedure is known as classical GA (Hariyadi et al., 2020). Certain studies, on the other hand, have witnessed a crossover-free GA. For example, Liu & Kroll (2016) designed a GA with no crossover operator but comprised composite mutation operators (slide, inversion, swap, insertion, and various combinations) to solve the multi-robot job allocation problem. The crossover-free GA achieves superior outcomes than the classical GA, according to the experimental results provided in this work. Different GA approaches may use different encoding, crossover, and mutation operators, resulting in search process divergence. As a result, it's critical to execute the above actions to ensure that the optimal/suboptimal solution is obtained. A hybrid GA with sophisticated mutation operators (swap, slide, reverse) and no crossover operator is designed to solve TTSP effectively. The essential components of the proposed GA for TTSP are detailed in the subsections below.

3.1. Chromosome representation

For an efficient GA, an appropriate chromosome representation must be used to solve TSP and its variants. The TSP solution can be generally denoted as a chromosome in different methods comprising path representation (Larranaga et. al., 1999), matrix representation (Khan et al., 2009), double chromosome representation (Riazi, 2019) etc. According to Hussain et al. (2017), path representation is extensively used to tackle TSP and its allied models. In this representation, the chromosome respective TSP solution is given by an arrangement of n distinct integers. A chromosome can be denoted as $(g_1, g_2, g_3, ... g_n)$ with the size of n genes, where $g_j \in V, 1 \le j \le n$ indicates a gene (city) in the chromosome. In our study, a modified path representation is used to represent the TTSP solution. Instead of generating a chromosome with

the size of n genes, it is enough to use a chromosome with the size of k genes (since the TTSP tour involves only k genes). Such a path representation with the length of k genes is simple to implement, which will be considered as TTSP solution. For example, if $n=10$, $k=8$, then a sample TTSP solution through path representation is demonstrated in Fig. 3. By attaching the starting city at the beginning and the ending to the chromosome, then the resultant tour 1, 4, 6, 7, 2, 10, 3, 5, 1 is achieved.

Fig. 3. An example solution of TTSP with 8 out of 10 cities

3.2. Initial population encoding

A finite set of chromosomes of a certain size is known as the initial population. This set consists of valid chromosomes, which are created by using the nearest neighbourhood technique and inserting them into the set one by one. It is clear that the optimal solution tour contains the shortest distance related edges. Thus, the distance matrix elements are arranged in increasing order together with their indices. The first chromosome is constructed by picking the shortest distance associated first city and applying the nearest neighbour heuristic (NNH). A second chromosome is formed by picking the next smallest distance related initial city and employing the NNH. In this manner, a set of $(n^2-n)/2$ valid unique chromosomes are produced for $n\times n$ symmetric matrix. Only preferred best chromosomes can then be selected for subsequent processing. Algorithm 3.1 shows the pseudo-code for the NNH. In Fig. 3, the chromosome denote one of the feasible solutions of the TTSP. The series of integers also known as cities are encoded in the genes. Such a chromosome representation is known as path representation, and it is used in this study.

Algorithm 3.1. Nearest neighbour heuristic pseudo-code

begin Nearest_Neighbour
Initialization
Distance matrix
Sort the elements of the distance matrix along with respective indices,
route= $\varnothing$
Create initial population with nearest-neighbour heuristic
    while termination condition not met **do**
        find the least distance corresponding nodes
        select the first node as current city from the two nodes
        start city $\longrightarrow$ current city
        route=route $\cup$current city
        mark the current city as visited
        nearest (current city) $\longrightarrow$ next city
        If route length=desired length, **end**.
        else, nearest city $\longrightarrow$ current city
    end
Output salesman route with desired length
end Nearest_Neighbour

3.3. Fitness function

The fitness values of all chromosomes in the population are computed using the fitness function. One of the critical steps in GA is the selection strategy, which is based on the fitness value. Higher fitness value chromosomes are always having a great probability of being picked for the next generation. The objective function (1) in our study is assumed to be the fitness function. As a result, the higher fitness value chromosome certainly possesses the smallest distance, thus, a higher genetic chance of being picked. The fitness value in the TTSP is nothing but the salesman's total covering distance when touring cities.

3.4. Selection operator

It is one more important part of the GA since it has an impact on its performance. The standard roulette wheel approach is employed as the GA selection operator. Depending on the chromosome fitness value, it chooses a chromosome from its population to pass into a reproducing pool.

3.5. Mutation operator

After the selection operation, mutation operation is executed. Its goal is to escape the GA from becoming stuck in a local optimum and boost the population's evolutionary divergence. The complex mutation operator, which includes the Swap, Reverse swap/Flip, and Slide processes, is used in this work. This complex operator helps to determine the shortest distance in a limited time. A parent chromosome is selected based on a mutation probability P_m. Two separate positions are picked at random from the parent chromosome for a swapping activity, and the genes of these two positions are swapped. In a reverse swap process, two distinct places to define the segment are chosen, and the genes between these positions are swapped. Similarly, two distinct places (say, i^{th} and j^{th} positions) are chosen for a sliding operation. A new offspring is generated by eliminating the gene in i^{th} point and replicate the same in j^{th} point of the parent chromosome. Therefore, genes at $(i+1),(i+2)$ places will be shifted to i^{th} and $(i+1)^{th}$ places, respectively and so on. In the same manner, the gene at i^{th} position is shifted to j^{th} position and the gene at j^{th} place should be relocated to $(j-1)^{th}$ place. Figures 4-6 show illustrations of Swap, Reverse swap/Flip, and Slide strategies, respectively.

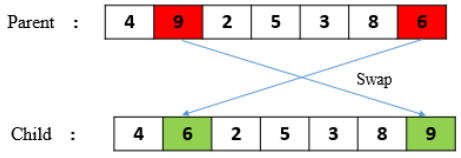


Fig. 4 An illustration for swap operation

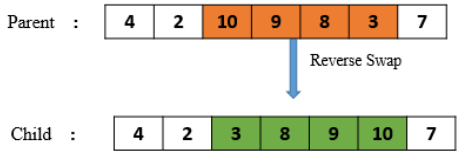


Fig. 5 An illustration for reverse swap operation

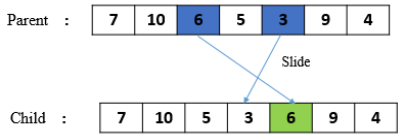


Fig. 6 An illustration for slide operation

3.6. GA parameters

The algorithm's performance is dependent on the parameters' values, such as population size, mutation probability rate, and termination criterion. The population size refers to the number of chromosomes in any single generation and it assumes to be large enough to be 100 in this study. Crossover operator is not addressed in this study. However, a complex mutation operator will fulfil its purpose. The mutation probability rate (P_m) reveals how often mutations occur to the parts of the chromosome. This operator effectively maintains the diversity in the population. Usually, P_m varies from 0.001 and 0.1. To our study, it is assumed as 0.01. Maximum number of generations is to be considered as the end condition for the present GA. The proposed GA is depicted in Fig. 7.

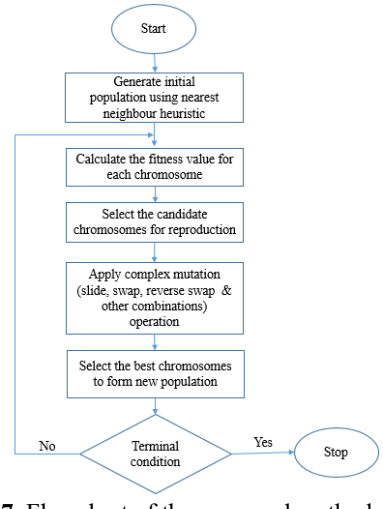


Fig. 7. Flowchart of the proposed methodology

4. Computational results

This section presents computational results. To produce new children in each generation, the proposed GA employs roulette wheel selection and complex mutation (Swap, Reverse Swap, and Slide) operators. The proposed GA was written in MATLAB R2017a on a PC with an Intel Core i3 2.00 GHz processor and 4GB of RAM running Windows 10 Pro 64 bit. Unlike traditional TSP, no benchmark test instances are available to check TTSP. As a result, the TSPLIB benchmark instances are used to generate TTSP instances. Nevertheless, the TTSP and TSP models are not same, but standard test instances and their optimal solutions may be beneficial in testing the GA's competency. The benchmark examples were Euclidean, 2-D symmetric, had different node scales and ranging from 14 to 200 cities.

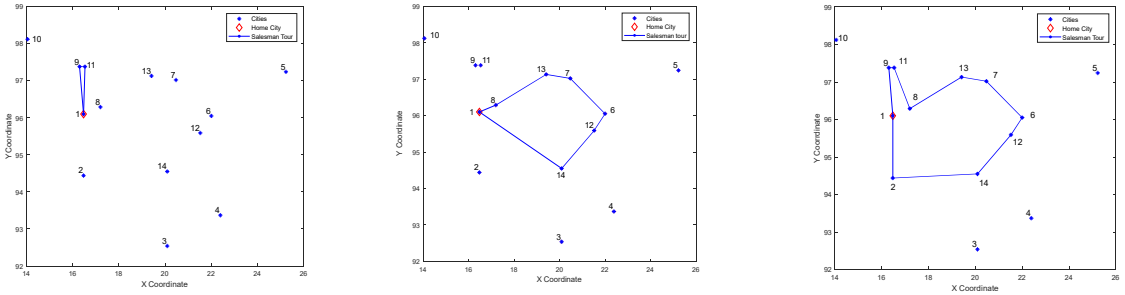
A comparison of our technique against the general variable neighbourhood search (GVNS) with various neighbourhood structures suggested by Venkatesh et al. (2018) is performed to assess the competency of the proposed GA. This comparative study is performed on the same benchmark instances as used in Venkatesh et al. (2018) and a overall 44 benchmark instances. The reference cited in this paragraph reports the results of GVNS with different neighbourhood structures in terms of three parameters namely, best, worst and average solutions on benchmark instances for distinct node scales. In this study, the comparison is done on the two parameters namely best and worst solutions. The developed GA is run ten times on each test instance independently, and the best and worst results are reported after each run. Tables 1, 2 and 3 provide comparative results of the present GA versus the GVNS techniques stated above. Tables 1, 2 and 3 are only varying by k value, which is fixed as $k=\left\lfloor \frac{n}{4} \right\rfloor, k=\left\lfloor \frac{n}{2} \right\rfloor \,\&\, k=\left\lfloor \frac{3*n}{4} \right\rfloor$, respectively.

The first column in each of these tables, *Instance*, contains the test instance name followed by total cities at the end. The size of the benchmark instance and the truncation parameter respectively denotes the next two columns n and k. The Best and Worst labelled columns represent the best and worst solutions generated by different GVNS techniques and by proposed GA, respectively. Finally, the column *Dev. %* denotes the best-known solution deviation percentage, which is evaluated using the formula (10) by utilizing the best-known and best solutions found by the present GA. However, the deviation (%) takes absolute values, both positive and negative values has considered. Here, positive deviation indicates that the solution found by developed GA is superior than best-known solution (BKS). Negative deviation; on the other hand, suggest that the best-found solution (BFS) is worse than the BKS. Both the solutions are identical when it is zero deviation.

$$Deviation\,\% = \left| \frac{Best\ known\ GVNS\ Solution - Best\ GA\ Solution}{Best\ known\ GVNS\ Solution} \right| \times 100 \tag{10}$$

Table 1 reports the best and worst solutions found by the proposed GA tested on 44 test instances with truncation parameter $k=\left\lfloor \frac{n}{4} \right\rfloor$. Of the 44 test instances, it is revealed that for 32 cases, the BKS and BFS are identical. For 6 cases, GA solutions are worse than BKS and for 6 cases, the GA solutions are better than BKS. It is also worth noting that the negative deviation ranges from -0.93% to -6.42%, where as positive deviations varying from 3.20% to 47.91%. Similarly, in Table 2, we considered the similar benchmark instances as used in Table 1 but with distinct $k=\left\lfloor \frac{n}{2} \right\rfloor$. Of the 44 test cases, it is seen that for 24 cases coincides the BKS and BFS. For 5 cases, GA results are worse than BKS and for 15 cases, the GA solutions are better than BKS. It is also evident that the negative deviation varies from -0.57% to -3.41%, where as positive deviations varying from 0.11% to 59.38%. Finally, in Table 3, the similar benchmark instances has considered as used in Table 1 but with distinct $k=\left\lfloor \frac{3*n}{4} \right\rfloor$. Of the 44 test cases, it is observed that for 21 cases coincides the BKS and BFS. For 6 cases, GA results are worse than BKS and for 17 cases, the proposed GA found improved solutions than BKS. It is also evident that the negative deviation varies from -0.56% to -3.94%, while positive deviation ranging from 0.11% to 56.18%.

From overall observations, it is evident that the proposed GA outperforms the existing GVNS approach and certainly provides the best solutions. Note that the improved solutions in all three tables are reported in boldface. Figure 6 depict the solutions produced by the proposed GA for the test instance *burma14* with various k values (3, 7, 10). The cities are depicted with star symbols marked with their respective city numbers in all plots, and the home city where the salesperson begins and concludes his trip is indicated as a red - coloured diamond symbol. This figure evidently show the how k value play a key role in in the salesman tour and its traversal distance.



(a) $k = 3$ & distance $= 280$ (b) $k = 7$ & distance $= 1236$ (c) $k = 10$ & distance $= 1575$

Fig. 8. Salesman route plans for *burma14* with distinct k values by proposed GA

Table 1

Results of proposed GA Vs. GVNS on benchmark instances with $k = \left\lfloor \frac{n}{4} \right\rfloor$

Instance	n	k	*GVNS* (N_1)		*GVNS* (N_2)		*GVNS* $(N_1 + N_2)$		*Proposed GA*		*Dev.* (%)
			Best	*Worst*	*Best*	*Worst*	*Best*	*Worst*	*Best*	*Worst*	
bayg29	29	7	350	350	350	357	350	350	350	350	0
bays29	29	7	418	418	418	418	418	418	418	425	0
berlin52	52	13	686	707	742	763	679	679	679	679	0
bier127	127	31	11477	11941	11994	13258	10747	10770	11050	11451	-2.59
burma14	14	3	359	359	359	359	359	359	**280**	**331**	22
ch130	130	32	1319	1415	1373	1598	1130	1130	1130	1220	0
ch150	150	37	1354	1444	1483	1491	1276	1276	1294	1431	-1.41
d198	198	49	5211	5236	5250	5275	5002	5127	5002	5248	0
dantzig42	42	10	145	158	155	156	145	145	145	149	0
eil101	101	25	149	149	155	155	107	108	108	117	-0.93
eil51	51	12	83	83	83	83	82	82	82	84	0
eil76	76	19	119	120	129	130	102	117	102	107	0
fri26	26	6	243	243	243	243	243	243	243	243	0
gr137	137	34	18742	18764	18742	18764	17509	17509	18634	19612	-6.42
gr17	17	4	234	234	245	245	234	234	234	234	0
gr21	21	5	324	324	324	324	324	324	324	324	0
gr24	24	6	264	264	264	264	264	264	264	264	0
gr48	48	12	874	874	984	984	874	874	874	874	0
gr96	96	24	11383	12031	11383	12031	10821	11041	**9543**	**9543**	11.81
hk48	48	12	2850	2850	3304	3304	2827	2827	2827	3135	0
kroA100	100	25	5318	5341	6226	6335	5050	5050	5203	5203	-3.02
kroA150	150	37	6742	6756	7286	7806	6295	6648	**6045**	**6045**	3.97
kroA200	200	50	7998	8273	7998	8273	6826	6961	**6607**	**6937**	3.20
kroB100	100	25	4605	4605	4930	4938	4605	4605	**4303**	**4495**	6.55
kroB150	150	37	7479	7573	7802	7896	6120	6180	6368	6368	-4.05
kroB200	200	50	8175	8381	8175	8381	6100	6539	6100	6353	0
kroC100	100	25	6248	6248	6577	6577	4967	5363	4967	5058	0
kroD100	100	25	5350	5485	5495	5579	4762	4787	4762	4928	0
kroE100	100	25	3905	4014	4569	4722	3905	3910	3905	3933	0
lin105	105	26	2779	2803	2881	2905	2606	2606	2606	2613	0
pr107	107	26	8615	8705	9266	9582	8443	8443	8443	8518	0
pr124	124	31	14516	14639	15618	17016	14325	14325	14325	16252	0
pr136	136	34	24315	25528	24315	25538	21367	23857	21367	23678	0
pr144	144	36	14437	14437	14437	14437	14327	14327	14327	14327	0
pr152	152	38	20029	20029	20029	20029	20029	20029	20029	20200	0
Pr76	76	19	27179	27179	28464	28464	23450	23450	23450	25606	0
rat195	195	48	584	598	648	705	565	575	565	570	0
rat99	99	24	302	302	305	305	291	291	291	298	0
rd100	100	25	1586	1607	1675	1696	1438	1500	1438	1455	0
St70	70	17	120	120	120	120	120	120	120	132	0
Swiss42	42	10	192	192	192	192	192	192	**100**	**102**	47.91
U159	159	39	9392	9459	9534	9601	9085	9085	9085	9863	0
Ulysses16	16	4	935	935	935	935	935	935	935	976	0
Ulysses22	22	5	747	747	970	970	747	747	747	752	0

Table 2

Results of proposed GA Vs. GVNS on benchmark instances with $k = \left\lfloor \frac{n}{2} \right\rfloor$

Instance	*n*	*k*	*GVNS* (N_1)		*GVNS* (N_1)		*GVNS* ($N_1 + N_2$)		*Proposed GA*		*Dev. (%)*
			Best	*Worst*	*Best*	*Worst*	*Best*	*Worst*	*Best*	*Worst*	
bayg29	29	14	643	643	686	744	626	626	626	644	0
bays29	29	14	784	784	826	826	733	733	733	751	0
berlin52	52	26	2163	2163	2269	2269	2006	2006	**1876**	**1876**	6.48
Bier127	127	63	26835	27935	28900	31153	26107	26338	26107	26536	0
burma14	14	7	1298	1298	1366	1366	1272	1272	**1236**	**1236**	2.83
ch130	130	65	2930	2952	3145	3464	2576	2653	**2573**	**2590**	0.11
ch150	150	75	3140	3148	3543	3650	2935	3029	2935	3114	0
d198	198	99	7378	7458	7543	7623	7086	7149	7086	7458	0
dantzig42	42	21	284	285	296	310	260	260	260	272	0
eil101	101	50	242	252	276	281	234	239	242	250	-3.41
eil51	51	25	198	201	198	201	175	175	175	188	0
eil76	76	38	233	233	252	254	219	222	219	234	0
fri26	26	13	446	446	489	489	414	414	414	425	0
gr137	137	68	31933	32012	33930	34011	30897	31784	31784	36567	-2.87
gr17	17	8	517	517	517	517	517	517	517	517	0
gr21	21	10	982	999	918	918	918	918	918	918	0
gr24	24	12	512	512	512	512	504	504	504	534	0
gr48	48	24	1925	1986	2034	2095	1925	1925	1925	1951	0
gr96	96	48	23653	24297	23844	25609	22027	22196	**19876**	**19876**	9.76
hk48	48	24	4759	4759	5409	5526	4759	4759	4759	5146	0
kroA100	100	50	11775	11893	12646	12754	10204	10208	**10050**	**10117**	1.50
kroA150	150	75	12834	12834	15061	15666	12722	12762	**12429**	**12429**	2.30
kroA200	200	100	15435	15732	16159	16620	14379	14542	**13965**	**13965**	2.87
kroB100	100	50	11694	12238	11694	12238	9917	10328	**9638**	**9638**	2.81
kroB150	150	75	13676	14192	15329	15645	12040	12350	12040	12105	0
kroB200	200	100	15713	16106	16593	17275	13113	14051	13113	13877	0
kroC100	100	50	12991	13181	12991	13181	9729	9820	9729	10382	0
kroD100	100	50	11498	11626	11498	11626	9614	9705	**9227**	**9823**	4.02
kroE100	100	50	10597	10950	11430	11980	10053	10065	**9576**	**9988**	4.74
lin105	100	52	6130	6360	6130	6360	5920	5944	5954	5980	-0.57
pr107	107	53	19131	19775	19425	20271	18028	18028	18028	18079	0
pr124	124	62	25038	25038	27488	29355	22998	24431	22998	24945	0
pr136	136	68	50303	52668	50303	52668	47909	47919	47909	50728	0
pr144	144	72	29283	34914	29283	34914	28964	34644	**28059**	**28059**	3.12
pr152	152	76	43976	44433	44094	45211	41641	43403	**38863**	**39142**	6.67
Pr76	76	38	44675	44849	51378	52764	41813	41970	42638	44481	-1.97
Rat195	195	97	1176	1217	1264	1342	1146	1153	1160	1163	-1.22
rat99	99	49	622	622	638	657	574	585	574	587	0
rd100	100	50	4012	4033	4052	4073	3392	3554	**3291**	**3327**	2.97
St70	70	35	302	311	302	311	302	306	**273**	**273**	9.60
Swiss42	42	21	469	515	469	515	458	458	**186**	**186**	59.38
U159	159	79	18841	19850	22879	23033	18491	18556	18841	21040	0
Ulysses16	16	8	2210	2232	2362	2362	1685	1685	1685	1959	0
Ulysses22	22	11	2489	2489	2498	2518	1902	1958	1902	2022	0

Table 3

Results of proposed GA Vs. GVNS on benchmark instances with $k = \left\lfloor \frac{3*n}{4} \right\rfloor$

Instance	*n*	*k*	*GVNS* (N_1)		*GVNS* (N_1)		*GVNS* $(N_1 + N_2)$		*Proposed GA*		*Dev. (%)*
			Best	*Worst*	*Best*	*Worst*	*Best*	*Worst*	*Best*	*Worst*	
bayg29	29	21	1028	1028	1055	1113	999	999	1028	1037	-2.90
bays29	29	21	1204	1204	1246	1246	1194	1194	1194	1198	0
berlin52	52	39	4555	4642	4860	4947	4213	4441	**4176**	**4253**	0.87
Bier127	127	95	54291	56616	54490	56815	50284	50903	50764	52990	-0.95
burma14	14	10	1693	1754	1643	1656	1642	1642	**1575**	**1575**	4.08
ch130	130	97	4403	4770	4719	5139	4213	4260	**4158**	**4226**	1.30
ch150	150	112	5255	5370	5311	5426	4637	4690	4720	4729	-1.78
d198	198	148	9795	10037	10045	10269	9363	9483	9363	9542	0
dantzig42	42	31	478	476	478	476	442	457	**439**	**439**	0.67
eil101	101	75	424	426	458	467	406	408	406	414	0
eil51	51	38	309	309	316	327	287	287	287	293	0
eil76	76	57	372	373	382	387	342	355	342	360	0
fri26	26	19	682	682	689	689	601	601	601	601	0
gr137	137	102	52699	53282	52699	53282	47465	48623	48623	50674	-2.43
gr17	17	12	951	951	951	951	951	951	951	951	0
gr21	21	15	1565	1582	1565	1582	1501	1501	1501	1501	0
gr24	24	18	852	852	852	852	844	844	844	886	0
gr48	48	36	3548	3627	3548	3627	3333	3352	**3234**	**3244**	2.97
gr96	96	72	41504	43679	41504	43679	31717	32965	**31095**	**31153**	1.96
hk48	48	36	7631	7883	7963	8165	7400	7411	**7357**	**7357**	0.58
kroA100	100	75	16288	16436	18256	18580	15740	15901	**15357**	**15357**	2.43
kroA150	150	112	20951	21457	21792	22475	18809	19223	**18295**	**19223**	2.73
kroA200	200	150	23898	25053	24262	25677	20135	20469	20135	21037	0
kroB100	100	75	16535	17282	18663	19238	15346	15493	15346	16363	0
kroB150	150	112	20951	22550	21876	22550	17349	17672	17349	17963	0
kroB200	200	150	25058	26124	25058	26124	20459	21272	21266	21725	-3.94
kroC100	100	75	17531	17739	18465	19047	14871	16738	**14835**	**14835**	0.24
kroD100	100	75	17079	17243	17721	17942	15630	15793	**14759**	**14770**	5.57
kroE100	100	75	16169	16604	17771	18506	15179	15236	15179	15882	0
Lin105	105	78	9444	9844	9469	9877	8999	9034	8999	9119	0
pr107	107	80	40731	41687	40959	42699	38579	39930	**37605**	**37605**	2.52
pr124	124	93	39978	40846	41102	43281	39203	39423	39423	41017	-0.56
pr136	136	102	78807	82446	78807	82446	70790	74732	70790	72819	0
pr144	144	108	48403	54366	48878	54823	44657	50147	**41703**	**41720**	6.61
pr152	152	114	59070	59527	59926	61777	56727	57075	**52393**	**52498**	7.64
Pr76	76	57	67800	68492	70925	77449	64990	65199	**64918**	**66329**	0.11
rat195	195	146	1787	1815	1890	2064	1713	1718	1713	1777	0
rat99	99	74	947	994	947	994	870	870	870	910	0
rd100	100	75	6247	6268	6247	6268	5175	6136	5175	5539	0
St70	70	52	482	487	494	510	477	486	**454**	**455**	4.82
Swiss42	42	31	773	918	773	918	760	760	333	368	56.18
U159	159	119	31296	31895	31296	31895	27612	28182	27612	27621	0
Ulysses16	16	12	3184	3264	3184	3264	3183	3183	3183	3385	0
Ulysses22	22	16	3110	3124	3240	3254	2968	2968	2968	3084	0

5. Conclusion

We have modelled, executed and tested a hybrid Genetic algorithm (GA) with composite mutation strategies for the TTSP. This algorithm successfully handles both subset collection and arrangement of the cities for the TTSP. To assess the performance, this algorithm tests 44 different sized benchmark instances with different truncation values and its results are compared with the results of existing GVNS with various neighbourhood strategies. Computational results exhibit the competence of the proposed GA over the existing GVNS algorithm. Furthermore, for many test cases, the proposed GA report improved results than the results of the GVNS algorithm. Our proposed GA approach being the first evolutionary algorithm for the TTSP, will act as a benchmark for future work on TTSP. Generating a good initial population for the TTSP and implementation of effective mutation operators using GA is still a challenging problem. Developing these things may be considered as a future scope of research.

References

Adasme, P., & Dehghan Firoozabadi, A. (2020). Degree-Constrained-Minimum Spanning Tree Problem. *Complexity*, *2020*.

Adasme, P., Soto, I., & Seguel, F. (2018, August). Finding degree constrained k-cardinality minimum spanning trees for wireless sensor networks. In *International Conference on Mobile Web and Intelligent Information Systems* (pp. 51-62). Springer, Cham.

Applegate, D. L., Bixby, R. E., Chvatal, V., & Cook, W. J. (2006). *The traveling salesman problem: a computational study*. Princeton university press.

Bahaabadi, M.R., Mohaymany, A.S., Babaei, M.: An Efficient crossover operator for travelling salesman problem. International Journal of Optimization in Civil Engineering 2(4), 607–619 (2012).

Baniasadi, P., Foumani, M., Smith-Miles, K., & Ejov, V. (2020). A transformation technique for the clustered generalized traveling salesman problem with applications to logistics. *European Journal of Operational Research*, *285*(2), 444-457.

Bhavani, V., & Murthy, M. S. (2006). Truncated M-travelling salesmen problem. *Opsearch*, *43*(2), 152-177.

Dasari, K. V., Pandiri, V., & Singh, A. (2021). Multi-start heuristics for the profitable tour problem. *Swarm and Evolutionary Computation*, *64*, 100897.

Demir, E., Huckle, K., Syntetos, A., Lahy, A., & Wilson, M. (2019). Vehicle routing problem: Past and future. In *Contemporary operations and logistics* (pp. 97-117). Palgrave Macmillan, Cham.

Elgesem, A. S., Skogen, E. S., Wang, X., & Fagerholt, K. (2018). A traveling salesman problem with pickups and deliveries and stochastic travel times: An application from chemical shipping. *European Journal of Operational Research*, *269*(3), 844-859.

Fatthi, W. N. A. W. A., Haris, M. H. M., & Kahtan, H. (2018, October). Application of travelling salesman problem for minimizing travel distance of a two-day trip in Kuala Lumpur via Go KL city bus. In *International Conference on Intelligent Computing & Optimization* (pp. 277-284). Springer, Cham.

Gensch, D. H. (1978). An industrial application of the traveling salesman's subtour problem. *Aiie Transactions*, *10*(4), 362-370.

Giardini, G., & Kalmár-Nagy, T. (2011). Genetic algorithm for combinatorial path planning: the subtour problem. *Mathematical Problems in Engineering*, *2011*.

Goldenberg, D. E. (1989). Genetic algorithms in search, optimization and machine learning. Addison-Wesley, Newyork.

Ha, Q. M., Deville, Y., Pham, Q. D., & Hà, M. H. (2020). A hybrid genetic algorithm for the traveling salesman problem with drone. *Journal of Heuristics*, *26*(2), 219-247.

Hariyadi, P. M., Nguyen, P. T., Iswanto, I., & Sudrajat, D. (2020). Traveling salesman problem solution using genetic algorithm. *Journal of Critical Reviews*, *7*(1), 56-61.

Hussain, A., Muhammad, Y. S., Nauman Sajid, M., Hussain, I., Mohamd Shoukry, A., & Gani, S. (2017). Genetic algorithm for traveling salesman problem with modified cycle crossover operator. *Computational intelligence and neuroscience*, *2017*.

Ibaraki, T. (1973). Algorithms for obtaining shortest paths visiting specified nodes. *Siam Review*, *15*(2), 309-317.

Khan, F. H., Khan, N., Inayatullah, S., & Nizami, S. T. (2009). Solving TSP problem by using genetic algorithm. *International Journal of Basic & Applied Sciences*, *9*(10), 79-88.

Kumar Thenepalle, J., & Singamsetty, P. (2018). Bi-criteria travelling salesman subtour problem with time threshold. *The European Physical Journal Plus*, *133*(3), 1-15.

Kumar, T. J., & Purusotham, S. (2017, November). An exact algorithm for k-cardinality degree constrained clustered minimum spanning tree problem. In *IOP Conference Series: Materials Science and Engineering* (Vol. 263, No. 4, p. 042112). IOP Publishing.

Laporte, G., Mercure, H., & Norbert, Y. (1984). Optimal tour planning with specified nodes. *RAIRO-Operations Research-Recherche Opérationnelle*, *18*(3), 203-210.

Larranaga, P., Kuijpers, C. M. H., Murga, R. H., Inza, I., & Dizdarevic, S. (1999). Genetic algorithms for the travelling salesman problem: A review of representations and operators. *Artificial Intelligence Review*, *13*(2), 129-170.

Liu, C., & Kroll, A. (2016). Performance impact of mutation operators of a subpopulation-based genetic algorithm for multi-robot task allocation problems. *SpringerPlus*, *5*(1), 1361.

Madani, A., Batta, R., & Karwan, M. (2021). The balancing traveling salesman problem: application to warehouse order picking. *Top*, *29*(2), 442-469.

Mittenthal, J., & Noon, C. E. (1992). An insert/delete heuristic for the travelling salesman subset-tour problem with one additional constraint. *Journal of the Operational Research Society*, *43*(3), 277-283.

Palhares, R. A., & Araújo, M. C. B. (2018, December). Vehicle routing: application of travelling salesman problem in a dairy. In *2018 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)* (pp. 1421-1425). IEEE.

Pandiri, V., & Singh, A. (2020). Two multi-start heuristics for the k-traveling salesman problem. *OPSEARCH*, *57*(4), 1164-1204.

Prakash, A., Balakrishna, U., & Thenepalle, J. (2022). An exact algorithm for constrained k-cardinality unbalanced assignment problem. *International Journal of Industrial Engineering Computations*, *13*(2), 267-276.

Riazi, A. (2019). Genetic algorithm and a double-chromosome implementation to the traveling salesman problem. *SN Applied Sciences*, *1*(11), 1397.

Saksena, J. P., & Kumar, S. (1966). The routing problem with "K" specified nodes. *Operations Research*, *14*(5), 909-913.

Sharma, S., & Jain, V. (2021, April). A Novel Approach for Solving TSP Problem Using Genetic Algorithm Problem. In *IOP Conference Series: Materials Science and Engineering* (Vol. 1116, No. 1, p. 012194). IOP Publishing.

Singamsetty, P., & Thenepalle, J. (2021). Designing optimal route for the distribution chain of a rural LPG delivery system. *International Journal of Industrial Engineering Computations*, *12*(2), 221-234.

Singamsetty, P., Thenepalle, J., & Ururuturu, B. (2021). Solving open travelling salesman subset-tour problem through a hybrid genetic algorithm. *Journal of Project Management*, *6*(4), 209-222.

Singh, R. K., Panchal, V. K., & Singh, B. K. (2018, August). A review on genetic algorithm and its applications. In *2018 Second International Conference on Green Computing and Internet of Things (ICGCIoT)* (pp. 376-380). IEEE.

Stanley, H. E., & Buldyrev, S. V., (2001). Statistical physics: The salesman and the tourist. *Nature*, *413 (6854),* 373+.

Stetsyuk, P. I. (2016). Problem statements for k-node shortest path and k-node shortest cycle in a complete graph. *Cybernetics and Systems Analysis*, *52*(1), 71-75.

Stodola, P., Otřísal, P., & Hasilová, K. (2022). Adaptive ant Colony optimization with node clustering applied to the travelling salesman problem. *Swarm and Evolutionary Computation*, *70*, 101056.

Venkatesh, P., Srivastava, G., & Singh, A. (2018). A general variable neighborhood search algorithm for the *k*-traveling salesman problem. *Procedia computer science*, *143*, 189-196.

Verweij, B., & Aardal, K. (2003). The merchant subtour problem. *Mathematical programming*, *94*(2-3), 295-322.

Westerlund, A., Göthe-Lundgren, M., & Larsson, T. (2006). A stabilized column generation scheme for the traveling salesman subtour problem. *Discrete Applied Mathematics*, *154*(15), 2212-2238.

Xu, H., Li, Q., Wang, J., Luo, G., Zhu, C., & Sun, W. (2018). An optimization routing algorithm for green communication in underground mines. *Sensors*, *18*(6), 1950.