British Journal of Mathematics & Computer Science
6(5): 422-434, 2015, Article no.BJMCS.2015.088
ISSN: 2231-0851

SCIENCEDOMAIN *international*
www.sciencedomain.org

New Variable Neighborhood Search Structure for Travelling Salesman Problems

A. Hande Erol Bingüler[1*] and Serol Bulkan[2]

[1]Department of Industrial Engineering, Institute for Graduate Studies In Pure and Applied Sciences, Marmara University, Göztepe 34722, Istanbul, Turkey.
[2]Department of Industrial Engineering, Faculty of Engineering, Marmara University, Göztepe 34722, Istanbul, Turkey.

Article Information

DOI: 10.9734/BJMCS/2015/14453
Editor(s):
(1) Farouk Yalaoui, Department of Industrial Systems Engineering, Troyes University of Technology, France.
(2) Tian-Xiao He, Department of Mathematics and Computer Science, Illinois Wesleyan University, USA.
Reviewers:
(1) Anonymous, Egypt.
(2) Stephen Akandwanaho, University of KwaZulu-Natal, South Africa.
(3) Ayman I.H. Srour, Facility of Information Science & Technology, National University of Malaysia, Malaysia.
Complete Peer review History: http://www.sciencedomain.org/review-history.php?iid=735&id=6&aid=7683

Method Article

Received: 30 September 2014
Accepted: 23 December 2014
Published: 09 January 2015

Abstract

In the traveling salesman problem, there are a collection of cities and travel cost between each pair of them. The aim is to find the minimum cost way of visiting all cities and returning to the starting point. This kind of problem is deceptive and one of the most intensely studied problems in computational mathematics. No effective solution method is known for the general case. Variable Neighborhood Search (VNS) is a recent metaheuristic for solving combinatorial and global optimization problems whose basic idea is systematic change of neighborhood within a local search. Its development has been rapid, with several dozen papers already published or to appear. Many extensions have been made, mainly to allow solution of large problem instances. In most of them, an effort has been made to keep the simplicity of the basic scheme. In this study, variable neighborhood search structure as a metaheuristic optimization technique and neighborhood approximation is developed. K-opt neighborhood structure is generated. This new structure's solvability in benchmark and symmetric traveling salesman problem instances is tested, and results are listed.

Keywords: Metaheuristic methods, neighborhood approximation, travelling salesman problem.

**Corresponding author: hande.erol@marmara.edu.tr;*

1 Introduction

The Traveling Salesman Problem (TSP) is a well-known and important combinatorial optimization problem. The goal is to find the shortest tour that visits each city in a given list exactly once and then returns to the starting city. Formally, the TSP can be stated as follows. The distances between n cities are stored in a distance matrix (D) with elements d_{ij} where *i, j= 1,…, n* and the diagonal elements d_{ii} are zero. A tour can be represented by a cyclic permutation Π of {*1, 2,…, n*} where Π(i) represents the city that follows city *i* on the tour.

The TSP is then the optimization problem to find a permutation Π that minimizes the length of the tour denoted by [1];

$$\sum_{i=1}^{n} d_{i\Pi(i)} \tag{1}$$

For this minimization task, the tour length of (n - 1)! permutation vectors have to be compared. This results in a problem which is very hard to solve and in fact known to be NP-complete. Solving TSPs is an important part of applications in many fields like vehicle routing, computer wiring, machine sequencing and scheduling, frequency assignment in communication networks. Applications in statistical data analysis include ordering and clustering objects As in Hubert and Baker's example, data analysis applications in psychology ranging from profile smoothing to finding an order in developmental data are proposed[2,3,4].

An optimization problem may be formulated as follows:

$$\min\{f(x) / x \in X, X \subseteq S\} \tag{2}$$

S, X, x and *f* are *solution space, feasible set, feasible solution* and real valued function, respectively. If *S* is a finite but large set a *combinatorial optimization* problem is defined. Most optimization problems are NP-hard and heuristic (suboptimal) solution methods are needed to solve them (at least for large instances or as an initial solution for some exact procedure). Metaheuristics, or general frameworks for building heuristics to solve problem (1), are usually based upon a basic idea, or analogy. Then, they are developed, extended in various directions and possibly hybridized. After complicated results, they use many parameters. This may enhance their efficiency but obscures the reasons of their success.

As TSP is known to be NP-hard, this means that no known algorithm is guaranteed to solve all TSP instances to optimality within reasonable execution time. So in addition to exact solution approaches, a number of heuristics and metaheuristics have been developed to solve problems approximately. Heuristics and metaheuristics trade optimality of the solutions that they output with execution times. They are used to find "good" quality solutions within reasonable execution times [5].

Finding the exact solution to a TSP with n cities requires checking (n-1)! possible tours. To evaluate all possible tours is infeasible for even small TSP instances. Held and Karp presented the dynamic programming to find the optimal tour in 1962 [1].

A different method, which can deal with larger instances, uses a relaxation of the linear programming problem and iteratively tightens the relaxation till a solution is found. This general method for solving linear programming problems with complex and large inequality systems is called cutting plane method and was introduced by Dantzig, Fulkerson, and Johnson in 1954. If no

further cutting planes can be found or the improvement in the objective function due to adding cuts gets very small, the problem is branched into two sub-problems which can be minimized separately. Branching which leads to a binary tree of sub-problems is used iteratively. Each sub-problem is either solved without further branching or is found to be irrelevant because its relaxed version already produces a longer path than a solution of another sub-problem. This method is called branch-and-cut which is a variation of the well known branch-and-bound procedure [1].

The development of computational methods to solve the TSP is an active field of research, and Applegate et. al. proposed a comprehensive review about solving TSP [6]. These methods can be classified into two broad categories, exact algorithms which are guaranteed to output optimal tours, and heuristics which generate good quality tours within reasonable execution time. The former category which includes cutting plane algorithms is shown in Table 1.

Table 1. Algorithms and references

Algorithms	References
Cutting plane algorithms	Dantzig et al. [7]; Grőtschel and Padberg [8]; Hong [9]
Branch and bound algorithms	Balas [10]; Held and Karp [11]; Lin [12]
Branch and cut algorithms	Hong [9]; Crowder and Padberg [13]; Grőtschel and Holland [14], Padberg and Rinaldi [15]

There are several construction heuristics such as the nearest neighborhood heuristic, the nearest, farthest, and cheapest insertion heuristics, Christofides' heuristic, as well as improvement heuristics such as Lin and Kernighan's local search [16], Fiechter'stabu search [17]; also tabu search proposed by Gendreau et al. [18]; Knox [19]; Potvin et al. [20]; Tsubakitani and Evans [21], Cerny's simulated annealing [22], Nguyen's genetic algorithms [23] and swarm algorithms of Goldbarg et al. [24] and Wang et al. [25,26].

Onder et al. proposed Artificial Bee Colony (ABC), Particle Swarm Optimization (PSO) supported GA techniques for finding the shortest route in condition of to visit every city one time but the starting city twice for TSP [27]. Nagata and Kobayashi used GA for TSP with edge assembly crossover (EAX) and found finding very high-quality solutions on instances with up to 200,000 cities [28]. Tasgetiren et al. [29] proposed iterated greedy algorithm with an Inver-Over operator to solve TSP. The proposed algorithm is applied to the well-known 14 TSP instances from TSPLIB and is competitive to the recent best performing algorithms. Gorkemli and Karaboga improved quick Artificial Bee Colony (qABC) algorithm as an improved version of ABC in which the onlooker bees behavior is modeled [30].

2 Methodology

2.1 Objectives

In this paper, exact algorithms, heuristic and meta-heuristic algorithms which are used to solve NP-hard problems are surveyed. These algorithms' solution approaches in problem instances are compared and new improved structure for TSP in the literature are also surveyed. The main objective is to solve TSP instances with an effective VNS structure based on *k-opt* neighborhoods and compare results to best known solutions to test the changes of the order of neighborhoods makes a significant difference.

2.2 Variable Neighborhood Search Method

Variable Neighborhood Search (VNS) is based upon a simple principle: systematic change of neighborhood within the search. Several dozen papers already published for VNS. Many extensions have been made and research papers mainly proposed to allow solution of large problem instances. In most of them, the aim is to keep the simplicity of the basic scheme.

VNS is a metaheuristic based on systematically changing of neighborhood set. Usual heuristic searches are based on transformations of solutions that determine one neighborhood structure on the solution space. VNS uses a series of neighborhood structure. The basic idea of the VNS is to change the neighborhood used when the local search is fascinated at a local minimum. During the past decade, this method has been successfully applied to a wide range of rich vehicle routing problems [31].

VNS is a stochastic local search method that is based on the systematic change of the neighborhood during the search. It has been shown to be a simple and effective method for solving single-objective optimization problems, including TSP and scheduling problems [30].

The concept of classical VNS algorithms employs a set of neighborhood search methods systematically to find the optimum or near-optimal solution. The base solution in VNS is compared with the neighboring solutions and updated during the search process. Most benchmark sets of single objective optimization problems are established using constraints with different levels [32]. Almost all methods in the literature treat and solve the instances above independently and separately. That is only one instance will be solved on each run which usually leads to limit the sharing of search information. However, from the observation on the search process, it is easy to find out that the info overlap among instances with adjacent levels of constraints [32,33].

This method includes the idea of neighborhood change systematically, both in descent to local minima and in escape from the valleys which contain them. VNS heavily relies upon three following observations [33]:

Fact 1. A local minimum for one neighborhood structure is not necessarily a local minimum for another neighborhood structure.
Fact 2. A global minimum is a local minimum for all possible neighborhood structures.

Fact 3. Local minima with respect to one or several neighborhoods are relatively close to each other.

This last observation is that a local optimum often provides some information about the global optimum. In this case, there may be several variables with the same value in both. However, it is not usually known which have the same value. Therefore, a VNS structure search different neighborhoods of a local optimum systematically [34].

2.2.1 Steps of variable neighborhood search method

Unlike most local search heuristics that uses only a simple neighboorhood structure, a VNS structure uses a finite set of pre-selected neighborhood structures denoted by N_k, for $k=1,\ldots,k_{max}$ and the set of solutions in the k^{th} neighborhood of x is denoted by $N_k(x)$. Neighborhoods N_k may be obtained by different local search approaches into a solution space S. A best solution (hopefully global minimum) x_{best} is a feasible solution where a minimum of the solutions in a neighborhood k is reached. The steps of a basic VNS structure are defined as in the Algorithm 1. [33].

Initialization:
- Determine the set of N_k, *for* $k=1,\ldots,k_{max}$, *that* will be used in VNS;
- Find an initial solution x by any construction heuristic and its objective function value $f(x)$, set $x_{best} \leftarrow x$, $f_{best} \leftarrow f(x)$; choose a stopping condition;

Repeat the following operations as long as the stopping conditions are not met:
(1) Set $k \leftarrow 1$;
(2) Repeat the below steps until $k=k_{max}$:
(a) ***Shaking.*** Generate a random solution point *x' and in the* k^{th} neighborhood of x;
(b) ***Local search.*** Find a solution point x'' as the local optimum applying some local search method with x' as initial solution;
(c) ***Check for improvements.*** If $f(x')$ is better than $f(x)$, set $f_{best} \leftarrow f(x'')$ and $x_{best} \leftarrow x''$ and $k=1$, otherwise set $k=k+1$
(or if $k=k_{max}$, set $k=1$); go to Step 1.

Algorithm 1. Steps of basic VNS algorithm [33].

In Step 1, x' is generated at random to avoid cycling (Successive N_k are often nested). In Step 2, if incumbent is changed then started over with N_1, otherwise continue search in N_{k+1} starting with the local optimum of N_k [33].

The main step can possibly be iterated until some other stopping condition is met (e.g. maximum number of iterations, maximum CPU time allowed, or maximum number of iterations between two improvements). Often successive neighborhoods N_k will be nested [35,36].

2.2.2 TSP and extensions

VNS is used for the TSP and its extensions. Another problem is asymmetric TSP that a generalized TSP in which distances between a pair of cities need not equal in the opposite direction [37]. Hansen and Mladenovic consider basic VNS for the euclidean TSP [38,39]. Burke et al. [40] apply guided VNS methods for the asymmetric TSP. In TSP, there is a collection of cities and travel cost between each pair of them. VNS includes structural and flexible changes of these pairs [41,42]. VNS for the Pickup and Delivery TSP is considered by Carrabs et al. [43]. Hu and Raidl study the effectiveness of neighborhood structures within a VNS approach for the generalized TSP [44]. Felipe et al. [45] use a VNS approach to solve a double TSP with multiple stacks. In the same year, a multi-start variant of VNS is applied by Mansini and Tocchella [46] to solve the travelling purchaser problem with budget constraints [35].

3 Variable Neighborhood Search Structure Proposed for TSP and Computational Study

3.1 Input Files

The symmetric and asymmetric TSP benchmark instances are taken from the TSP Library (http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/) are used in this study. The travel costs between nodes are Euclidean distance between the two corresponding nodes. Each benchmark problem is solved five times in order to have an average performance of the VNS structure.

The distances between n cities are stored in a distance matrix (D) with elements d_{ij} where $i, j= 1,\ldots, n$ and the diagonal elements d_{ii} are zero. A tour can be represented by a cyclic permutation Π of $\{1, 2,\ldots, n\}$ where $\Pi(i)$ represents the city that follows city i on the tour. The TSP is then the optimization problem to find a permutation Π that minimizes the length of the tour. The goal is to find the shortest tour that visits each city in a given list exactly once and then returns to the starting city.

There are six structures proposed in this paper. The proposed VNS algorithms are developed based on a neighborhood structures; k-opt algorithm. The VNS-I structure uses five Opt(fourOpt(threeOpt(twoOpt()))) as a neighborhood structure. Let S be a permutation given in the VNS before applying any operation and N be the number of cities.

3.2 Numerical Experiment

The simplest variant for shaking is to perform sequence of k moves. In this paper, the proposed VNS structures are developed based on four neighborhood structures; k-opt (2-opt, 3-opt, 4-opt, 5-opt) structures. For initial solution, Hill Climbing method is used. Neighborhood definition is *k*-opt where k_{max}=5, i.e. $N_k(x)$ is the set of solutions having *k* edges different from *x*. Local search method used in Step 2 is 2-opt.

This experiment is to use the six structures to solve all benchmark symmetric TSP instances. Each instance is solved by these structures five times with a random initial solution. These structures are terminated when any of the two following conditions is met;

1. The optimal solution is found,
2. The 100-*th* is reached. After five runs and 100 iterations (maximum iteration number as control parameter), and if there is no improvement in cost value, VNS process is ended. Then the computation time is recorded.

The result of the experiment in solution values is given in Table 2. In this table, the information in each column can be defined as; "*Problem name*" is symmetric TSP instance name that is used for the proposed VNS-I structure as the main structure, "*Best*" is the value of best solutions of application over five trial runs, "*Best known solutions*" is the value of best known solution, "Avg." is the average Central Processing Unit (CPU) times value taken from all five solutions.

Table 2. VNS-I application results for symmetric TSP instances

Problem name	Performance over best known solution (%)		Best solution	Best known solutions	Avg. (seconds)
	Min	Max			
ulysses16.tsp	0	0	**6859**	6859	19.719
ulysses22.tsp	0	0	**7013**	7013	25.047
fri26.tsp	0	0	**937**	937	27.078
bays29.tsp	0	0	**2020**	2020	30.109
bayg29.tsp	0	0	**1610**	1610	29.078
att48.tsp	0	0.006	**10628**	10628	42.266
hk48.tsp	0	0	**11461**	11461	42.375
berlin52.tsp	0	0	**7542**	7542	45.250
eil51.tsp	0	0	**426**	426	44.210
eil76.tsp	0	0.014	**538**	538	64.968
eil101.tsp	0.024	0.088	644	629	84.672
ch130.tsp	0.042	0.050	6368	6110	112.515
gr24.tsp	0	0	**1272**	1272	26.391
gr48.tsp	0	0	**5046**	5046	42.375
gr96.tsp	0.010	0.016	55774	55209	81.921
gr120.tsp	0.026	0.039	7127	6942	102.188
gr202.tsp	0.087	0.097	43668	40160	189.218
kroA100.tsp	0.016	0.031	21624	21282	85.047
kroB100.tsp	0.026	0.136	22715	22141	84.813

Table 2. VNS-I application results for symmetric TSP instances (continued)

Problem name	Performance over best known solution (%)		Best solution	Best known solutions	Avg. (seconds)
	Min	Max			
kroB150.tsp	0.098	0.101	28700	26130	138.656
kroB200.tsp	0.071	0.102	31540	29437	203.250
kroC100.tsp	0.003	0.004	20818	20749	84.328
kroD100.tsp	0.015	0.020	21621	21294	84.313
kroE100.tsp	0.016	0.049	22424	22068	84.218
lin105.tsp	0.015	0.020	14596	14379	89.250
pr76.tsp	0.004	0.010	108644	108159	65.906
pr107.tsp	0.040	0.051	46071	44303	89.657
pr124.tsp	0.013	0.015	59813	59030	111.016
pr136.tsp	0.049	0.058	101477	96772	118.235
pr144.tsp	0.022	0.014	59834	58537	126.547
pr152.tsp	0.063	0.075	78294	73682	134.813
pr226.tsp	0.101	0.111	88494	80369	215.703
rd100.tsp	0.014	0.072	8022	7910	85.063
tsp225.tsp	0.182	0.187	4630	3916	219.922
ts225.tsp	0.155	0.161	146326	126643	224.625
swiss42.tsp	0	0	1273	1273	38.547
rat99.tsp	0.028	0.044	1245	1211	82.328
rat195.tsp	0.161	0.285	2697	2323	179.703

Bold *values are the best known solutions obtained by VNS-I*

For symmetric TSP instances, the minimum and maximum errors as performance measures with respect to the best known solutions for test problem sets are listed. Symmetric TSP instances are solved and best solutions of five runs, the best known solutions and average CPU times (in seconds) are shown in Table 2. The proposed VNS-I heuristics solved eleven of twelve problem instances optimally. The last column reports the average of five runs' CPU times of the VNS-I heuristics. Because of the random choices made in the VNS heuristic, it is not deterministic. It is therefore executed five times on each instance.

VNS-I structure is then modified to five different variants for TSP instances.The other VNS structures are as follows:

- VNS-II structure uses threeOpt(fourOpt(fiveOpt(twoOpt()))),
- VNS-III structure uses threeOpt(fiveOpt(fourOpt(twoOpt()))),
- VNS-IV structure uses fourOpt(fiveOpt(threeOpt(twoOpt()))),
- VNS-V structure uses fourOpt(threeOpt(fiveOpt(twoOpt()))),
- VNS-VI structure uses threeOpt(fourOpt(twoOpt(fiveOpt()))).

Table 3 shows the solution values found by the proposed structures on thirty three symmetric TSP instances and Table 4 shows the solution values found by the proposed structures on seven asymmetric TSP instances. In these tables, the information in each column can be defined as follows:

- "VNS variants" is the name of each structure used to solve the problem.
- "Best" is the value of the best found solution over five trial runs.
- "Avg." is the average CPU times taken from all five solutions.
- The best found solution value is marked an asterisk if it is the best in the comparison.
- The best found solution value is bold if it is the optimal solution value.

Table 3. Solution values found by the proposed structures on the benchmark instances

Problem name	VNS-I		VNS-II		VNS-III		VNS-IV		VNS-V		VNS-VI		Best Known solution
	Best	CPU times (s)	Best	CPU times (s)	Best	CPU times (s)	Best	CPU times (s)	Best	CPU times (s)	Best	CPU times (s)	
ulysses16.tsp	**6859**	20	**6859**	20	**6859**	20	**6859**	20	**6859**	20	**6859**	20	6859
ulysses22.tsp	**7013**	25	**7013**	26	**7013**	26	**7013**	26	**7013**	26	**7013**	25	7013
fri26.tsp	**937**	27	**937**	27	**937**	27	**937**	26	**937**	27	**937**	26	937
bays29.tsp	**2020**	30	**2020**	30	**2020**	30	**2020**	20	**2020**	30	**2020**	31	2020
bayg29.tsp	**1610**	29	**1610**	30	**1610**	30	**1610**	30	**1610**	30	**1610**	31	1610
att48.tsp	**10628**	42	**10628**	43	**10628**	39	**10628**	40	**10628**	39	**10628**	40	10628
hk48.tsp	**11461**	42	**11461**	43	**11461**	43	**11461**	43	**11461**	43	**11461**	43	11461
berlin52.tsp	**7542**	45	**7542**	46	**7542**	45	**7542**	45	**7542**	45	**7542**	46	7542
eil51.tsp	**426**	44	**426**	45	428	45	**426**	45	**426**	45	428	45	426
eil76.tsp	**538**	65	540	65	541	65	548	68	549	70	541	65	538
eil101.tsp	*644*	85	648	85	663	88	655	89	659	92	663	89	629
ch130.tsp	*6368*	113	6401	113	6682	118	6472	112	6438	111	6682	117	6110
gr24.tsp	**1272**	26	**1272**	27	**1272**	27	**1272**	26	**1272**	26	**1272**	27	1272
gr48.tsp	**5046**	42	**5046**	43	**5046**	43	**5046**	43	**5046**	42	**5046**	43	5046
gr96.tsp	*55774*	82	56178	81	55984	81	56066	80	57078	81	58675	82	55209
gr120.tsp	*7127*	102	7255	109	7430	102	7394	102	7189	104	7255	109	6942
gr202.tsp	43668	189	44207	191	43705	192	43187	192	*42590*	192	43705	193	40160
kroA100.tsp	21624	85	22002	85	21603	86	*21398*	85	21741	86	21603	87	21282
kroB100.tsp	*22715*	85	22942	85	24020	85	22954	85	23109	85	24020	86	22141
kroB150.tsp	28700	139	*27672*	138	28236	138	27897	137	27724	137	28236	139	26130
kroB200.tsp	*31540*	203	33793	201	32420	205	33196	206	33002	204	32420	207	29437
kroC100.tsp	*20818*	84	22792	84	21202	86	21188	85	21667	86	21202	86	20749
kroD100.tsp	*21621*	84	21907	84	21949	85	22142	85	22146	86	21949	87	21294
kroE100.tsp	22424	84	*22340*	84	22481	85	22655	85	22597	85	22481	85	22068
lin105.tsp	14596	89	14683	89	14681	90	*14442*	89	14830	89	14681	90	14379
pr76.tsp	108644	66	109085	65	109411	65	*108444*	65	109164	66	109411	64	108159
pr107.tsp	46071	90	45244	90	45806	89	45821	90	*45211*	91	45806	90	44303
pr124.tsp	59813	111	60611	112	61365	112	45389	90	*44945*	90	61365	111	59030

Table 3. Solution values found by the proposed structures on the benchmark instances (continued)

Problem name	VNS-I		VNS-II		VNS-III		VNS-IV		VNS-V		VNS-VI		Best Known solution
	Best	CPU times (s)	Best	CPU times (s)	Best	CPU times (s)	Best	CPU times (s)	Best	CPU times (s)	Best	CPU times (s)	
pr136.tsp	*101477*	118	102782	120	102977	119	106438	120	119813	120	102977	120	96772
pr144.tsp	59834	127	*59158*	126	60315	127	60771	126	59921	126	60315	126	58537
pr152.tsp	78294	135	77290	137	*74961*	136	77825	136	76365	137	*74961*	137	73682
pr226.tsp	88494	216	90365	218	92088	215	89344	215	*87615*	213	92088	215	80369
rd100.tsp	*8022*	85	8197	85	8222	86	8350	89	8166	86	8222	86	7910

***Bold** values are the best known values.*

Table 4. Solution values found by the proposed structures on asymmetric TSP benchmark instances

Problem name	VNS-I		VNS-II		VNS-III		VNS-IV		VNS-V		VNS-VI		Best Known solution
	Best	CPU times (s)	Best	CPU times (s)	Best	CPU times (s)	Best	CPU times (s)	Best	CPU times (s)	Best	CPU times (s)	
ftv70.atsp	1968	*63*	1981	61	1975	65	1964	63	*1962*	68	1975	67	**1950**
ftv170.atsp	2815	61	**2755**	60	2815	60	3058	60	2893	61	2815	62	**2755**
ftv160.atsp	**2683**	60	2738	61	2832	60	2932	60	**2683**	61	2848	62	**2683**
kro124p.atsp	44230	85	46126	84	41443	84	37962	88	*36352*	89	41443	86	**36230**
rbg323.atsp	*1330*	330	1334	301	1333	311	1332	322	*1330*	328	1333	334	**1326**
rbg358.atsp	*1165*	388	1170	383	1168	384	1167	387	1169	384	1168	387	**1163**
p43.atsp	**5620**	42	5622	42	5625	43	5623	43	5621	43	5621	42	**5620**

***Bold** values are the best known values.*

According to the results from Table 4, VNS-II can find best known solution for ftv170, and VNS-I can find best known solution for ftv160 and p43. Thus, based on the data in Table 4, VNS-II seems to be the best structure since it returns the optimal solution in ftv170 and also returns nearly the best found solution on ftv70, ftv170, kro124p, rbg323.tsp in VNSV

Even though VNSI results are closer to best known solutions on symmetric TSP instances compared to the other five VNS structures, pairwise t-test does not show a significant difference between the best results as shown in Table 5.

Even though VNSII and VNSV results are closer to best known solutions on asymmetric TSP instances compared to the other five VNS structures, pairwise t-test does not show a significant difference between the best results as shown in Table 5.

Table 5. Statistical analyses for proposed structures on TSP benchmark instances

Structures	t-value (symmetric TSP benchmark instances)	t-value (asymmetric TSP benchmark instances)
VNSI-VNSII	0,048*	0,176**
VNSI-VNSIII	0,085*	0,194**
VNSI-VNSIV	0,347*	0,199**
VNSI-VNSV	0,440*	0,181**
VNSI-VNSVI	0,052*	0,195**
VNSII-VNSIII	0,048*	0,186**
VNSII-VNSIV	0,203*	0,194**
VNSII-VNSV	0,441*	0,180**
VNSII-VNSVI	0,260*	0,187**
VNSIII-VNSIV	0,205*	0,204**
VNSIII-VNSV	0,432*	0,174**
VNSIII-VNSVI	0,179*	0,255**
VNSIV-VNSV	0,238*	0,121**
VNSIV-VNSVI	0,169*	0,203**
VNSV-VNSVI	0,392*	0,173**

** < t value(0,05;32)= 2,037 for symmetric problems;*
*** < t value(0,05;6)= 2,247 for asymmetric problems.*

4 Conclusion

This paper examines new heuristic methods for getting an approximate solution of the TSP. A neighborhood structure is defined on the solution space, and used to develop a local search procedure. In this paper, we propose VNS structure for solving the TSP instances and examine the performance of our approach based on solution quality and execution time. Six variations of VNS are implemented and tested on 40 benchmark problems. From the application, VNS-I and VNS-V performs best in terms of solution quality for asymmetric TSP instance. VNS-I structure performs best for symmetric TSP instances.

An important extension of this work could be to develop a new solution technique with a hybrid VNS structure. As a future research, the neighborhood structures used in the local search stage of the VNS structure will be improved in order to enhance the search performance and all symmetric and asymmetric TSP problem sets will be solved by this structure.

Competing Interests

Authors have declared that no competing interests exist.

References

[1] Hahsler M, Hornik K. TSP-Infrastructure for the traveling salesperson problem. Journal of Statistical Software. 2007;23(2):1-21.

[2] Hubert LJ, Baker FB. Applications of combinatorial programming to data analysis: the traveling salesman and related problems. Psychometrika. 1978;43(1):81-91.

[3] Erol AH. A heuristic solution algorithm for the quadratic assignment problems. Master thesis, Marmara University, Institute for Graduate Studies in Pure and Applied Sciences, Turkey; 2010. Supervisor: Asst. Prof. Dr. Serol Bulkan.

[4] Erol AH, Bulkan S. New genetic algorithm for the travelling salesman problem. Proceedings of the 2012 International Conference on Industrial Engineering and Operations Management, Istanbul, Turkey, July 3 – 6, 2012.

[5] Basu S, Ghosh D. A review of the tabu search literature on travelling salesman problems. Working Paper Series, Indian Institute of Management, Ahmedabad, India, W.P. 2008;2008-10-01.

[6] Applegate DL, Bixby RE, Chvatal V, Cook WJ. The traveling salesman problem: a computational study. Princeton Series in Applied Mathematics. Princeton University Press; 2006. Princeton, NJ.

[7] Dantzig G, Fulkerson R, Johnson S. Solution of a large-scale traveling-salesman problem. Operations Research. 1954;2:393-410.

[8] Grotschel M, Padberg MW. Polyhedral theory. In Lawler et al. 1985;252-305.

[9] Hong S. A linear programming approach for the traveling salesman problem. Ph.D. thesis. Johns Hopkins University; 1972, Baltimore, MA.

[10] Balas E. An additive algorithm for solving linear programs with zero-one variables. Operations Research. 1965;13:517-546.

[11] Held M, Karp RM. The traveling salesman problem and minimum spanning trees. Operations Res. 1970;18:1138-1162.

[12] Lin S. Computer solutions of the traveling salesman problem. The Bell System Technical Journal. 1965;44:2245-2269.

[13] Crowder H, Padberg MW. Solving large-scale symmetric traveling salesman problems to optimality. Management Science. 1980;26:495-509.

[14] Grotschel M, Holland MO. Solution of large-scale symmetric traveling salesman problems. Mathematical Programming. 1991;51:141-202.

[15] Padberg M, Rinaldi G. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. SIAM Review. 1991;33:60-100.

[16] Lin S, Kernighan BW. An effective heuristic algorithm for the traveling-salesman problem. Operations Research. 1973;21:498-516.

[17] Fiechter CN. A parallel tabu search algorithm for large scale traveling salesman problems. Working Paper 90/1 Department of Mathematics, Ecole Polytechnique Federale de Lausanne; 1990, Switzerland.

[18] Gendreau M, Hertz A, Laporte G. A tabu search heuristic for the vehicle routing problem. Management Science. 1994;40:1276-1290.

[19] Knox J. Tabu search performance on the symmetric traveling salesman problem. Computers & Operations Research. 1994;21:867-876.

[20] Potvin JT, Kervahut T, Garcia BL, Rousseau JM. The vehicle routing problem with time windows Part I: Tabu Search. INFORMS Journal on Computing. 1996;8:158-164.

[21] Tsubakitani S, Evans JR. Optimizing tabu list size for the traveling salesman problem, Computers & Operations Research. 1998;25:91-97.

[22] Cerny V. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. Journal of Optimisation Theory and Application. 1985;45:41-51.

[23] Nguyen DH. Hybrid genetic algorithms for combinatorial optimization. Ph.D. Thesis. Department of Systems Engineering. University of Miyazaki, Japan; 2004.

[24] Goldbarg EFG, Goldbarg MC, Souza GR. Particle swarm optimization algorithm for the traveling salesman problem. In Greco. 2008;202-224.

[25] Wang KP, Huang L, Zhou CG, Pang W. Particle swarm optimization for traveling salesman problem. International Conference on Machine Learning and Cybernetics. 2003;1583-1585.

[26] Ghosh D, Basu S. Diversified local search for the traveling salesman problem. Working Paper Series, Indian Institute of Management, Ahmedabad, India, W.P. 2011;2011-01-03.

[27] Onder E, Ozdemir M, Yıldırım BF. Combinatorial optimization using artificial bee colony algorithm and particle swarm optimization supported genetic algorithm, Kafkas University Journal of Economics and Administrative Sciences Faculty. 2013;4(6). ISSN: 1309-4289.

[28] Nagata Y, Kobayashi S. A powerful genetic algorithm using edge assembly crossover for the traveling salesman problem. INFORMS Journal on Computing. 2013;25(2):346-363.

[29] Tasgetiren MF, Buyukdagli O, Kızılay D, Karabulut K. A Populated iterated greedy algorithm with Inver-over operator for traveling salesman problem. Swarm, Evolutionary, and Memetic Computing Lecture Notes in Computer Science. 2013;8297:1-12.

[30] Gorkemli B, Karaboga D. Quick combinatorial Artificial Bee Colony -qCABC- Optimization Algorithm for TSP. ISCIM - International Symposium on Computing in Informatics and Mathematics. 2013;97-101.

[31] Wen M, Krapper E, Larsen J, Stidsen TK. A multilevel variable neighborhood search heuristic for a practical vehicle routing and driver scheduling problem. Networks. 2011;58:311-322.

[32] Arroyo JEC, Ottoni RS, Santos A. Multi-objective Variable Neighborhood Search Algorithms for a Just-in-Time Single Machine Scheduling Problem. 2011;978-1-4577-1676-8/11@ 2011 IEEE.

[33] Hansen P, Mladenović N. Variable neighborhood search: Principles and applications. European Journal of Operational Research. 2001;130(3):449-467.

[34] Hansen P, Mladenović N, Pérez JAM. Variable neighbourhood search: methods and applications. Ann. Oper Res. 2010;175:367-407.

[35] Mladenović, N., Hansen, P. Variable neighborhood search. Computers & Operations Research. 1997;24(1):1097-1100.

[36] Mladenović N, Urošević D. VNS for the TSP and its variants. BALCOR. 2011, Thessaloniki, Greece.

[37] Piriyaniti I, Pongchairerks P. Novel VNS Algorithms on Asymmetric Traveling Salesman Problems. Second International Conference on Computer and Network Technology; 2010. DOI: 10.1109/ICCNT.2010.75.

[38] Hansen P, Mladenović N. An introduction to variable neighborhood search. In S. Voss et al. (Eds.), Metaheuristics, advances, trends in local search paradigms for optimization. 1999;433-458. Amsterdam: Kluwer.

[39] Hansen P, Mladenović N. First improvement may be better than best improvement: An empirical study. Discrete Applied Mathematics. 2006;154:802–817.

[40] Burke EK, Cowling P, Keuthen R. Effective local and guided variable neighborhood search methods for the asymmetric travelling salesman problem. In Lecture Notes in Computer Science. 2001;2037, 203–212. Berlin: Springer.

[41] Available: http://www.math.uwaterloo.ca/tsp/problem/, Accessed date: 11/01/2014

[42] Available: http://www.cse.hcmut.edu.vn/~dtanh/download/Appendix_B_LG.ppt, Accessed date: 11/01/2014

[43] Carrabs F, Cordeau JF, Laporte G. Variable neighbourhood search for the pickup and delivery traveling salesman problem with LIFO loading. INFORMS Journal on Computing. 2007;19(4):618–632.

[44] Hu B, Raidl GR. Effective neighborhood structures for the generalized traveling salesman problem. In Lecture Notes in Computer Science. 2008;4972, 36–47. Berlin: Springer.

[45] Felipe Á, Ortuño MT, Tirado G. The double traveling salesman problem with multiple stacks: a variable neighborhood search approach. Computers and Operations Research 2009;36(11):2983–2993.

[46] Mansini R, Tocchella B. The traveling purchaser problem with budget constraint. Computers and Operations Research. 2009;*36*(7):2263-2274.

Peer-review history:
The peer review history for this paper can be accessed here (Please copy paste the total link in your browser address bar)
www.sciencedomain.org/review-history.php?iid=735&id=6&aid=7683