Applied Soft Computing 68 (2018) 83–91

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Improving variable neighborhood search to solve the traveling salesman problem

Samrat Hore [a], Aditya Chatterjee [b,*], Anup Dewanji [c]

[a] Department of Statistics, Tripura University, India
[b] Department of Statistics, University of Calcutta, India
[c] Applied Statistics Unit, Indian Statistical Institute, Kolkata, India

ARTICLE INFO

Article history:
Received 22 September 2017
Received in revised form 23 February 2018
Accepted 28 March 2018
Available online 1 April 2018

Keywords:
Neighborhood structure
Hybrid VNS
Stopping rule
Near-optimal solution
Cost matrix

ABSTRACT

The traveling salesman problem (TSP) is one of the classical combinatorial optimization problems and has wide application in various fields of science and technology. In the present paper, we propose a new algorithm for solving the TSP that uses the variable neighborhood search (VNS) algorithm coupled with a stochastic approach for finding the optimal solution. Such neighborhood search with various other local search algorithms, named as VNS-1 and VNS-2, has been reported in the literature. The proposed algorithm is compared in detail with these algorithms, in the light of two benchmark TSP problems (one being symmetric while the other is asymmetric) suggested in the TSPLIB dataset in programming language R, along with two asymmetric problems obtained through simulation experiment. The present algorithm has been found to perform better than the conventional algorithms implemented in R for solving TSP's, and also, on an average, found to be more effective than the VNS-1 and the VNS-2 algorithms. The performance of the proposed algorithm has also been tested on 60 benchmark symmetric TSPs from the TSPLIB dataset. Apart from solving the TSP, the flexibility of the proposed hybrid algorithm to solve some other optimization problems related to other disciplines has also been discussed.

1. Introduction

The TSP is a well-known, popular, and an extensively studied problem in the field of combinatorial optimization. It has immense applications in the field of engineering science like designing hardware devices and radio electronic devices in communications, in the architecture of computational networks, designing of microchips, DNA sequencing [1], data association, vehicle routing [2,3], data transmission in computer networks [4], image processing and pattern recognition [5], to quote a few. The concept of the TSP is very simple but yet it remains one of the most challenging problems in operations research. More succinctly, it is an optimization problem to find the shortest connectivity among all the nodes in a closed network by touching all the nodes exactly once. Although seems simple, the computer running time for an extensive search approach lies within a polynomial factor of $O(n!)$, where n is the number of nodes, thus making the problem most difficult, even for $n = 20$. In theoretical computer science, such kind of problem is known as the classical NP-complete problem [6], which means, for the worst-case, the running time for any algorithm for the TSP increases super-polynomially (or, perhaps exponentially) with the number of nodes [7]. The graph-theoretic method identifies the problem as finding the Hamiltonian cycle with the least weight for a given complete weighted graph [8].

Since 1950s, researchers are trying to develop various computer intensive algorithms to reach an optimal or near-optimal solution. Among these algorithms, branch and bound [9] has been adopted for solving the TSP with smaller scale. Apart from TSP with one salesman, one may consider the problem with multiple salesmen, departing from several starting cities and returning back to their respective starting cities. Such TSP is known as fixed destination multi-depot multiple traveling salesmen problem (MmTSP). Dynamic programming is one of the conventional methods for solving the TSP, both symmetric and asymmetric and also MmTSP, which starts with a minimum sub-tour and the optimal tour is reached by augmenting the city not yet in the sub-tour [10,11]. Particle swarm optimization [12,13] has been invoked to solve the TSP with a larger scale say, for $n = 51–76$. Such solution has been successfully applied for vehicle routing problems [2,14,15]. Another metaheuristic, known as ant colony optimization has been applied to solve such MmTSP [16,17]. This algorithm mimics the behavior of the real ants in search of food, where every ant leaves an amount of pheromone on the path it passes and chooses the path with more pheromone left on it from the previous ants. Since more ants pass the shorter roots, gradually, more and more ants choose the shorter

* Corresponding author.

https://doi.org/10.1016/j.asoc.2018.03.048
1568-4946/

root and as a result, the colony finds the best way to get to the food. Genetic algorithms are an alternative metaheuristic optimization algorithm which encode a potential solution to a specific problem on a simple chromosome-like data structure, and apply recombination operators to these structures in such a way as to preserve critical information [18–20]. Simulated annealing [21,22] is one of the widely used algorithms to achieve a global optimal solution for computation intensive combinatorial optimization problems in recent days. Simulated annealing is a method for improving local optimization, and it needs less memory space. However, to find a better solution, simulated annealing algorithm generally requires longer computational time compared to other heuristics of discrete optimization, a brief review of which is discussed in the next few paragraphs.

The simplest of its kind is the nearest neighbor algorithm (NNA) [23]. This algorithm starts with a sub-tour, or path, containing a subset of randomly chosen cities from the list of n cities and then adds the nearest city that is yet to visit. The algorithm stops when all cities are included in the tour. An extension to this algorithm is to repeat it with each city as the starting point and then return the best tour found, which is called the repetitive nearest neighbor algorithm (RNNA) [23]. Another approach is known as Insertion Algorithm (IA) [23], where it starts with a sub-tour consisting of two arbitrary cities (say, i and j) and then chooses in each step a city k not yet in the sub-tour. This city is inserted between the two cities i and j, such that the insertion cost (i.e., the net increase in the cost of the tour) given by $d(i,k)+d(k,j)-d(i,j)$ is minimum. This insertion process is repeated with the current sub-tour $i \to k \to j$ and a fourth city is inserted between either i and k, or k and j, with the objective of minimizing the net increase in the cost of the next sub-tour. As before, the algorithm stops when all the cities are included in the tour. According to Hahsler and Hornik [24], four variants of IA are available, depending upon the way the city is to be inserted next is chosen, which are (1) Nearest Insertion (NI), (2) Farthest Insertion (FI), (3) Cheapest Insertion (CI) and (4) Arbitrary Insertion (AI), among which the first two are most popular. For the comparative analysis of the proposed algorithm, we have considered only the first two.

Some algorithms based on local improvements of the existing tour through simple tour modifications are also available in the literature. Such algorithms are specified in terms of a class of operations (exchanges or moves) that can be used to convert one tour into another by reducing the tour length until a tour is reached for which no such operation yields an improvement (i.e., a locally optimal tour). Among such simple local search algorithms, the most famous are 2-*Opt* and 3-*Opt* algorithms. The 2-*Opt* algorithm was developed by Croes [25], although it was earlier proposed by Flood [26]. The procedure deletes two edges of the tour, thus breaking it into two paths, and then reconnects those paths in the other possible way. A modified 2-*Opt* algorithm named as the *Quicker* 2-*Opt* was introduced in [27], by reducing search space for a selected pair of links, instead of deleting and reconnecting all possible pairs of links in the 2-*Opt* algorithm. In the 3-*Opt* algorithm [28], the exchange replaces up to three edges of the current tour. The idea of 2-*Opt* and 3-*Opt* algorithms may be extended to k-*Opt* algorithms where the exchange replaces up to k edges of the current tour. This extension by Lin and Kernighan [29] does not use a fixed value for k for its k-*Opt* moves but tries to find the best choice of k for each move.

To obtain a locally optimal solution, the search operation is confined among the neighbors of an initial solution, called neighborhood, where the neighbors are iteratively obtained through a systematic change of the nodes of the initial tour. Through this change, and by means of local search, an algorithm has been developed that leads to one kind of metaheuristic algorithm, known as variable neighborhood search (VNS) algorithm [27,30]. The VNS-1 and the VNS-2 algorithms are recognized as such neighborhood search approaches, where the local searches (i.e., improvements over the present solution) are carried out through 2-*Opt* and quicker 2-*Opt* algorithms, respectively.

The objective of the present work is to suggest some modifications over the variable neighborhood search (VNS) algorithm, suggested by Hansen and Mladenović [27,30] to solve TSP, with regard to the following three aspects:

- a reasonably good and computationally efficient choice of the initial tour,
- a clever choice of neighborhood construction,
- an incorporation of a stochastic approach, close to the more popular annealing schedule, to avoid ending at a local optimum and achieving the global optimal solution.

Through the above three aspects, the present hybrid VNS algorithm aims to deal with some imprecision and uncertainty in order to achieve near-optimal, robust and low-cost solution. These contributions have been attempted and highlighted by demonstrating the distributional properties of the final tour lengths obtained by the proposed algorithm and various other algorithms. The application aspect of the proposed algorithm is discussed through its ability to handle moderately large TSP with little computational complexity and at the same time its flexibility to take care of other areas of discrete optimization with equal competence. Such an algorithm has been suggested by Hore et al. [31,32] to find a near-optimal or optimal allocation design with different treatments to several experimental units with the known covariate(s). The proposed algorithm is similar to the algorithm suggested in the context of the design issue addressed there, with the necessary modifications of enlargement of the search space.

The paper has been structured as follows. In Section 2, a brief review of the VNS algorithm has been done. In order to obtain a near-optimal tour, a moderately acceptable initial solution is needed. A procedure to obtain such initial solution is discussed in Section 3. Section 4 describes the proposed algorithm, based on the initial tour. A detailed analysis of two TSPs, one each for the symmetric and asymmetric case, from the TSPLIB dataset of R, along with two more asymmetric TSPs generated through simulation experiments, are considered in Section 5. These examples establish the efficacy of the proposed algorithm in comparison to existing algorithms, as available and implemented in R and other VNS algorithms. The performance of the present algorithm is also judged in light of 60 benchmark symmetric TSPs from the TSPLIB dataset. The datasets and corresponding optimum solutions are available at [35]. To demonstrate the flexibility of the proposed hybrid VNS algorithm for solving combinatorial optimization problems, some applications other than the TSP are described in Section 6. The paper ends with some concluding remarks and scopes for further research.

2. Review of the VNS algorithm

We start with a brief review of the variable neighborhood search (VNS) algorithm with special emphasis on VNS-1 and VNS-2, as proposed in [27,30]. Consider the situation where a finite number $k_{\max}$ of pre-selected ordered neighborhood structures is available and let the set of neighbors in the kth neighborhood of the tour x be denoted by $\mathcal{N}_k(x)$, for $k = 1, 2, \ldots, k_{\max}$. Note that the neighborhood search is permitted up to $k_{\max}$ number of times. The basic VNS may be summarized as follows and for details we refer to [27,30].

Initialization: Find an initial solution x; select the set of neighborhood structures $\mathcal{N}_k(x)$, $k = 1, 2, \ldots, k_{\max}$, that will be used in the search; choose astopping condition;
Repeat the following until the stopping condition is met:
(1) *Set* $k \longleftarrow 1$;
(2) *Until* $k = k_{\max}$, *repeat the following steps:*
(a) *Shaking.* Generate a point x' at random from the kth neighborhood of x ($x' \in \mathcal{N}_k(x)$).
(b) *Local Search.* Apply some local search method with x' as the initial solution; denote the so obtained local optimum by x''.
(c) *Move or Not.* If this local optimum is better than the incumbent, move there ($x \longleftarrow x''$), and
continue the search with $\mathcal{N}_1(x'')$ for $k \longleftarrow 1$; otherwise set $k \longleftarrow k+1$.
Stopping Condition: Stop if maximum number of iterations is achieved.

The stopping condition may be based on the maximum amount of CPU time allowed, or the maximum number of iterations permitted, or the number of iterations between two significant improvements, and so on. In the simulation study and real life experiments, we have considered the maximum number of iterations as the stopping condition. The most popular version of local search algorithm for solving TSP is the 2-*opt* algorithm. The basic VNS rules using 2-*Opt* and *Quicker* 2-*Opt* as local search method are known as VNS-1 and VNS-2, respectively. Thus, the only difference between VNS-1 and VNS-2 lies in the choice of the local search method, although the definition of neighborhood for both VNS-1 and VNS-2 are the same. Here $\mathcal{N}_k(x)$ is the kth neighborhood of a tour x, for $k = 1, 2, \ldots, n-1$, which is obtained by randomly exchanging $(k+1)$ elements of x, i.e., having $(k+1)$ edges different from x. In the simulation study and real life examples, we have observed that VNS-2 consistently outperforms VNS-1 with varying extent.

There is scope for improvement in both VNS-1 and VNS-2 with respect to three important aspects, namely, (1) determination of the initial tour, (2) construction of neighborhood structure including the method of local search and (3) developing an efficient stopping rule. In the next two sections, we suggest some improvements in these three aspects and, in Section 5, illustrate them to find optimal or near-optimal tour by means of some real-life examples and simulation studies. Apart from solving the TSP, the hybrid VNS algorithm has also been applied in the fields of optimal design of experiments [31,32], wireless sensor networks [33] and reliability theory and survival analysis [34], and such applications are briefly mentioned in Section 6.

3. The initial tour

We propose a greedy algorithm to obtain an initial solution, which is intuitively very simple and quite similar to NNA. In NNA, a tour on the basis of a subset of randomly chosen cities adds one city which is not included in the tour and is closest to the last city of the existing tour. Here we suggest that the tour should start from that particular sub-tour with two cities which has the least distance among all such possible sub-tours. Although such algorithm usually does not give the global optimum solution, we may consider it as the initial solution [36].

Let $I = \{1, 2, \ldots, n\}$ be the set of n distinct cities. The cost function between any two cities r and s is given by $d(r, s)$ with $d(r, r) = 0$, for $r, s \in I$. For symmetric TSP, $d(r, s) = d(s, r)$, forall $r, s \in I$. Suppose $d(i, j) = \min_{r,s}\{d(r, s)\,;\, r, s \in I\}$ and assume this to be unique; if not, one can choose the minimum randomly. The tour should start from i to j, denoted by $\{i, j\}$. Let k be the nearest city with the tour $\{i, j\}$, in terms of the minimum cost $d(i, j, k)$ or $d(k, i, j)$ depending upon whichever is minimum, where $d(i, j, k) = \min_l\{d(i, j) + d(j, l),\ l \in I,\ l \neq i, j\}$ or $d(k, i, j) = \min_l\{d(l, i) + d(i, j),\ l \in I,\ l \neq i, j\}$, respectively. Then the city k is augmented with the already existing tour $\{i, j\}$ either at the beginning or at the end, and the augmented tour is denoted by either of $\{k, i, j\}$ or $\{i, j, k\}$, respectively. The procedure is continued till all the cities are included in the tour.

A good initial solution with less computational effort is the key to arrive at an optimal or near-optimal solution through any kind of metaheuristic algorithm. Otherwise, an inferior choice of the initial solution may mislead the entire optimization approach to a local optimum and hence an inferior final solution. We have proposed a simple method to obtain an initial solution with the minimum computational burden. According to Hansen and Mladenović [27,30], the most popular heuristic to solve TSP is 2-*opt* algorithm. Thus, it might be a good choice of initial solution, but needs extensive computation, whereas our suggested initial method requires lesser computational burden with no uncertainty. A comparative study between the suggested initial and the initial obtained from the 2-*opt* algorithm for moderately large number of cities with regard to their abilities to converge to the optimal or near-optimal solution considering various examples from TSPLIB dataset is reported in Section 5. Some graphical comparisons are also reported through Figs. 1–3. For obtaining the optimal or near-optimal solution with the suggested initial solution, an extension of the VNS algorithm through stochastic approach and named as hybrid VNS, has been described in details in the next section.

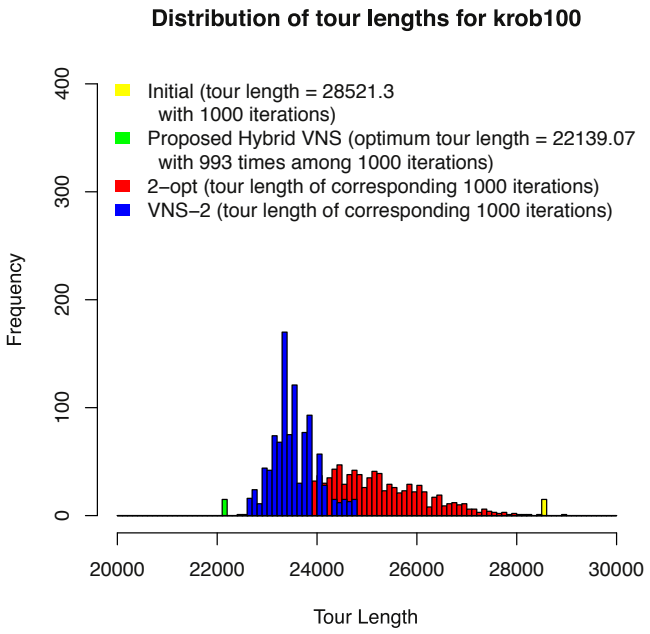

Fig. 1. Histogram of final tour lengths for *krob*100 data with regard to different algorithms.

4. The hybrid VNS algorithm

Once the initial tour is judiciously obtained, an iterative algorithm is proposed to arrive at the optimal or near-optimal tour. Consider a tour given by a permutation $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ of $\{1, 2, \ldots, n\}$, the labels of n cities. Let us define the first order neighborhood of a tour α, denoted by $\mathcal{N}_1(\alpha)$, as $\{\alpha_{(11)}, \alpha_{(12)}, \ldots, \alpha_{(1n)}\}$, where $\alpha_{(1i)}$ is the permutation obtained from α by exchanging the elements at the ith and $(i+1)$st locations, i.e. α_i and α_{i+1}, respectively, for $i = 1, 2, \ldots, n$, with $\alpha_{n+1} = \alpha_1$, keeping other elements unchanged. Thus,

$$\alpha_{(11)} = (\alpha_2, \alpha_1, \alpha_3, \ldots, \alpha_{n-1}, \alpha_n),$$
$$\alpha_{(12)} = (\alpha_1, \alpha_3, \alpha_2, \ldots, \alpha_{n-1}, \alpha_n),$$
$$\ldots\ldots\ldots\ldots\ldots$$
$$\alpha_{(1\ n-1)} = (\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_n, \alpha_{n-1}),$$
$$\alpha_{(1\ n)} = (\alpha_n, \alpha_2, \alpha_3, \ldots, \alpha_{n-1}, \alpha_1).$$

The neighborhood construction for obtaining the neighbors of a selected initial tour may be generalized on the basis of this location exchanging concepts in subsequent orders of neighborhood. The

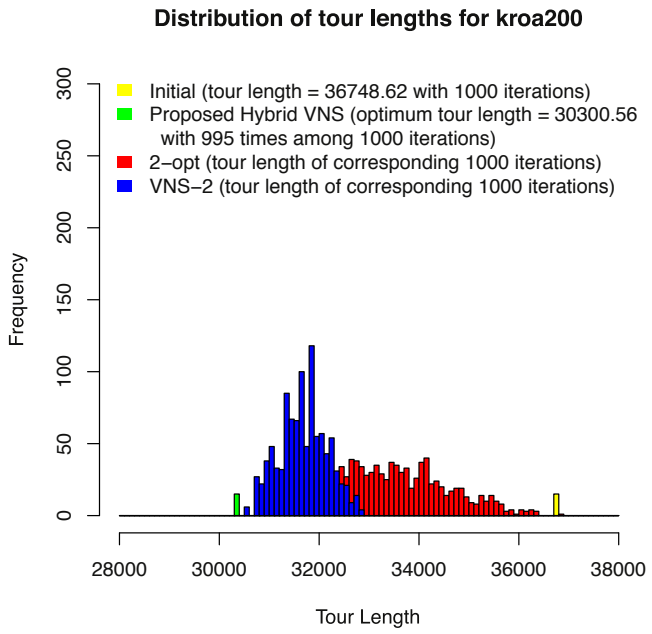

Fig. 2. Histogram of final tour lengths for *kroa*200 data with regard to different algorithms.

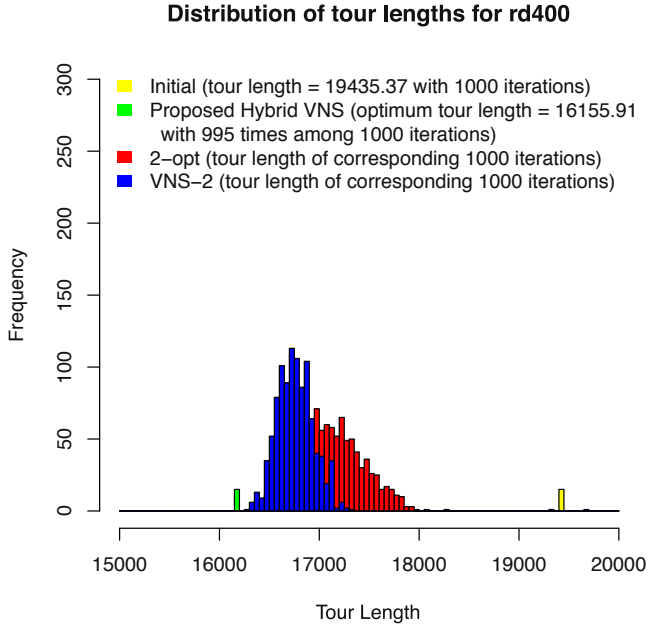

Fig. 3. Histogram of final tour lengths for *rd*400 data with regard to different algorithms.

kth order neighborhood of a tour α, denoted by $\mathcal{N}_k(\alpha)$, is constructed by exchanging α_i and $\alpha_{i+k\,(mod\,n)}$, for $i = 1, 2, \ldots, n$, while keeping other elements unchanged. Note that $\mathcal{N}_i(\alpha) = \mathcal{N}_{n-i}(\alpha)$, for $i = 1, 2, \ldots, n$, so that maximum value of k is $k_{\max} = [n/2]$. The cost of the tour corresponding to the permutation α is denoted by $V(\alpha)$, where $V(\alpha) = d(\alpha_1, \alpha_2) + d(\alpha_2, \alpha_3) + \cdots + d(\alpha_{n-1}, \alpha_n) + d(\alpha_n, \alpha_1)$. All the $n!$ possible tours are connected through this concept of neighborhood in the sense that any tour may be obtained from any another tour through a finite number of such exchange operations. The proposed algorithm for obtaining the optimum tour, starting from an initial tour, is described in the following steps along with explanation after each step.

Step 1: Start with the initial tour $\alpha^{(0)}$ with corresponding cost $V(\alpha^{(0)})$, say.

The initial tour is selected by the algorithm suggested in Section 3. The performance of the proposed algorithm depends upon a reasonable choice of the initial tour, failing which one may end up with one that is far from the optimal tour.

Step 2: Consider $k = 1$.

The exploration of neighborhoods starts with the first order neighbors of the initial tour.

Step 3: Consider all the kth order neighbors in $\mathcal{N}_k(\alpha^{(0)})$, and compute $V(\alpha')$, for all $\alpha' \in \mathcal{N}_k(\alpha^{(0)})$.

Note that, for each k, a $n-$dimensional vector represents a neighbor and there are n such different neighbors. The cost for each of them is computed.

Step 4: If min $\{V(\alpha'), \alpha' \in \mathcal{N}_k(\alpha^{(0)})\} < V(\alpha^{(0)})$, choose the next improved tour to be $\alpha^{(1)} = \arg\min\{V(\alpha'), \alpha' \in \mathcal{N}_k(\alpha^{(0)})\}$, and replace $\alpha^{(0)}$ by $\alpha^{(1)}$, to start search again from Step 2.

A better tour than the initial one is found from among the kth order neighbors and the algorithm starts afresh from Step 2, with the better tour replacing the initial one, to find a still better tour than the present one.

Step 5: Otherwise, set $k = k + 1$ and go to Step 3, till $k = [n/2]$.

If no better tour is found until the kth order, then extend the search domain to include the $(k+1)$st order neighbors of $\alpha^{(0)}$ and the search is continued till a better tour is obtained from among all the neighbors of the previous neighborhoods and including the present one.

Step 6: When $k > [n/2]$, check the stopping condition described in Step 7. If it is not satisfied, choose $\alpha^{(1)}$ randomly from $\alpha^{(0)} \cup \mathcal{N}_1(\alpha^{(0)}) \cup \cdots \cup \mathcal{N}_{[n/2]}(\alpha^{(0)})$ with probabilities given by $p_0 = \frac{V^{-1}(\alpha^{(0)})}{D(\alpha^{(0)})}$ at $\alpha^{(0)}$, $p_{\alpha'} = \frac{V^{-1}(\alpha')}{D(\alpha^{(0)})}$, for $\alpha' \in \mathcal{N}_1(\alpha^{(0)}) \cup \cdots \cup \mathcal{N}_{[n/2]}(\alpha^{(0)})$, where $D(\alpha^{(0)}) = V^{-1}(\alpha^{(0)}) + \sum_{\alpha' \in \mathcal{N}_1(\alpha^{(0)}) \cup \cdots \cup \mathcal{N}_{[n/2]}(\alpha^{(0)})} V^{-1}(\alpha')$. Set $\alpha^{(0)} = \alpha^{(1)}$ and go to Step 2.

If no better tour is found till Step 5 with $k = [n/2]$, a tour is selected randomly with preassigned probabilities from among all the tours already searched in all the $[n/2]$ neighborhoods, including the initial tour, and the algorithm is started afresh from Step 2. The preassigned probabilities are computed so as to put smaller chances of being selected as the next tour for those having larger cost. From the above-described method, it is clear that the algorithm may get stuck at a local minimum if there is no better alternative among the searched neighborhoods. A remedy in this regard may be suggested as to restart the algorithm with another initial tour chosen randomly from among all the searched neighborhoods. This may not entirely solve the problem of local optimum, but has been found to be effective for the search of the global optimum.

Step 7: The algorithm continues till a solution reached with some desired accuracy by satisfying some convergence criterion.

This accuracy may be achieved in several ways, for example, by limiting the number of returns, say 10, to a certain tour. In the present paper, we have generalized the approach mentioned in [31], by suggesting a probabilistic method, as follows. At the ith return $(i \geq 1)$ to a tour $\alpha^{(0)}$, the preassigned probabilities p_0 and $p_{\alpha'}$, respectively, for choosing $\alpha^{(0)}$ and α' in $\mathcal{N}_1(\alpha^{(0)}) \cup \cdots \cup \mathcal{N}_{[n/2]}(\alpha^{(0)})$, as described in Step 6 above, are modified as $p_0^{(i)} = \frac{V^{(i)}(\alpha^{(0)})}{D^{(i)}(\alpha^{(0)})}$, $p_{\alpha'}^{(i)} = \frac{V^{(i)}(\alpha')}{D^{(i)}(\alpha^{(0)})}$ with $D^{(i)}(\alpha^{(0)}) = V^{(i)}(\alpha^{(0)}) + \sum_{\alpha' \in \mathcal{N}_1(\alpha^{(0)}) \cup \cdots \cup \mathcal{N}_{[n/2]}(\alpha^{(0)})} V^{(i)}(\alpha')$, $V^{(i)}(\alpha^{(0)}) = V^{-1}(\alpha^{(0)}) + (D(\alpha^{(0)}) - V^{-1}(\alpha^{(0)})) \times \frac{i}{n^{[n/2]}}$, $V^{(i)}(\alpha') = \max\{V^{-1}(\alpha') - (D(\alpha^{(0)}) - V^{-1}(\alpha^{(0)})) \times \frac{i}{n^{[n]}}, 0\}$, for $\alpha' \in \mathcal{N}_1(\alpha^{(0)}) \cup \cdots \cup \mathcal{N}_{[n/2]}(\alpha^{(0)})$. This increases the probability of choosing the same tour $\alpha^{(0)}$ at successive returns. We suggest stopping the

algorithm when this probability exceeds a preassigned value close to unity, usually taken as 0.95 or 0.99.

The VNS algorithm for a minimization problem, as introduced in [27], is a descent method, while the proposed method is a descent-ascent method with the inclusion of Step 6. This is because, even if a new solution is worse than the present one, a better solution might be obtained from among all the neighbors of the worse tour in any subsequent step. The concept of such descent-ascent method is motivated from the well-known simulated annealing (SA) method [21], where the similarity lies in the selection of a new solution from among all its neighbors through some stochastic mechanism. The proposed algorithm, including the stochastic approach, may be summarized as follows.

Initialization: Obtain the initial tour $\alpha^{(0)}$ using the method of Section 3.
Repeat the following procedure until the stopping condition is achieved:
(1) *Set* $k \longleftarrow 1$;
(2) *Until* $k = [n/2]$, *repeat the following steps:*
(a) *Construction of Neighborhood:*
Construct the kth order neighborhood $\mathcal{N}_k(\alpha^{(0)})$ of $\alpha^{(0)}$, and compute the cost $V(\alpha')$ corresponding to the tour α', for $\alpha' \in \mathcal{N}_k(\alpha^{(0)})$
(b) *Exploration of Neighborhood:*
Find the best neighbor $\alpha^{(1)}$ of $\alpha^{(0)}$ by satisfying
$V(\alpha^{(1)}) = \arg\min\{V(\alpha'), \quad \alpha' \in \mathcal{N}_k(\alpha^{(0)})\}$
(c) *Move or Not:*
If the solution $\alpha^{(1)}$ is better than $\alpha^{(0)}$, move there by setting $\alpha^{(1)} \longleftarrow \alpha^{(0)}$ and go to (1) to continue the search in $\mathcal{N}_1(\alpha^{(0)})$; otherwise set $k \longleftarrow k+1$.
(3) When $k > [n/2]$, check the stopping condition. If the condition is not satisfied choose a tour
$\alpha^{(1)}$ randomly with preassigned probabilities
among all the neighbors in $\alpha^{(0)} \cup \mathcal{N}_1(\alpha^{(0)}) \cup \cdots \cup \mathcal{N}_{[n/2]}(\alpha^{(0)})$; then set $\alpha^{(1)} \longleftarrow \alpha^{(0)}$ and go to (1).
Stopping Condition:
Until the probability of returning to the same tour is above 0.95 or 0.99.

The performance of the hybrid VNS algorithm in comparison to the existing algorithms are assessed through various simulation studies and real-life examples and are reported in the next section.

5. Illustrations

In this section, we investigate the performance of the proposed algorithm in comparison with the various other algorithms available in R. We consider two examples, namely USCA50 and ft53, for symmetric and asymmetric TSPs, respectively, from the TSPLIB dataset of R. We also consider two asymmetric TSPs with 10 and 50 cities, respectively, where the distances between the cities are simulated from an *exponential* distribution with mean = 25. The performance of the proposed algorithm is compared with NNA, RNNA, 2-*Opt*, FI, NI, along with the VNS-1 and the VNS-2 algorithms by considering the two real life as well as the two simulated datasets, as described before. The efficiency of the proposed algorithm over any of the above seven algorithms is defined by the ratio of the values of the cost for the tour obtained by the particular competing algorithm and the same for the tour obtained by the proposed algorithm. Formally, the efficiency is defined as $\frac{V(\alpha*)}{V(\alpha^P)}$, with α^* and α^P denoting the tours obtained by the competing and the proposed algorithm, respectively. The efficiency values may be multiplied by 100 to express them in percentage. As the proposed algorithm has an in-built stochastic component, the computation of the efficiency value is repeated 1000 times and the average, along with the minimum and maximum efficiency values, are reported, for each of the seven cases. Tables 1 and 2, respectively, present the findings of USCA50 and ft53 datasets, while Tables 3 and 4 report the findings from the simulation experiments on the basis of 10 and 50 cities, respectively.

For the USCA50 dataset, which is an example of symmetric TSP, the mean efficiencies of the proposed algorithm with respect to the seven competing algorithms are greater than 100% indicating the superiority of the proposed algorithm over those. However, the gain in average efficiency of the proposed algorithm over the 2-*Opt*, *FI*, VNS-1, VNS-2 algorithms are marginal compared to those over the rest. While comparing with these four algorithms, the minimum efficiency gain of the proposed algorithm over 1000 such repetitions falls marginally below 100% indicating that, in some situations, these algorithms may outperform the proposed one, albeit to a small extent. For the ft53 dataset, which is an asymmetric TSP, the gain in average efficiency of the proposed algorithm over all the seven competing algorithms is found to be moderate to very high, with an increase by 9–58%. Moreover, the minimum efficiency values for all other competing algorithms, except the VNS-2 algorithm, are found to be larger than 100%. This indicates that in the worst case also, the proposed algorithm possibly outperforms all except the VNS-2 algorithm, in which case also the efficiency loss is marginal. Thus the performance of the proposed algorithm over the seven competing algorithms may be claimed to be marginal to moderately superior for an asymmetric TSP.

For the two simulated examples of asymmetric TSP, the mean efficiency is never less than 100% and sometimes may be as high as 200%. Although in some cases, the minimum efficiency in comparison to 2-*opt*, VNS-1 and VNS-2 are less than 100%, the loss in minimum efficiency is always marginal.

The distributional property resulting from the choice of the initial tour in comparison to the more popular 2-*opt* approach is studied by constructing histogram of the final tour length as obtained over 1000 different iterations. Such histograms of the final solutions, for obtaining the optimal or near-optimal tour by the 2-

Table 1
Efficiency of the tours obtained by proposed algorithm with respect to the different algorithms over 1000 repetitions for USCA50 data.

Efficiency measures	Algorithms						
	α^{NNA}	α^{RNNA}	α^{2-Opt}	α^{FI}	α^{NI}	α^{VNS-1}	α^{VNS-2}
Minimum	1.014	1.014	0.969	0.974	1.014	0.969	0.962
Mean	1.117	1.072	1.009	1.012	1.068	1.008	1.006
Maximum	1.523	1.237	1.124	1.138	1.194	1.053	1.053

Table 2
Efficiency of the tours obtained by proposed algorithm with respect to the different algorithms over 1000 repetitions for ft53 data.

Efficiency measures	Algorithms						
	α^{NNA}	α^{RNNA}	α^{2-Opt}	α^{FI}	α^{NI}	α^{VNS-1}	α^{VNS-2}
Minimum	1.161	1.161	1.015	1.023	1.033	1.003	0.994
Mean	1.581	1.183	1.195	1.168	1.151	1.107	1.091
Maximum	2.099	1.319	1.244	1.217	1.293	1.162	1.153

Table 3
Efficiency of the tours obtained by proposed algorithm with respect to the different algorithms over 1000 repetitions for $n = 10$ randomly selected cities.

Efficiency measures	Algorithms						
	α^{NNA}	α^{RNNA}	α^{2-Opt}	α^{FI}	α^{NI}	α^{VNS-1}	α^{VNS-2}
Minimum	1.131	1.131	0.972	1.021	1.093	0.951	0.939
Mean	1.734	1.524	1.325	1.381	1.512	1.010	1.007
Maximum	2.028	1.938	1.503	1.632	1.822	1.049	1.032

Table 4
Efficiency of the tours obtained by proposed algorithm with respect to the different algorithms over 1000 repetitions for $n = 50$ randomly selected cities.

Efficiency measures	Algorithms						
	α^{NNA}	α^{RNNA}	α^{2-Opt}	α^{FI}	α^{NI}	α^{VNS-1}	α^{VNS-2}
Minimum	1.107	1.081	1.058	1.061	1.073	0.992	0.973
Mean	2.061	1.664	1.429	1.513	1.608	1.135	1.129
Maximum	2.352	1.837	1.634	1.719	1.814	1.417	1.328

opt algorithm, the VNS-2 algorithm with the final solution of the 2-*opt* algorithm as the initial tour and the proposed hybrid VNS with the suggested initial tour have been considered. The objective of drawing such histogram is to show the consistency and accuracy of an algorithm, for finding the optimal or near-optimal tour starting from the respective initial tour. Three instances (viz. *krob*100, *kroa*200 and *rd*400) from TSPLIB dataset are considered here to show the convergence from initial to final optimal or near-optimal tour. The VNS-1 is not considered owing to its inferiority to the VNS-2 algorithm as reported in [27,30]. In Fig. 1, for *krob*100 data, the histograms of the distribution of the final tour lengths as obtained by the 2-*opt* algorithm, VNS-2 algorithm and the proposed hybrid VNS algorithm, are given. The tour length corresponding to the suggested initial tour is also marked. In each case, the respective algorithm has been repeated 1000 times and the histogram of the corresponding final tour lengths are drawn. It may be noted that our initial algorithm is non-random, yielding the same tour length for a given problem, while our proposed algorithm predominantly converges to an optimal tour with same tour length, which is found to be better than the final solutions of the other algorithms. Note that such final solutions obtained by the 2-*opt* and VNS-2 algorithms are not that consistent, yielding various optimum tour lengths over several iterations. The performance of the VNS-2 algorithm has been found comparatively better than the 2-*opt* in the sense of lower tour length and narrower range of the optimal solutions. It has also been noticed that the 2-*opt* algorithm performs uniformly better than our suggested initial. The main advantage of the proposed hybrid VNS lies in its ability to yield better solution than the 2-*opt* and the VNS-2 algorithms even when the starting point is relatively inferior. The same exercise has been undertaken for two other datasets, viz. *kroa*200 and *rd*400, and with comparison through the histograms, as shown in Figs. 2 and 3, the efficacy and the relative performances of the algorithms are assessed. The nature of findings has been found to be almost similar. Among 1000 iterations, the proposed hybrid VNS algorithm performs uniformly better than the VNS-2 and the 2-*opt* algorithms.

Following Geng et al. [37], we also obtain the best, the average and the worst cost values out of 1000 replications of the proposed algorithm for some of the benchmark problems of the TSPLIB datasets. An error in percentage for each instance, computed on the basis of the percentage of the deviation of the best value as obtained through the proposed algorithm from the optimum value as reported for the said dataset, has also been computed. The corresponding CPU times (in second) have also been noted. The results are given in Table 5. We find that the loss of efficiency in the best solution, in comparison with the optimum one, as reported in the TSPLIB dataset, is mostly within 10% and seldom exceeds that, specifically for problems with a larger number of cities. Moreover, in some instances, the proposed algorithm gives better solutions (for example, for *krob*100, *pr*107 and *pr*144 datasets) than those reported in the corresponding TSPLIB dataset, with negative error percentage entries in the table. The entire computation for the proposed algorithm is carried out in `R` software version 3.3.1 with a modest supporting system `AMD Phenom(tm) II N930 Quad-Core Processor 2.00 GHz` with 4.00 GB RAM in a standard laptop.

The hybrid VNS algorithm for solving the TSP, has been successfully implemented in several other application areas and these applications are briefly described in Section 6.

6. Some other applications

The hybrid VNS algorithm, as introduced in the present paper, is an extension of the usual VNS algorithm, where the improvement from one solution to its neighbor is done through a stochastic step, similar to the annealing schedule. Such stochastic step enables the algorithm to escape a local optimality and attain the actual or approximate global optimal solutions. We have discussed the applicability of this hybrid discrete optimization procedure in three other problems, where the issue of discrete optimization is to be addressed. This ensures the versatility and applicability of the proposed procedure in various other complex problems, apart from its ability to solve the conventional TSP with moderate to high computational complexity.

6.1. Optimal design of experiments

Finding the optimal design for allocating two or more treatments to a fixed group of experimental units with several known covariates is an important problem in many scientific research and has been extensively studied by Hore et al. [31,32]. The design problem here is to determine an optimal allocation rule for the experimental units with known covariate(s) into multiple treatments such that both treatment and covariate effects, or only treatment effect, are efficiently estimated in terms of the commonly used *D*- or *A*-optimality. The actual search may be judiciously confined to a smaller number of schemes leading to an optimal solution, while finding the exact optimal allocation design is generally a computationally challenging issue, especially when the number of experimental units and the number of covariates are large. However, a computationally tractable algorithm with lesser amount of computer time is always preferred to achieve an optimal or near-optimal solutions for such problems. In this context, this improved hybrid VNS algorithm, with hybridization of the stochastic approach, has been effectively developed and implemented in [31,32]. The algorithm generates efficient allocation in comparison to random allocation and other heuristics allocation rules suggested by Harville [38], which have been demonstrated through simulation experiments and real-life data analysis. In the process, the computational complexity could be reduced to a great extent.

6.2. Wireless sensor network

Sensor networks aim at monitoring their surroundings for event detection and object tracking. Owing to failure or death of sensors, false signal may be transmitted. The distributed fault detection in wireless sensor network (WSN) has been considered as problem of interest in Nandi et al. [33]. In particular, they have considered how to take decision regarding fault detection in a noisy environment as a result of false detection or false response of event by some sensors, where the sensors are placed at the center of regular hexagons and the event can occur at only one hexagon. The fault detection schemes explicitly introduce the error probabilities into the optimal event detection process. Two types of detection probabilities, one for the center node, where the event occurs, and the other one for the adjacent nodes has been introduced. This second type of detection probability is new in sensor network literature. The construction of the adjacent nodes corresponding to center node may be framed by the suggested neighborhood structure and the selection of any of the adjacent node may be designed through the hybrid VNS procedure. It may be noted that the total number of such nodes in the WSN is extremely large, thus requiring some systematic optimization procedure to arrive at the best solution. The researchers have developed schemes under the model selection procedure and multiple model selection procedure and use the concept of Bayesian model averaging to identify a set of likely fault sensors and obtain an average predictive error.

Table 5
Efficiency of the proposed algorithm for TSP benchmark instances from the TSPLIB data set.

Sl no.	Instances	No. of cities	Optimum	Best	Average	Worst	Error (in %)	CPU time (in seconds)
1	gr17	17	2085	2085	2085	2085	0.00	16.51
2	gr21	21	2707	2707	2707	2707	0.00	18.38
3	gr24	24	1272	1272	1272	1272	0.00	23.14
4	fri26	26	937	937	937	937	0.00	39.57
5	bays29	29	2020	2020	2020	2020	0.00	48.29
6	dantzig42	42	699	699	699	699	0.00	151.38
7	swiss42	42	1273	1273	1273	1273	0.00	151.41
8	gr48	48	5046	5046	5046	5046	0.00	184.67
9	eil51	51	426	428.98	428.98	428.98	0.69	254.57
10	berlin52	52	7542	7544.36	7544.36	7544.36	0.03	261.32
11	brazil58	58	25395	25425	25592.72	25664	0.12	401.55
12	st70	70	675	677.11	677.11	677.11	0.31	532.61
13	eil76	76	538	545.39	552.57	566.50	1.37	614.28
14	pr76	76	108159	108159	108159	108159	0.00	614.36
15	rat99	99	1211	1240.38	1241.26	1242.40	2.42	828.57
16	rd100	100	7910	7910.4	7918.36	7930.39	0.00	937.68
17	kroa100	100	21282	21618.2	21695.79	21846.4	1.57	938.24
18	krob100	100	22141	22139.07	22140.20	22144.10	-0.009	938.26
19	kroc100	100	20749	20750.76	20809.29	20926.35	0.00	937.92
20	krod100	100	21294	21294.29	21490.62	21883.29	0.00	938.52
21	kroe100	100	22068	22174.6	22193.8	22222.36	0.48	937.83
22	eil101	101	629	642.31	648.27	657.91	2.11	941.39
23	lin105	105	14379	14383	14395.64	14412.75	0.00	962.47
24	pr107	107	44303	44301.68	44314.92	44324.84	-0.003	972.64
25	pr124	124	59030	59030.74	59051.82	59075.67	0.00	1029.57
26	bier127	127	118282	118974.6	119006.39	119054.4	0.58	1058.43
27	ch130	130	6110	6140.66	6153.72	6165.14	0.50	1082.19
28	pr136	136	96772	97979.11	97985.84	98012.75	1.24	1101.27
29	pr144	144	58537	58535.22	58563.97	58587.14	-0.003	1135.84
30	ch150	150	6528	6639.52	6644.95	6666.66	1.71	1201.63
31	kroa150	150	26524	26943.31	26947.17	26962.6	1.58	1201.49
32	krob150	150	26130	26527.57	26537.04	26576.12	1.52	1201.49
33	pr152	152	73682	73847.6	73855.11	73885.15	0.22	1234.27
34	u159	159	42080	42436.23	42467.61	42593.14	0.84	1281.73
35	rat195	195	2323	2450.14	2453.81	2464.32	5.47	1382.34
36	d198	198	15780	16075.84	16079.28	16085.53	1.87	1403.94
37	kroa200	200	29368	30300.56	30339.67	30365.87	3.17	1468.61
38	krob200	200	29437	30447.30	30453.22	30472.42	3.43	1468.61
39	pr226	226	80369	80469.31	80514.64	80695.97	0.01	1567.28
40	gil262	262	2378	2492.85	2501.86	2537.91	4.81	1596.73
41	pr264	264	49135	51155.38	51197.14	51364.2	4.03	1602.33
42	pr299	299	48191	50271.69	50373.12	50778.86	4.32	1687.52
43	lin318	318	42029	43924.08	43964.93	44128.35	4.51	1734.81
44	rd400	400	15281	16155.91	16250.21	16199.47	5.42	1953.49
45	fl417	417	11861	12180.78	12183.14	12192.56	2.69	2015.37
46	pr439	439	107217	111750.3	111771.2	111854.8	4.22	2106.84
47	pcb442	442	50778	50783.55	50800.24	50867	0.01	2183.27
48	U574	574	36905	39573.88	39629.11	39850.02	7.23	2351.39
49	rat575	575	6773	7349.81	7362.51	7413.27	8.52	2461.87
50	u724	724	41910	45725.39	45729.71	45746.97	9.10	3527.63
51	rat783	783	8806	9707.166	9707.364	9708.156	10.23	4028.14
52	pr1002	1002	259045	280368.2	280563.9	281346.7	8.23	6473.88
53	pcb1173	1173	56892	63354.82	63435.95	63760.47	11.36	9531.54
54	d1291	1291	50801	56088.31	56095.33	56123.41	10.40	11287.52
55	rl1323	1323	270199	295607.3	295611.2	295626.8	9.45	13421.56
56	fl1400	1400	20127	21040.65	21085.98	21085.98	4.53	15423.80
57	d1655	1655	62128	69992.49	70337.23	71716.2	12.65	18634.29
58	vm1748	1748	336556	366755.5	366757.8	366768.7	8.97	20314.86
59	u2319	2319	234256	252683.8	262595.6	252742.4	7.86	24534.72
60	pcb3038	3038	137694	154550.7	154565.4	154619.8	12.24	29412.80

6.3. Reliability theory and survival analysis

In the determination of optimum Type-II progressive censoring scheme, the experimenter needs to carry out an exhaustive search within the set of all admissible censoring schemes. The existing recommendations are only applicable for small sample sizes. The implementation of exhaustive search techniques for large sample sizes is not feasible in practice. The hybrid VNS algorithm may be invoked to overcome this computationally challenging issue, especially for large sample sizes. It is also noted, that the algorithm gives exactly the same solution as obtained through the exhaustive search, where it is possible to implement, i.e. for small sample set up. A cost function-based optimality criterion has been proposed in [34], which is scale invariant for location-scale and log-location-scale families of distributions. This approach accommodates the constrained optimization set up in the hybrid VNS procedure. A sensitivity analysis is also carried out to study the effect of misspecification of parameter values or cost coefficients on the optimum solution.

7. Concluding remarks

The present paper proposes a VNS algorithm with an inbuilt stochastic search approach for solving both symmetric as well as asymmetric TSPs, with equal ease of computation. While the performance of the algorithm seems to be marginally better than the presently available methods for symmetric problems, a significant improvement is observed for asymmetric cases. The average computation times for the proposed algorithm with the already mentioned computational strength for the 60 symmetric TSP benchmark problems are noted. All the standard algorithms are primarily developed to deal with symmetric TSPs and the asymmetric TSPs are reformulated as symmetric TSPs doubling the number of cities by introducing a dummy city corresponding to each city [39]. The main advantage of the proposed algorithm is its applicability to the asymmetric problems with equal flexibility, without converting the asymmetric problem into a symmetric one. The adaptive simulated annealing algorithm with greedy search (ASA-GS) for the TSP has been developed in [37] that has achieved better results with faster convergence for large-scale problems of the TSPLIB dataset. But, the method has been framed for only symmetric TSPs, while the proposed algorithm has been framed for a general TSP format. It is important to note that the present algorithm yields a better solution for a few symmetric problems given in *krob*100, *pr*107, and *pr*144, in comparison to those reported in the TSPLIB dataset. A modified simulated annealing (SA) algorithm, named as list-based simulated annealing (LBSA) algorithm has been developed for solving symmetric TSPs in [40] by incorporating a selected list-based cooling schedule to obtain an optimal or near-optimal tour, with faster convergence rate. However, the LBSA algorithm also fails to achieve better optimum values for the above three datasets [40]. The flexibility of the proposed algorithm to solve some other optimization problems, apart from the TSP, arising in several other disciplines has been mentioned. This indicates that the hybrid VNS is an interesting extension of the usual VNS and has the capability to be implemented in various other fields.

Till date, it is generally believed that Concorde [41,42] is possibly the best available algorithm for solving symmetric TSP's based on the Branch-and-Cut [43] and Branch-and-Bound [44] method. The optimal tour value of USCA50 and ft53 data from the TSPLIB dataset by using Concorde are reported in [24,45], respectively. Although the average performance of the proposed algorithm has been found to be marginally inferior to Concorde, its performance has been found to be equally effective as Concorde in some replications. The detailed description and steps of the Concorde algorithm are not available in public domain and the software has to be installed separately, governed by a different license [42] and may not be compatible with all systems.

Applegate et al. [41] established new breakthrough results for the TSP. For instance, the research team solved an instance of the TSP of 13,509 cities corresponding to all US cities with populations of more than 500 people. The approach involved ideas from polyhedral combinatorics and combinatorial optimization, integer and linear programming, computer science data structures and algorithms, parallel computing, software engineering, numerical analysis, graph theory, and more. The solution of this instance demanded the use of three Digital Alpha Server 4100s (with a total of 12 processors) and a cluster of 32 Pentium-II PCs. The complete calculation took approximately three months of computer time. The code had more than 1000 pages and is based on state-of-the-art techniques from a wide variety of scientific fields [46]. This speaks of the volume of labor and effort needed to solve a large TSP. However, our method has the ability to tackle moderately large TSPs and to give relatively efficient solution by consuming very limited computer time in a standard laptop, without concerning about the problem being symmetric or asymmetric. At the same time, the present algorithm is intuitively very simple and may be judiciously implemented to solve various other problems of discrete optimization coming from different fields, as have been discussed [31–34]. Some interesting extensions of the existing TSPs might be formulated and may be solved by suitably modifying the proposed algorithm in a constrained setup. One such extension might be assigning some infinitely large values in the cost matrix, thus blocking the corresponding paths in the tour. Another possible extension is to allow multiple visits to a selected group of cities, either as an option or as a mandate. The TSP with profits [47] may be an interesting extension, where it is not necessary to visit all the vertices and a profit is associated with each vertex. Extension of the algorithm to a multi-dimensional TSP, e.g. the spherical TSP, in which all the nodes (cities) and paths (solutions) are located on the surface of a sphere [48], might be an interesting problem. Works in these directions are already in progress and would be reported elsewhere.

Acknowledgments

The authors like to thank Michael Hahsler, Southern Methodist University, USA and Kurt Hornik, Wirtschaftsuniversitiat Wien, Austria for some helpful discussions and their R – guide notes 'TSP-Infrastructure for the Traveling Salesperson Problem and Package TSP'. The authors are also grateful to William Cook, Professor in Combinatorics and Optimization, University of Waterloo for his valuable suggestions and his reference at www.iwr.uni-heidelberg.de/groups/comopt/index.html. The authors are also thankful to the associate editor and two anonymous reviewers for their valuable comments, which improved the presentation of the paper to a considerable extent.

References

[1] Z.C.S.S. Hlaing, M.A. Khine, Solving traveling salesman problem by using improved ant colony optimization algorithm, Int. J. Inf. Educ. Technol. 1 (5) (2011) 404–409.
[2] Y. Marinakis, M. Marinaki, G. Dounias, A hybrid particle swarm optimization algorithm for the vehicle routing problem, Eng. Appl. Artif. Intell. 23 (2010) 463–472.
[3] K. Savla, E. Frazzoli, F. Bullo, Traveling salesperson problems for the Dubins vehicle, IEEE Trans. Autom. Control 53 (6) (2008) 1378–1391.
[4] M.K. Mehmet Ali, F. Kamoun, Neural networks for shortest tour computation and routing in computer networks, IEEE Trans. Neural Netw. 4 (5) (1993) 941–953.
[5] D. Banaszak, G.A. Dale, A.N. Watkins, J.D. Jordan, An optical technique for detecting fatigue cracks in aerospace structures, Proceeding of 18th ICIASF (1999) 1–7.
[6] C.H. Papadimitriou, Euclidean traveling salesman problem is NP-complete, Theor. Comput. Sci. 4 (1978) 237–244.
[7] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman, New York, 1979.
[8] R.L. Graham, Handbook of Combinatorics, MIT Press, 1995.
[9] G. Finke, A. Claus, E. Gunn, A two-commodity network flow approach to the traveling salesman problem, Congressus Numerantium 41 (1984) 167–178.
[10] R. Bellman, Dynamic programming treatment of the travelling salesman problem, J. ACM 9 (1) (1962) 61–63.
[11] M. Held, R.M. Karp, A dynamic programming approach to sequencing problems, J. Soc. Ind. Appl. Math. 10 (1) (1962) 196–210.
[12] J. Kennedy, R.C. Eberhart, Particle swarm optimization, Proceedings of the IEEE International Conference on Neural Networks IV (1995) 1942–1948, http://dx.doi.org/10.1109/ICNN.1995.488968.
[13] Y. Shi, R.C. Eberhart, A modified particle swarm optimizer, Proceedings of IEEE International Conference on Evolutionary Computation (1998) 69–73.
[14] Y. Marinakis, M. Marinaki, A hybrid multi-swarm particle swarm optimization algorithm for the probabilistic traveling salesman problem, Comput. Oper. Res. 37 (2010) 432–442.
[15] T.A.S. Masutti, L.N. de Castro, A self-organizing neural network using ideas from the immune system to solve the traveling salesman problem, Inf. Sci. 179 (10) (2009) 1454–1468.
[16] M. Dorgio, V. Maniezzo, A. Colorni, Ant system: optimization by a colony of cooperating agents, IEEE Trans. Syst. Man Cybern. B: Cybern. 26 (1) (1996) 29–41.

[17] S. Ghafurian, N. Javadian, An ant colony algorithm for solving fixed destination multi-depot multiple traveling salesman problems, Appl. Soft Comput. 11 (1) (2011) 1256–1262.
[18] D. Whitley, A genetic algorithm tutorial, Stat. Comput. 4 (1994) 65–85.
[19] R. Baraglia, J.I. Hidalgo, R. Perego, A hybrid heuristics for the traveling salesman problem, IEEE Trans. Evol. Comput. 5 (6) (2001) 613–622.
[20] L. Jiao, L. Wang, A novel genetic algorithm based on immunity, IEEE Trans. Syst. Man Cybern. A: Syst. Hum. 30 (5) (2000) 826–830.
[21] S. Kirkpatrick, C.D. Gelatt, M. Vecchi, Optimization by simulated annealing, Science 220 (4598) (1983) 671–680.
[22] V. Cerny, Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm, J. Optim. Theory Appl. 45 (1985) 41–51.
[23] D.J. Rosenkrantz, R.E. Stearns, M.L.I. Philip, An analysis of several heuristics for the traveling salesman problem, SIAM J. Comput. 6 (3) (1977) 563–581.
[24] M. Hahsler, K. Hornik, TSP – infrastructure for the traveling salesperson problem, J. Stat. Softw. 23 (2) (2007) 1–21.
[25] G.A. Croes, Method for solving traveling-salesman problems, Oper. Res. 6 (6) (1958) 791–812.
[26] M.M. Flood, The traveling salesman problem, Oper. Res. 4 (1956) 61–75.
[27] P. Hansen, N. Mladenović, An introduction to variable neighborhood search, in: S. Voss, S. Martello, I. Osman, C. C. Roucairol (Eds.), Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization, Kluwer Academic Publishers, MA, USA, 1999, pp. 433–458.
[28] S. Lin, Computer solutions of the traveling-salesman problem, Bell Syst. Technol. J. 44 (1965) 2245–2269.
[29] S. Lin, B. Kernighan, An effective heuristic algorithm for the traveling salesman problem, Oper. Res. 1 (2) (1973) 498–516.
[30] P. Hansen, N. Mladenović, Variable neighborhood search: principles and applications, Eur. J. Oper. Res. 130 (2001) 449–467.
[31] S. Hore, A. Dewanji, A. Chatterjee, Design issues related to allocation of experimental units with known covariates into two treatment groups, J. Stat. Plan. Inference 155 (2014) 117–126.
[32] S. Hore, A. Dewanji, A. Chatterjee, On optimal allocation of experimental units with known covariates into multiple treatment groups, Calcutta Stat. Assoc. Bull. 68 (1-2) (2016) 69–81.
[33] M. Nandi, A. Dewanji, B. Roy, S. Sarkar, Model selection approach for distributed fault detection in wireless sensor networks, Int. J. Distrib. Sens. Netw. (2014), http://dx.doi.org/10.1155/2014/148234.
[34] R. Bhattacharya, B. Pradhan, A. Dewanji, On optimum life testing plans under type-II progressive censoring scheme using variable neighborhood search algorithm, TEST 25 (2) (2014) 309–330, http://dx.doi.org/10.1007/s11749-015-0449-z.
[35] TSPLIB. https://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/ (accessed 15.09.17).
[36] J. Bang-Jensen, G. Gutin, A. Yeo, When the greedy algorithm fails, Discret. Optim. 1 (2) (2004) 121–127.
[37] X. Geng, Z. Chen, W. Yang, D. Shi, K. Zhao, Solving the traveling salesman problem based on an adaptive simulated annealing algorithm with greedy search, Appl. Soft Comput. 11 (4) (2011) 3680–3689.
[38] D.A. Harville, Nearly optimal allocation of experimental units using observed covariate values, Technometrics 16 (4) (1974) 589–599.
[39] R. Jonker, T. Volgenant, Transforming asymmetric into symmetric traveling salesman problems, Oper. Res. Lett. 2 (1983) 161–163.
[40] S. Zhan, J. Lin, Z. Zhang, Y. Zhong, List-based simulated annealing algorithm for traveling salesman problem, Comput. Intell. Neurosci. (2016), http://dx.doi.org/10.1155/2016/1712630.
[41] D. Applegate, R.E. Bixby, V. Chvatal, W. Cook, TSP cuts which do not conform to the template paradigm, in: M. Junger, D. Naddef (Eds.), Computational Combinatorial Optimization, Optimal or Provably Near-Optimal Solutions Lecture Notes in Computer Science, vol. 2241, Springer-Verlag, London, 2000, pp. 261–304.
[42] D. Applegate, R.E. Bixby, V. Chvatal, W. Cook, Concorde TSP Solver, 2006 http://www.tsp.gatech.edu/Concorde/.
[43] M. Padberg, G. Rinaldi, Facet identification for the symmetric traveling salesman polytope, Math. Program. 7 (2) (1990) 219–257.
[44] A. Land, A. Doig, An automatic method for solving discrete programming problems, Econometrica 28 (1960) 497–520.
[45] Best Known Solutions for Asymmetric TSPs, 2017 (accessed 15.09.17) http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/ATSP.html.
[46] P. Moscato, C. Cotta, A Gentle Introduction to Memetic Algorithms Handbook of Metaheuristics, Kluwer Academic Publishers, New York, 2003.
[47] D. Feillet, P. Dejax, M. Gendreau, Traveling salesman problems with profits, Transp. Sci. 39 (2) (2005) 188–205.
[48] X. Chen, Y. Zhoua, T. Zhonghua, L. Qifang, A hybrid algorithm combining glowworm swarm optimization and complete 2-opt algorithm for spherical travelling salesman problems, Appl. Soft Comput. 8 (September) (2017) 104–114.