EXPERIMENTAL ANALYSIS OF HEURISTICS FOR THE BOTTLENECK TRAVELING SALESMAN PROBLEM

JOHN LARUSIC, ABRAHAM P. PUNNEN, AND ERIC AUBANEL

ABSTRACT. In this paper we develop efficient heuristic algorithms to solve the bottleneck traveling salesman problem (BTSP). Results of extensive computational experiments are reported. Our heuristics produced optimal solutions for all the test problems considered from TSPLIB, JM-instances, National TSP instances, and VLSI TSP instances in very reasonable running time. We also conducted experiments with specially constructed 'hard' instances of the BTSP that produced optimal solutions for all but seven problems. Some fast construction heuristics are also discussed. Our algorithms could easily be modified to solve related problems such as the maximum scatter TSP and testing hamiltonicity of a graph.

Key Words: Traveling salesman problem, bottleneck TSP, experimental analysis, maximum scatter TSP.

1. INTRODUCTION

Let $G = (V, E)$ be a complete undirected graph on n nodes. For each edge $(i, j) \in E$ a non-negative cost c_{ij} is prescribed. Then the bottleneck traveling salesman problem (BTSP) is to

$$\begin{aligned} &\text{Minimize } \max\{c_{ij} : (i,j) \in H\} \\ &\text{Subject to} \\ &\quad H \in \Pi, \end{aligned}$$

where Π is the collection of all Hamiltonian cycles (tours) in G. The matrix $C = (c_{ij})_{n\times n}$ is called the cost matrix associated with the BTSP.

To the best of our knowledge, BTSP was introduced by Gilmore and Gomory [11] in 1964, where the edge costs were assumed to be specially structured. Related works on special cases of the BTSP include various extensions of the Gilmore and Gomory class of problems [17, 18, 20, 23, 40, 41] and BTSP on a Halin graph [27]. Many of these special cases can be solved in polynomial time. For a state-of-the-art discussion on polynomially solvable cases of the BTSP, we refer to book chapter by Kabadi and Punnen [18]. The BTSP with a general cost matrix was first studied by Gabovich et al. [9] in 1971. The BTSP is well known to be NP-hard. In fact, unless P = NP,

This work was supported by NSERC discovery grants awarded to Abraham P. Punnen and Eric Aubanel.

there does not exist a polynomial time ϵ-approximation algorithm for the BTSP for any $\epsilon > 0$ [6, 26, 34]. However, polynomial algorithms with guaranteed performance ratios exist for various special cases. For example, when the edge costs satisfy the τ-triangle inequality (i.e. $c_{ij} \leq \tau(c_{ik} + c_{kj})$ for all $i, j, k \in V$ and $\tau \geq \frac{1}{2}$), a 2τ-approximate solution can be obtained in polynomial time [18] and this performance bound seems to be the best possible for problems in this class. That is, unless P = NP, there exists no polynomial $2\tau - \epsilon$ approximation algorithm for the BTSP for any $\epsilon > 0$ even if the edge costs satisfy the τ-triangle inequality for $\tau > 1/2$ [18]. The special case of the above result where $\tau = 1$ (i.e. the edge costs satisfy the triangle inequality) has been obtained independently by Parker and Rardin [26] and Doroshko and Sarvanov [6]. For similar results on other bottleneck problems, including the BTSP, we refer to Hochbaum and Shmoys [13]. It is well known that the Euclidean TSP can be solved by a polynomial approximation scheme [3]. However, for any $\epsilon > 0$ it is NP-hard to obtain a $(2 - \epsilon)$-approximate solution for the BTSP even with a Euclidean cost matrix [34]. In this sense the Euclidean version of BTSP is harder than its TSP counterpart.

Despite these negative results, our main heuristic algorithms produced optimal solutions (in many cases with a proof of optimality) for almost all of the test problems considered, even for problems with over 30,000 nodes. Furthermore, these excellent computational results were achieved in a very reasonable computational time.

All of the known exact algorithms for the BTSP are of an enumerative type. Garfinkel and Gilbert [8] proposed a branch and bound algorithm to solve the BTSP and its directed version. They reported computational results on random problems with the number of nodes ranging from 10 to 100. Carpeneto, Martello and Toth [5] reported computational results with branch and bound algorithms on random problems of size between 40 to 200 nodes. Sergeev [35, 36] proposed dynamic programming based algorithms to solve the BTSP. One of our heuristic algorithms can easily be modified to get an exact optimal solution for the BTSP.

Although heuristic algorithms for the traveling salesman problem (TSP) have been thoroughly investigated in literature with well designed experimental analysis [16], systematic experimental analysis of heuristic algorithms for the BTSP reported in literature are limited to small-size problems. Garfinkel and Gilbert reported experimental results using a construction heuristic on random problems of sizes up to 100 nodes. The algorithm produced optimal solutions in many cases. In one set of problems the solution quality appeared to be decreasing as the problem size varied from 10 to 100. Ramakrishnan, Punnen and Sharma [30] considered a simple binary search based threshold algorithm using the 2-max bound [23] as a starting lower bound and the objective function value of the nearest neighbor heuristic as the upper bound. The $2 - max$ bound is obtained as follows: For each node

i find the second smallest cost of an edge incident on i, counting multiplicities. Let $\zeta(i)$ be this value. Note that if the smallest cost of an arc incident on node i is not unique, then $\zeta(i)$ takes the value of this smallest cost. The 2-max bound is defined as $\max\{\zeta(i) \mid i \in V\}$. Computational results on 72 TSPLIB problems of sizes up to 783 nodes are reported in [30] with optimality guaranteed for 37 problems. The optimality of these solutions were established by showing that the objective function value of the solution produced was equal to a lower bound to the optimal objective function value. The optimality of the remaining 35 problems was not proven because of the low quality of the 2-max lower bound. Our experiments show that the solutions produced by the algorithm of Ramakrishnan et al [30] for the remaining 35 problems were also optimal. They also discussed a generalized threshold algorithm without computational results.

In this paper we develop several heuristics for the BTSP. One of our algorithms could be viewed as a generalization of the threshold heuristic of Ramakrishnan et al. [30]. Results of extensive computational experiments conducted using these algorithms are reported. The algorithms produced provably optimal solutions for all problems from TSPLIB with sizes varying from 14 to 18512 nodes with the exception of linhp318[1]. We could not attempt to solve pla33810 and pla85900 due to memory limitations. Our algorithms also produced provably optimal solutions on random problems generated by the TSP instance generator of Johnson and McGeoch [14, 16, 32] of sizes 1000 to 30,000 from the categories of Euclidean problems with random uniform points, Euclidean problems with random clustered points, and random distance matrices. Further, we obtained optimal solutions to the National TSP instances and VLSI TSP instances of [37]. All these excellent experimental results were achieved within a very reasonable computational time. Our implementation used an explicit cost matrix and hence we could not solve problems of size more than 32000 cities due to memory limitations.

The optimality of many of the solutions is established by showing that the objective function values of the solutions obtained are equal to a lower bound. For some solutions without such proof of optimality, we used an exact solver obtained by modifying our heuristic algorithm to verify optimality. Since for most of the standard TSP test problems the lower bound value was optimal, we have tested the algorithms on specially constructed hard instances that are generated in such a way that our lower bound for the optimal objective function value is not equal to the optimal objective function value of the corresponding BTSP. Out of 81 such hard instances all except seven problems produced optimal solutions. In addition to these powerful (heuristic) algorithms, we also have results on some simple and

[1]We did not test linhp318 because its format in TSPLIB was different from other problems and required separate processing to be consistent with our input format

fast heuristics which are adaptations of the corresponding TSP heuristics. As expected, the performance of these heuristics was not as good as that of our primary algorithms in terms of solution quality, but these algorithms computed reasonable solutions quickly.

Another problem closely related to the BTSP is the *maximum scatter traveling salesman problem* (MSTSP) [2] which is defined as follows:

Maximize $\min\{c_{ij} : (i,j) \in H\}$
Subject to
$H \in \Pi$

Any instance of the MSTSP can be reduced to an equivalent instance of the BTSP using cost matrix $\bar{C} = (\bar{c_{ij}})_{n\times n}$ where $\bar{c_{ij}} = \max_{rs\in E} c_{rs} - c_{ij}$. Although this transformation preserves optimality, it does not preserve theoretical approximation ratios. Nevertheless, our BTSP heuristics can be used to solve MSTSP using these transformations.

Applications of the BTSP model are not widely discussed in the literature as that of the TSP. However, many applications of the TSP model have an interpretation in the BTSP context. Specific applications of the BTSP model discussed in the literature include sequencing problems on single state variable machines [11], minimizing makespan in a two machine flow-shop [31], printed circuit board assembly [4], reconstructing sequential order from inaccurate adjacency information [19], medical image processing [2], and obtaining riveting sequences in joining metals in the aircraft industry [2].

The paper is organized as follows: In Section 2 we discuss our heuristic algorithms for BTSP. Computational results with these algorithms are reported in Section 3 and concluding remarks are given in Section 4.

2. Algorithms for the bottleneck TSP

We first introduce some notation. For any subgraph S of G with cost matrix C, $Max(S,C) = \max_{ij\in S} c_{ij}$ and $Sum(S,C) = \sum_{ij\in S} c_{ij}$. Let $BOBJ(C)$ denote the optimal objective function value of the BTSP with cost matrix C and TOBJ(C) be that of the TSP with cost matrix C. Let us now discuss some simple construction heuristics for the BTSP.

2.1. **Nearest Neighbor Heuristics.**

Perhaps the simplest construction heuristic for both the TSP and the BTSP is the nearest neighbor heuristic (NNH) [21]. This algorithm was used by Ramakrishnan et al [30] to obtain a starting upper bound for their threshold heuristic. The NNH selects a starting node and moves to its closest neighbor. From there, it moves to the closest unvisited neighbor and repeats this process until all nodes are visited. The quality of the solution produced might be improved by selecting different starting nodes and choosing the overall best solution. Such an algorithm where each node is tried as a starting node is called the *enhanced*

nearest neighbour heuristic (ENNH). For any real number δ, let C^δ be the $n \times n$ matrix defined by

$$c_{ij}^{\delta} = \begin{cases} 0 & \text{if } c_{ij} \leq \delta, \\ c_{ij} & \text{otherwise} \end{cases} \tag{1}$$

Instead of using the original cost matrix C, we could also use the cost matrix C^δ where δ is a lower bound on the optimal objective function value for the BTSP. This provides additional flexibility in selecting edge-costs in the NNH allowing possible random selections at various iterations. NNH applied on C^δ is called the bottleneck nearest neighbor heuristic (BNNH). Note that NNH when applied on the original cost matrix will produce the same solution for both TSP and BTSP if the same tie breaking rules are followed.

2.2. **Bottleneck Node Insertion Heuristic.** Simple modifications of the well known node insertion heuristic (NIH) for the TSP [21] yields another construction heuristic for the BTSP. We call this algorithm the *bottleneck node insertion heuristic* (BNIH). A general step of the BNIH can be described as follows:

Let $H^k = (\pi(1), \pi(2), \ldots, \pi(k), \pi(1))$ be a tour on k nodes in G. From the remaining $n-k$ nodes, choose a node r randomly and insert it into H^k optimally. Replace H^k by the new tour H^{k+1} on $k+1$ nodes and continue this process until a tour H^n in G is obtained. A straightforward implementation of this algorithm takes $O(n^3)$ time. We now observe that the BNIH can be implemented in $O(n^2)$ time. Let

$$Max(H^k, C) = \max_{1 \leq i \leq k} \{c_{\pi(i)\pi(i+1)}\} = c_{\pi(p)\pi(p+1)}, \tag{2}$$

and

$$Max(H^k - \{(\pi(p), \pi(p+1))\}, C) = \max_{\substack{1 \leq i \leq k \\ i \neq p}} \{c_{\pi(i)\pi(i+1)}\}$$

where $k+1 \equiv 1$. Note that the index p in (2) may not be unique and hence $Max(H^k)$ could be equal to $Max(H^k - \{(\pi(p), \pi(p+1))\})$. The cost Δ_i^r of inserting node r between nodes $\pi(i)$ and $\pi(i+1)$ is given by

$$\Delta_i^r = \begin{cases} \max\{Max(H^k, C), c_{\pi(i)r}, c_{r\pi(i+1)}\} & \text{if } i \neq p \\ \max\{Max(H^k - \{(\pi(p), \pi(p+1))\}, C), c_{\pi(i)r}, c_{r\pi(i+1)}\} & \text{if } i = p. \end{cases} \tag{3}$$

Choose q such that $\Delta_q^r = \min_{1 \leq i \leq k} \Delta_i^r$. Then node r is inserted between nodes $\pi(q)$ and $\pi(q+1)$ in H^k to obtain the new cycle H^{k+1} on $k+1$ nodes. The algorithm outputs a tour H^n on n nodes. It may be noted that given $Max(H^k, C)$ and $Max(H^k - \{(\pi(p), \pi(p+1))\}, C)$ for H^k and the index q, corresponding values for H^{k+1} can be obtained in $O(1)$ time. It may be noted that updating other relevant information per iteration, including computation of q, takes $O(n)$ time. Thus the complexity of BNIH can be verified

to be $O(n^2)$. We omit the step by step description of this algorithm and its complexity analysis. The performance of the BNIH might be enhanced by using the cost matrix C^δ instead of C and repeating the algorithm for different starting cycles and random tie breaking rules.

NNH, BNNH, ENNH, and BNIH are construction type heuristic algorithms that produce approximate solutions quickly (ENNH is found to be very time-consuming and does not appear to be cost-effective in solving BTSP even if one needs only an approximate solution.). The quality of these construction heuristic solutions is not very impressive as we will see in Section 3. However, the objective function value of the solutions provide upper bounds for the optimal objective function value of BTSP. This information could be used effectively in other algorithms to obtain improved running time.

2.3. **Feasibility Oracle.** Let us now discuss a procedure central to the rest of our algorithms.

The δ-feasibility problem: Given a real number δ, does there exist a tour H in G with $Max(H, C) \leq \delta$?

Clearly the δ-feasibility problem can be answered by solving the TSP on the cost matrix C^δ. Let H^* be an optimal solution to the TSP with cost matrix C^δ. Then $Sum(H^*, C^\delta) = 0$ if and only if there exists a tour H in G such that $Max(H, C) \leq \delta$. In fact H^* will be one such tour. Let $z_1, z_2, \ldots, z_m$ be an ascending arrangement of all distinct $c_{ij} : (i, j) \in G$. Choose the smallest index $i \in \{1, 2, \ldots, m\}$ such that the z_i-feasibility problem has a "yes" answer. Then an optimal solution to TSP(C^{z_i}) is an optimal solution to the BTSP. The index i can be identified by sequential search or binary search. This is the well known threshold algorithm [7] for bottleneck problems specialized to the BTSP. However, for this approach we need to solve in the worst case $O(\log m)$ TSPs which may not be computationally efficient.

In the past few decades, excellent heuristic algorithms have been developed by researchers to solve the TSP that produce optimal or near optimal solutions. For a state-of-the-art discussion of such heuristics we refer to the book chapter by Johnson and McGeoch [16]. Our algorithms exploit the power of these TSP heuristics to develop powerful heuristics to solve the BTSP. Suppose that we use a TSP heuristic, say α, instead of an exact algorithm to solve the δ-feasibility problem. Let H^α be the solution produced. If $Sum(H^\alpha, C^\delta) = 0$ then clearly there exists a tour H in G such that $Max(H, C) \leq \delta$, and in this case we have an "yes" answer for the δ-feasibility problem. But unlike the case of an exact TSP solver, if $Sum(H^\alpha, C^\delta) \neq 0$ then we cannot conclude a "no" answer for the δ-feasibility problem. This is because α, being a heuristic algorithm, could possibly return a non-optimal solution. To explore this case further we "shake" the cost matrix to see if

we can get an "yes" answer for the δ-feasibility problem. Note that the c_{ij} values in C^δ could be any positive number for the case of the δ-feasibility problem as long as we preserve the zero values. By changing the c_{ij} values and applying α again on the changed problem we might get a better solution. The *shake operation* does precisely this. Consider the matrix $C^{\delta,R} = (c_{ij}^{\delta,R})$ defined by

$$c_{ij}^{\delta,R} = \begin{cases} 0 & \text{if } c_{ij} \leq \delta \\ \text{a positive random number} & \text{if } c_{ij} > \delta \end{cases} \tag{4}$$

Note that for a tour H, $Sum(H, C^\delta) = 0$ if and only if $Sum(H, C^{\delta,R}) = 0$. The shake operation simply applies algorithm α on the cost matrix $C^{\delta,R}$. If no solution H^α with $Sum(H^\alpha, C^{\delta,R}) = 0$ is obtained, the shake operation is repeated a prescribed number, say #S, of times. If we have no tour H with $Sum(H, C^{\delta,R}) = 0$, we are tempted to conclude a "no" answer to the δ-feasibility problem. This approximate solution approach for the δ-feasibility problem can be incorporated in the threshold algorithm to obtain a heuristic for BTSP. We will come to this later. Let us now discuss how to extract a heuristic solution for BTSP by one application of the heuristic approach discussed above (with minor modifications) to solve the δ-feasibility problem.

2.3.1. *Extracting a heuristic solution.* Suppose δ is a lower bound for the optimal objective function value. Such a lower bound can be obtained by solving the bottleneck biconnected spanning subgraph problem (BB-SSP) [29] on G using the cost matrix C. This is one of the strongest known lower bounds for the BTSP. Other popular lower bounds include the 2-max bound [18, 30], bottleneck spanning tree bound [5, 18] and the bottleneck assignment bound [5]. Several algorithms are available to solve the BB-SSP [6, 23, 26, 29, 38]. Punnen and Nair [29] proposed an $O(m + n \log n)$ algorithm to solve this problem, and Manku [23] improved the algorithm with an $O(m)$ time bound, where m is the number of arcs in the underlying graph. For our applications, both these algorithms have complexity $O(n^2)$ since G is a complete graph.

Choosing δ as a lower bound to BOBJ(C), solve the δ-feasibility problem approximately by one application of a TSP heuristic α on the matrix C^δ. Let H^α be the solution produced and output H^α. If $Sum(H^\alpha, C^\delta) = 0$ then H^α is an optimal solution. Otherwise use at most $\#S$ shake operations and solve the TSP on the resulting cost matrix $C^{\delta,R}$ and output the best solution obtained. We call this heuristic the *simple threshold heuristic* or $(ST - heuristic)$. The steps of the ST-heuristic is summarized below.

ST-Heuristic

Step 1: Set the value of the control parameter $\#S$ and compute the lower bound δ (possibly by solving BBSSP).

Step 2: Use a TSP heuristic α to solve the TSP with cost matrix C^δ. (For example, use some recent variations of the Lin-Kernighan heuristic [1, 16, 22]). Let H^α be the resulting solution. If $Sum(H^\alpha, C^\delta = 0$ output H^α and stop. Otherwise, designate H^α as the "current solution" and $Max(H^\alpha, C)$ as the "current objective function value".

Step 3: (Shake operation begins if #S>0) Solve the TSP on the cost matrix $C^{\delta,R}$ using α and let H^α be the resulting solution. If $Sum(H^\alpha, C^\delta) = 0$ Stop. H^α is an optimal solution to BTSP. Otherwise update "current solution" and "current objective function value" if necessary and repeat this step #S times or until some other termination criterion is satisfied. (Note that the matrices $C^{\delta,R}$ generated in each repetition are different with high probability.)

Clearly, if $Sum(H^\alpha, C^\delta) = 0$ then H^α is an optimal solution to the BTSP. Our experimental results show that this is indeed the case for many of the test problems we have considered. But if $Sum(H^\alpha, C^\delta) \neq 0$, then H^α uses some edges of non-zero costs. In the ST-algorithm we perform a shake operations by choosing different random numbers for the non-zero values of C^δ (i.e. by using $C^{\delta,R}$) to see if $Sum(H^\alpha, C^\delta) = 0$. Ideally we would like to make the largest value of such non-zero costs in H^α as small as possible so that an improved bottleneck objective value is achieved. We try to achieve this using "controlled shake operations" discussed below. Let $z_1 < z_2 < \ldots < z_u$ be an ascending arrangement of the distinct edge costs c_{ij} such that $c_{ij} > \delta$. Let $r_1 < r_2 < \cdots < r_u$ be random numbers generated from the interval $[a, b]$ for some a and b. Consider the matrix

$$c_{ij}^{\delta,[a,b]} = \begin{cases} 0 & \text{if } c_{ij} \leq \delta \\ z_s + r_s & \text{if } c_{ij} = z_s \end{cases} \tag{5}$$

It can be verified that the BTSP with cost matrix C and the BTSP with cost matrix $C^{\delta,[a,b]}$ have the same set of optimal solutions. In the simple threshold algorithm we could use $C^{\delta,[a,b]}$ instead of $C^{\delta,R}$. The advantage of using $C^{\delta,[a,b]}$ is that the difference between consecutive non-zero values of elements of $C^{\delta,[a,b]}$ is larger compared to corresponding values of C^δ. This in turn encourages the TSP heuristic α when applied on $C^{\delta,[a,b]}$ to pick smaller non-zero values and hence a solution with better bottleneck objective function value could result. The built in randomness allows us to generate different matrices having this desirable property and could mimic the effect of a shake operation with additional advantages. Thus we call the operation of using α on the matrix $C^{\delta,[a,b]}$ a *type 1 controlled shake*, and the parameter #CS1 denotes the number of type 1 controlled shakes permitted.

We have also tested using another cost matrix defined as follows. Let U be an upper bound on the optimal objective function value of the BTSP and $z_1 < z_2 < \ldots < z_p$ be an ascending arrangement of the distinct edge costs c_{ij}, $\delta < c_{ij} \leq U$. Define $d_i = z_i + R_i$ where R_i is a random number in the interval $(R_{i-1} + \theta, R_{i-1} + \tau)$ and θ and τ $(\theta < \tau)$ are prescribed parameters. Consider the cost matrix $C^{\delta,\theta,\tau} = (c_{ij}^{\delta,\theta,\tau})$ defined by

$$c_{ij}^{\delta,\theta,\tau} = \begin{cases} 0 & \text{if } c_{ij} \leq \delta \\ d_r & \text{if } c_{ij} = z_r \text{ and } c_{ij} \leq U \\ M & \text{otherwise} \end{cases} \tag{6}$$

where M is a large number. Note that the value of M is selected such that it is significantly larger than d_p. The matrix $C^{\delta,\theta,\tau}$ has similar properties as $C^{\delta,[a,b]}$ with more control on the random numbers. The shake operation performed using $C^{\delta,\theta,\tau}$ is called *type 2 controlled shake* and the parameter #CS2 denotes the number of type II controlled shakes performed. Using these controlled shake operations we obtain an enhanced version of our ST-heuristic which we call *enhanced threshold heuristic* or ET-heuristic. An informal description of the ET-heuristic is given below.

ET-Heuristic

Step 1: Set the values of the control parameters θ, τ, a, b, #CS1 and #CS2

Step 2: Compute the lower bound δ (possibly by solving the BBSSP)

Step 3: Use a TSP heuristic α to solve TSP on the cost matrix C^{δ}. (For example, use some recent variations of the Lin-Kernighan heuristic [1, 16, 22].) Let H^{α} be the solution produced. Set H^{α} as the "current solution" and $Max(H^{\alpha}, C)$ as the current objective function value. If $Sum(H^{\alpha}, C^{\delta}) = 0$ Stop. H^{α} is an optimal solution to the BTSP. Otherwise designate H^{α} as the "current solution" and $Max(H^{\alpha}, C)$ as the "current objective function value".

Step 4: (Type 1 controlled shake begins if #CS1> 0) Solve the TSP on the cost matrix $C^{\delta,[a,b]}$ using α and let H^{α} be the resulting solution. If $Sum(H^{\alpha}, C^{\delta}) = 0$ Stop. H^{α} is an optimal solution to BTSP. Otherwise update "current solution" and "current objective function value" if necessary and repeat this step #CS1 times or until some other termination criterion is satisfied. (Note that the matrices $C^{\delta,[a,b]}$ generated in each repetition are different with high probability.)

Step 5: (Type 2 controlled shake operation begins if #CS2 > 0) Solve a TSP on the cost matrix $C^{\delta,\theta,\tau}$ using α and let H^{α} be the resulting solution. If $Sum(H^{\alpha}, C^{\delta}) = 0$ Stop. H^{α} is an optimal solution to BTSP. Otherwise update "current solution" and "current objective function value" if necessary and repeat this step #CS2 times until some other termination criterion is satisfied. (Note that the matrices

$C^{\delta,\theta,\tau}$ generated in each repetition are different with high probability in each repetition.)

Step 6: Output "current solution"

2.4. **Enhanced binary search algorithm.** By choosing δ as any real number (instead of a lower bound to BOBJ(C)) in the ET-heuristic we can (heuristically) solve the δ-feasibility problem. Thus, the ET-heuristic can be used repeatedly in a binary search fashion among possible candidates for BOBJ(C). Let ETH(δ) denotes the algorithm where Step 2 of the ET-heuristic is removed and δ is considered as an input parameter. We also use the ET-heuristic to obtain a heuristic solution, say H, to calculate an upper bound $U = Max(H, C)$ for BOBJ(C). Let L be a lower bound for BOBJ(C). If $L = U$, then H is an optimal solution to the BTSP. Otherwise $L \leq BOBJ(C) \leq U$. Choose δ as the median of the set $\{c_{ij} | L \leq c_{ij} < U\}$ and call ETH(δ) to test (heuristically) if there is tour H in G such that $Max(H, C) \leq \delta$. If the answer is "yes", we have an improved solution, say H^*. Then the upper bound can be updated as $U = Max(H^*, C)$. Note that this value will be less than or equal to δ. (Whenever we update the upper bound U, we also update the solution associated with it.) If the answer is "no" we conclude (heuristically) that there is no solution with bottleneck objective function value smaller than δ and update $L = \delta$. The process is continued until $\{c_{ij} | L \leq c_{ij} < U\}$ becomes empty. At this stage the current solution has bottleneck objective function value L.

In the binary search phase, we also use some special updates for the lower bound. Note that when an upper bound is updated at any iteration, it continues to remain a valid upper bound for BOBJ(C). However, when the lower bound L is updated, we make a heuristic decision. The updated L may not be a lower bound for BOBJ(C), although the probability of this happening is less if α is powerful. Nonetheless, we need to take care of this possibility of ETH(δ) generating a solution H^* at some iteration with $Max(H^*, C) < L$. If this happens, we reset L back to the original lower bound obtained in the beginning of the algorithm and set $U = Max(H^*, C)$. We never observed this phenomenon and this was not included in our final code. However, we keep this step in the description of our primary algorithm for completeness.

The foregoing discussions lead to our main algorithm to solve BTSP which is called the *enhanced binary search threshold algorithm* or EBST-algorithm. A formal description of the algorithm is given in Figure 1 (as Algorithm 1 EBST-Algorithm). Note that ETH(δ) is used to solve the δ-feasibility problem within the algorithm. Thus in the description of the algorithm we assume that ETH(δ) produce a solution H such that if $Max(H, C) \leq \delta$ we have an "yes" answer to the δ-feasibility problem and a no answer otherwise.

Algorithm 1 EBST-Algorithm

Solve the bottleneck biconnected spanning subgraph problem on G with cost matrix C. Let δ be the objective function value produced.
Call ETH(δ). Let H be the resulting tour. {* *ETH(δ) also produced* FLAG. If FLAG = YES, *the algorithm is terminated and we have an optimal solution.* *}
Tour $\Leftarrow H$, OBJ $\Leftarrow \max_{ij\in H}\{c_{ij}\}$
Let $z_1 < z_2 < \cdots < z_p$ be an ascending arrangement of all distinct values of the set $\{c_{ij} : \delta \le c_{ij} < OBJ\}$. $\ell \Leftarrow 1$, $u \Leftarrow p$
while $u - \ell > 1$ **do**
  $k \Leftarrow \lfloor \frac{\ell+u}{2} \rfloor$, $\delta = z_k$
  Call ETH(δ). Let H be the output tour.
  if FLAG = "yes" **then**
    {* *update the upper bound* *}
    Choose q such that $z_q = \max_{ij\in H}\{c_{ij}\}$
    $u \Leftarrow q$, OBJ $\Leftarrow z_u$, Tour $\Leftarrow H$
    {* special reset *}
    if $q \le \ell$ **then**
      $\ell \Leftarrow 1$, $u \Leftarrow q$
    end if
  else
    {* update the lowerbound*}
    $\ell \Leftarrow k + 1$
    {* special update of the upperbound *}
    if $\max_{ij\in H} c_{ij} < z_u$ **then**
      choose q such that $z_q = \max_{ij\in H} c_{ij}$
      $u \Leftarrow q$
    end if
  end if
end while
output Tour and OBJ

The performance of the algorithm depends on various parameters such as #CS1,#CS2, θ, a, b and τ. Extensive computational experiments have been carried out to fine tune the values of these parameters. Considering the outcomes of these experiments and using our intuition we have decided to set the default values of these parameters are set to #CS1 = 3, #CS2 = 3, $a = 1$, $b = n$, $\theta = 5$ and $\tau = 50$. No claim is made that these are the best values. In fact for hard problems, one may try to change the parameter values to explore improved solutions. Nevertheless, for definiteness we decided that it is important to give definite values for all parameters. We have used the implementation of the LK-algorithm in the Concorde Library [1] as our TSP heuristic. It may be noted that a simple binary search algorithm using Helsgaun's implementation of the LK-algorithm [12] in place of ETH(δ)

has been used by Ramakrishnan et al [30] and reported good experimental results on TSPLIB problems of size up to 783 cities. Their algorithm is a special case of our EBTS-algorithm by setting #CS1 = #CS2 = 0, and using Helsgaun's implementation of LK-algorithm in ETH(δ), the 2-max lower bound [26, 30] is used in place of the superior BBSSP lower bound, and using the NNH solution to compute the starting upper bound. The heuristic reported in [30] considered only 72 problems all, from the TSPLIB with size no more that 783. Optimality was guaranteed for 37 problems. From our experiments, their solutions for the remaining 35 problems were also optimal. Our algorithm also obtained optimal solutions to these problems, often without entering into the binary search phase, except for one problem.

3. Computational Results for BTSP

The algorithms discussed in the previous section were coded in ANSI C and tested using the GNU GCC compiler version 3.2 under Redhat Linux 8.0. Two classes of test problems are considered; standard benchmark problems and specially constructed hard problems. All the experiments with standard benchmark problems were carried out on the 164-processor Sun V60 clustered computer, Chorus at the University of New Brunswick, Fredericton. Chorus consists of 60 slave nodes of dual 2.8GHz Intel Xeon processors with 2 to 3 GB of RAM. Detailed information about the cluster can be found on UNB's ACRL site [39]. We did not use the multi-processing capabilities of the workstation; all tests were carried out using a single processor. The specially constructed hard problems were tested on a Dell Precision workstation with 2.0 GHz Intel Xeon Processor and 512 MB of memory. We used the Lin-kernighan implementation of Concorde Library [1] for solving all TSPs within our algorithms. Computational time was measured in seconds of CPU time corrected to two decimal digits. This does not include input and output times. All computational times reported for benchmark problems are average over 10 runs. Also, whenever 'best objective function value' is indicated, it is based on these 10 independent runs. We have done these 10 repetitions to investigate the effect of randomness within the algorithm on solution quality as well as running time and found no significant effect. The purposes of our preliminary experiments are to assess the performance of the algorithms on various classes of test problems and to fine-tune the parameters to set their default values.

The test problems we considered can be divided into five different categories: (1) TSPLIB problems [33], (2) Random problems of Johnson and McGeoch [16, 14], (3) Specially designed 'hard' problems, (4) National TSP instances [37] and (5) VLSI instances [37]. TSPLIB is a well known collection of TSP instances selected from practical applications and primarily used as a test bed for TSP algorithms. Our heuristics obtained optimal solutions for all TSPLIB problems except two very large ones with size more than 35,000 nodes. These very large problems were not tested due to memory

limitations. We call the test problems in class 2 the *JM-instances*. These instances were introduced by Johnson and McGeoch [16]; again, for testing TSP algorithms. The JM-instances are classified into three different groups: uniform point, clustered points, and random distance matrices. For most of the problems in classes (1) and (2) the BBSSP lower bound turned out to be the optimal objective function value. Thus, we have generated a special class of problems which do not have this property. The cost matrices in this class have a partitioned structure. See Figure 1.

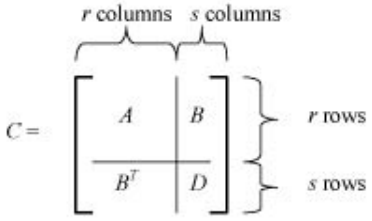

FIGURE 1. Structure of the special cost matrix

Here A is a symmetric $r \times r$ matrix with entries in the range $[\beta + 1, \gamma]$; B is a $r \times s$ matrix with entries in the range $[0, \beta]$; D is a $s \times s$ symmetric matrix with entries in the range $[\beta + 1, \gamma]$; and $r \geq 2$, $s \geq 2$, $r + s = n$. We generated problem instances from 100 to 2500 nodes for various values of β, γ, and r. Unless the dimensions of A and B are equal, this class of problems assures that the BBSSP lower bound is strictly less than the optimal BTSP objective function value. In fact the BBSSP lower bound will be no more that β and the optimal BTSP objective function value will be at least $\beta + 1$.

Our first set of experiments were performed on TSPLIB problems of size up to 7397 nodes to assess the relative strength of 2-max bound and BBSSP bound. The BBSSP lower bound was optimal for all problems except ts225 and the 2-max bound was optimal for 52 out of the 101 problems tested. The computational time for 2-max bound was, however, lower. Both algorithms took less than 0.01 seconds for problems within 225 nodes. The time for 2-max bound gradually increased to 0.15 seconds as problem size increased to 4461 nodes, whereas for BBSSP bound the computational time increased to 7.68 seconds. As the number of nodes increased to 7397 BBSSP computational time increased to 48.39 seconds but the time for 2-max bound was only 0.37 seconds. Because of the superior quality of the lower bound and its reasonable computational time, we have decided to use BBSSP lower bound in our experiments. Experimental results on these lower bound heuristics are reported in Table 1. For larger problems, the time bound for computation of this lower bound could be reduced by using the

linear time algorithm of Manku [23] or Punnen and Nair [29]. Our implementation used the $O(n^2 \log n)$ threshold algorithm discussed in Parker and Rardin [26].

Problem	2-max bound			BBSSP bound		
	LB	%-gap	Time	LB	%-gap	Time
ts225	500.00	50.00	0.00	500.00	50.00	0.01
att532	227.00	0.87	0.00	229.00	0.00	0.10
ali535	3889.00	0.00	0.00	3889.00	0.00	0.12
si535	186.00	18.06	0.00	227.00	0.00	0.06
pa561	16.00	0.00	0.00	16.00	0.00	0.08
u574	345.00	0.00	0.00	345.00	0.00	0.12
rat575	23.00	0.00	0.00	23.00	0.00	0.11
p654	934.00	23.63	0.00	1223.00	0.00	0.17
d657	1368.00	0.00	0.00	1368.00	0.00	0.13
gr666	4264.00	0.00	0.00	4264.00	0.00	0.20
u724	152.00	10.59	0.00	170.00	0.00	0.14
rat783	26.00	0.00	0.01	26.00	0.00	0.14
dsj1000	92501.00	68.74	0.01	295939.00	0.00	0.84
pr1002	1662.00	21.94	0.01	2129.00	0.00	0.43
si1032	107.00	70.44	0.01	362.00	0.00	0.27
u1060	2378.00	0.00	0.01	2378.00	0.00	0.73
pcb1173	243.00	0.00	0.01	243.00	0.00	0.61
d1291	1289.00	0.00	0.01	1289.00	0.00	0.78
rl1304	789.00	48.60	0.01	1535.00	0.00	0.62
rl1323	1905.00	23.46	0.01	2489.00	0.00	0.66
nrw1379	105.00	0.00	0.02	105.00	0.00	0.56
fl1400	530.00	0.00	0.01	530.00	0.00	0.52
u1432	300.00	0.00	0.02	300.00	0.00	1.06
fl1577	137.00	68.21	0.02	431.00	0.00	1.58
d1655	1476.00	0.00	0.02	1476.00	0.00	1.38
u1817	234.00	0.00	0.02	234.00	0.00	1.54
rl1889	752.00	16.07	0.02	896.00	0.00	1.11
d2103	1133.00	0.00	0.03	1133.00	0.00	1.52
u2152	105.00	0.00	0.03	105.00	0.00	2.66
u2319	224.00	0.00	0.04	224.00	0.00	2.26
pr2392	401.00	16.63	0.04	481.00	0.00	4.10
pcb3038	181.00	8.59	0.07	198.00	0.00	5.68
fl3795	528.00	0.00	0.10	528.00	0.00	3.57
fnl4461	132.00	0.00	0.15	132.00	0.00	7.68
rl5915	580.00	3.65	0.24	602.00	0.00	12.24
rl5934	533.00	40.51	0.24	896.00	0.00	10.47
pla7397	69772.00	14.33	0.37	81438.00	0.00	48.39

TABLE 1. TSPLIB - Comparison of lower bounds

Problem	Size	Optimal Value	NNH			BNNH			ENNH			BNIH		
			obj. value	%-gap	Time	obj. value	%-gap	Time	obj. value	%-gap	Time	obj. value	%-gap	Time
pr1002	1002	2129	5,764.50	170.76	0.18	6,381.70	199.75	0.20	3,446.50	61.88	14.29	4,254.70	99.84	1.00
si1032	1032	362	427.80	18.18	0.20	411.20	13.59	0.20	395.00	9.12	15.86	394.20	8.90	1.04
u1060	1060	2378	7,133.70	199.99	0.20	6,936.60	191.70	0.20	4,280.80	80.02	16.87	3,662.80	54.03	1.10
pcb1173	1173	243	1,053.70	333.62	0.20	1,283.50	428.19	0.20	763.00	213.99	22.96	774.60	218.77	1.46
d1291	1291	1289	2,872.90	122.88	0.21	2,499.50	93.91	0.28	1,479.40	14.77	30.36	1,327.10	2.96	1.68
rl1304	1304	1535	5,401.00	251.86	0.20	6,485.40	322.50	0.30	3,884.60	153.07	31.02	4,415.00	187.62	1.80
rl1323	1323	2489	5,616.40	125.65	0.24	7,149.50	187.24	0.30	3,926.50	57.75	33.14	4,440.70	78.41	1.90
nrw1379	1379	105	812.50	673.81	0.30	1,040.30	890.76	0.30	568.80	441.71	37.22	659.30	527.90	2.18
fl1400	1400	530	1,570.80	196.38	0.30	1,180.30	122.70	0.30	1,050.40	98.19	38.30	905.30	70.81	2.10
u1432	1432	300	2,635.60	778.53	0.30	2,480.50	726.83	0.30	1,438.00	379.33	41.29	1,937.40	545.80	3.45
fl1577	1577	431	898.50	108.47	0.34	968.50	124.71	0.39	559.60	29.84	55.00	664.60	54.20	2.80
d1655	1655	1476	2,881.20	95.20	0.40	2,313.70	56.75	0.40	1,629.10	10.37	63.19	1,489.10	0.89	2.98
u1817	1817	234	1,063.70	354.57	0.50	1,152.80	392.65	0.50	642.20	174.44	84.82	740.20	216.32	4.18
rl1889	1889	896	6,167.30	588.31	0.50	6,921.20	672.46	0.50	3,806.40	324.82	93.77	4,502.20	402.48	4.16
d2103	2103	1133	2,153.40	90.06	0.66	2,139.50	88.83	0.70	1,572.80	38.82	131.23	1,207.20	6.55	5.81
u2152	2152	105	1,052.60	902.48	0.68	1,004.60	856.76	0.70	590.80	462.67	140.68	752.80	616.95	5.65
u2319	2319	224	2,567.50	1,046.21	0.80	2,368.70	957.46	0.80	1,402.50	526.12	179.05	1,800.20	703.66	6.77
pr2392	2392	481	5,462.20	1,035.59	0.81	5,852.00	1,116.63	0.89	3,292.40	584.49	193.16	4,044.90	740.94	7.24
pcb3038	3038	198	1,748.80	783.23	1.35	1,580.40	698.18	1.38	1,015.40	412.83	396.11	1,118.60	464.95	13.65
fl3795	3795	528	1,285.90	143.54	2.03	1,341.70	154.11	2.08	818.50	55.02	750.58	698.20	32.23	23.78
fnl4461	4461	132	1,790.30	1,256.29	2.79	1,716.20	1,200.15	2.97	912.80	591.52	1,230.14	1,148.70	770.23	35.36
rl5915	5915	602	6,489.00	977.91	4.88	6,761.70	1,023.21	5.10	3,688.70	512.74	2,858.90	4,398.70	630.68	72.95
rl5934	5934	896	6,336.20	607.17	4.90	6,935.10	674.01	5.17	3,736.20	316.99	2,882.75	4,482.00	400.22	73.92
pla7397	7397	81438	304,004.40	273.30	7.55	295,798.80	263.22	8.08	156,139.20	91.73	5,549.28	236,969.60	190.98	131.94

TABLE 2. TSPLIB - Results for construction heuristics

Let us now discuss experimental results on NNH, BNNH, ENNH, and BNIH heuristics. Table 2 summarizes the results of these experiments for midsize problems in our test set. In the table %-gap refers to the quantity $\frac{(H-LB)}{LB}100$ where H is the heuristic value and LB is the lower bound value. As the data indicates, the quality of these solutions are not very impressive but computational times are low except for ENNH. Due to poor quality of solutions, we decided not to test these heuristics on other classes of test problems. However, if one needs to select a construction heuristic, we recommend BNIH.

We now consider our ST and ET-heuristics. Preliminary experiments have been conducted using TSPLIB problems to set the values of the parameters #S, #CS1, #CS2, a, b, τ, and θ. We first considered ST-Heuristic with no shake operation and a single shake, 3 shakes, and 5 shakes. The random numbers for $C^{\delta,R}$ are selected from the interval $[1, n^2]$. Problems up to size 2103 produced provably optimal solution without shake except ts225. Five more problems did not produce optimal solutions for ST-heuristic with no shakes. Computational results on these problems including ts225 are reported in Table 3.

Problem	No Shake		one Shake		3 Shakes		5 Shakes	
	%-gap	Time	%-gap	Time	%-gap	Time	%-gap	Time
ts225	253.6	0.12	253.60	0.50	253.60	1.30	253.60	2.20
u2152	2.10	23.11	0.00	21.27	0.00	55.23	0.00	18.73
pr2392	3.37	21.62	0.60	27.00	0.00	21.50	3.01	203.56
rl5915	12.51	46.24	0.50	95.15	0.50	194.51	3.80	433.90
rl5934	8.78	48.03	0.00	44.14	0.00	43.99	0.00	47.15
pla7397	40.21	64.72	35.51	101.50	67.02	277.02	53.61	347.75

TABLE 3. TSPLIB instances with ST-heuristic: Effect of shake operations. %-gap = $\frac{(H-LB)100}{LB}$ where H is the heuristic solution value and LB is the BBSSP lower bound value.

Although additional shake operations did produce optimal solutions in some cases, random shaking did not perform consistently better. (See Table 3.) In some cases, 5 shakes produced worse solutions than one shake. We believe this is dues to the randomness in our algorithm and possibly in the LK algorithm as well. Let us now look at the effect of controlled shaking used by the ET-heuristic. We initially set #CS2 = 0 to assess the effect of type 1 controlled shake. The value of a is set to 1 and b is set to n. We considered three different values, 1, 3, and 5, for #CS1 and obtained optimal solutions to all but 3 problems (pla7397, rl5915, ts225) for #CS1 = 1 and 3. Further the accuracy was better for #CS1 = 3 for problems that were not solved to optimality. For #CS1 = 5, we obtained optimal solution for all but two problems (ts225,pla7397). Considering the computational time and solution quantity, we believe #CS1 = 3 could be a reasonable choice if #CS2 = 0 is selected. These results are summarized in Table 4

Problem	one shake		3 shakes		5 Shakes	
	%-gap	Time	%-gap	Time	%-gap	Time
ts225	182.8	0.30	164.41	0.64	190.32	0.95
u2152	0.00	23.68	0.00	30.08	0.00	28.08
pr2392	0.00	23.15	0.00	28.60	0.00	34.04
rl5915	9.70	113.51	3.21	153.45	0.00	162.96
rl5934	0.00	52.08	0.00	55.25	0.00	54.10
pla7397	26.81	211.62	20.10	254.21	33.51	451.67

TABLE 4. TSPLIB instances ET-heuristic with only Type I controlled shake. %-gap $= \frac{(H-LB)100}{LB}$ where H is the heuristic solution value and LB is the BBSSP lower bound value.

Similar experiments were conducted using #CS1 = 0 and #CS2 = 1, 3, and 5. We also examined the effect of varying the values of $\{\theta, \tau\}$ as {5,10}, {10,25}, {5,100}, and {50,100}. No significant difference in quality of solutions is observed. #CS2 = 3 and $\{\theta, \tau\} = \{10,25\}$ appeared a good choice, although no significant difference is observed in solution quality for other choices of the parameters. It solved optimally all but 3 problems (ts225, rl5915,pla7397). Table 5 shows results of these experiments.

Problem	$\theta = 5, \tau = 10$		$\theta = 10, \tau = 25$		$\theta = 5, \tau = 100$		$\theta = 50, \tau = 100$	
	%-gap	Time	%-gap	Time	%-gap	Time	%-gap	Time
ts225	253.6	0.40	253.60	0.40	253.60	0.45	253.6	0.46
u2152	0.00	42.74	0.00	47.12	2.10	34.91	0.00	32.69
pr2392	0.00	33.23	0.00	39.01	0.00	34.32	0.00	49.34
rl5915	9.82	196.63	9.70	232.78	9.65	145.64	49.45	214.76
rl5934	0.00	58.11	0.00	51.22	0.00	56.33	8.78	75.81
pla7397	33.51	229.96	40.21	238.55	46.91	257.63	40.21	265.27

TABLE 5. TSPLIB instances ET-heuristic with only Type II controlled shakes(3 shakes). %-gap $= \frac{(H-LB)100}{LB}$ where H is the heuristic solution value and LB is the BBSSP lower bound value.

The result in this experiment was not as good as that in Type I controlled shake. We believe that this might be due to the usage of M for elements in the shake matrix corresponding to $c_{ij} > U$. Thus it may be reasonable to avoid using M and choose controlled random numbers of value more than M. No harm is done with this change and used this strategy in our final experiments. Finally, we have conducted experiments with the EBST-heuristic. In one case, for the corresponding ETH(δ) we set #CS1 = 0 and in the other case for the corresponding ETH(δ) we set #CS2 = 0. The values of

the other parameters are set as follows: $a = 1, b = n$, $\tau = 25$, and $\theta = 10$. For the run with #CS1 = 0 we obtained optimal solutions for all problems tested from the TSPLIB except ts225, and for the run with #CS2= 0 we obtained optimal solutions for all but two problems with very minimal % deviation from optimum (.08% and .55%). The computation times for the two runs are given in Tables 6 and 7.

Problem	Size	Optimal Value	Type I			Type 2		
			obj. value	%-gap	Time	obj. value	%-gap	Time
ts225	225	1000	1,000.00	0.00	8.67	1,000.00	0.00	11.62
d2103	2103	1133	1,133.00	0.00	42.29	1,133.00	0.00	42.29
u2152	2152	105	105.00	0.00	44.06	105.00	0.00	45.35
u2319	2319	224	224.00	0.00	24.89	224.00	0.00	24.80
pr2392	2392	481	481.00	0.00	30.10	481.00	0.00	51.18
pcb3038	3038	198	198.00	0.00	32.91	198.00	0.00	32.83
fl3795	3795	528	528.00	0.00	52.45	528.00	0.00	52.22
fnl4461	4461	132	132.00	0.00	46.85	132.00	0.00	46.68
rl5915	5915	602	602.50	0.08	197.15	602.00	0.00	163.96
rl5934	5934	896	896.00	0.00	66.52	896.00	0.00	70.59
pla7397	7397	81438	81,884.90	0.55	737.69	81,438.00	0.00	418.32

TABLE 6. EBST heuristic for TSPLIB problems. %-gap = $\frac{(H-LB)100}{LB}$ where H is the heuristic solution value and LB is the BBSSP lower bound value.

Problem size range	Number of problems	Type I Avg. Time	Type II Avg. Time
14-99	22	0.25	0.25
100-198	25	1.39	1.41
200-318	11	3.27	3.33
400-493	6	9.92	10.06
532-575	6	11.06	10.97
654-783	5	16.33	16.03
1000-1291	6	29.79	30.14
1304-1432	5	25.45	25.41
1577-1889	4	31.65	30.62

TABLE 7. EBST heuristic results for grouped TSPLIB problems - Average computational time. %-gap = $\frac{(H-LB)100}{LB}$ where H is the heuristic solution value and LB is the BBSSP lower bound value.

Problem	Size	Type I shake			Type II shake		
		Av. Sol.	Best Sol.	Av. Time	Av. Sol	Best Sol.	Av. Time
C1k.0	1000	290552.00	290552.00	32.95	290552.00	290552.00	31.86
C1k.1	1000	335184.00	335184.00	34.56	335184.00	335184.00	34.23
C1k.2	1000	225295.00	225295.00	29.68	225295.00	225295.00	30.60
C1k.3	1000	416768.00	416768.00	41.93	416768.00	416768.00	37.19
C1k.4	1000	318930.00	318930.00	34.09	318930.00	318930.00	31.95
C1k.5	1000	260389.00	260389.00	47.43	260389.00	260389.00	57.74
C1k.6	1000	175740.00	175740.00	32.18	175740.00	175740.00	32.15
C1k.7	1000	301366.00	301366.00	36.01	301366.00	301366.00	35.94
C1k.8	1000	246519.00	246519.00	32.84	246519.00	246519.00	31.61
C1k.9	1000	208091.00	208091.00	30.10	208091.00	208091.00	30.22
C3k.0	3162	252245.00	252245.00	69.17	252245.00	252245.00	81.19
C3k.1	3162	167466.00	167466.00	102.30	167466.00	167466.00	68.28
C3k.2	3162	194007.00	194007.00	67.67	194007.00	194007.00	67.75
C3k.3	3162	180852.00	180852.00	72.52	180852.00	180852.00	79.93
C3k.4	3162	180583.00	180583.00	67.29	180583.00	180583.00	67.58
C10k.0	10000	161062.00	161062.00	376.47	161062.00	161062.00	325.26
C10k.1	10000	100349.00	94139.00	33354.23	100333.20	94139.00	5945.74
C10k.2	10000	121209.00	121209.00	430.84	121209.00	121209.00	343.23

TABLE 8. JM - Instances Type C. All solutions are optimal and their values are equal to the BBSSP bound

Thus EBST-heuristic clearly very effective in obtaining good quality solutions in reasonable running time. We have also tested our algorithm on other classes of problems with type I and type II shakes separately. Table 8 report experimental results with JM-instances of clustered type with size ranging from 1000 to 10000 nodes. All averages hereafter are taken over 10 trials. In the table, the columns corresponding to type I shake, we set #CS2 =0 and other parameters have default values. Likewise, the columns corresponding to type II shake, we set #CS1 = 0 and other parameters have default values. The running times were almost uniform except for C10K.1 which is an outlier. Table 9 represents similar data on JM-instances of Euclidian and random types. All the solutions obtained for this class are guaranteed to be optimal and computational times have a uniform behavior.

Taking guidance from previous experiments, we tested specially structured hard instances with parameter values set as #CS1 = 3, #CS2 = 3, $a = 1$, $b = n$, $\theta = 5$ and $\tau = 50$. The parameters β, γ and r used to generate the test problems are given along with the seed value to initialize the random number generator. When $r = n/2$ the BBSSP lower bound may coincide with the optimal objective function value. For all other values of r the

Problem	Size	Type I shake			Type II shake		
		Av. Sol.	Best Sol.	Av. Time	Av. Sol	Best Sol.	Av. Time
E1k.0	1000	64739.00	64739.00	16.87	64739.00	64739.00	16.90
E1k.1	1000	67476.00	67476.00	18.05	67476.00	67476.00	17.98
E1k.2	1000	88522.00	88522.00	22.67	88522.00	88522.00	22.36
E1k.3	1000	59220.00	59220.00	15.86	59220.00	59220.00	15.84
E1k.4	1000	68259.00	68259.00	17.97	68259.00	68259.00	17.91
E1k.5	1000	61406.00	61406.00	16.07	61406.00	61406.00	15.99
E1k.6	1000	68777.00	68777.00	18.81	68777.00	68777.00	18.65
E1k.7	1000	70389.00	70389.00	18.71	70389.00	70389.00	18.60
E1k.8	1000	57597.00	57597.00	15.20	57597.00	57597.00	15.13
E1k.9	1000	68420.00	68420.00	18.30	68420.00	68420.00	18.26
E3k.0	3162	39854.00	39854.00	41.96	39854.00	39854.00	41.93
E3k.1	3162	37500.00	37500.00	39.37	37500.00	37500.00	39.59
E3k.2	3162	35145.00	35145.00	38.39	35145.00	35145.00	38.56
E3k.3	3162	44428.00	44428.00	43.92	44428.00	44428.00	43.98
E3k.4	3162	36621.00	36621.00	39.36	36621.00	36621.00	39.15
E10k.0	10000	20174.00	20174.00	239.55	20174.00	20174.00	239.07
E10k.1	10000	22883.00	22883.00	243.48	22883.00	22883.00	243.08
E10k.2	10000	20208.00	20208.00	237.07	20208.00	20208.00	237.03
M1k.0	1000	9328.00	9328.00	52.21	9328.00	9328.00	52.39
M1k.1	1000	8856.00	8856.00	51.71	8856.00	8856.00	51.46
M1k.2	1000	11282.00	11282.00	55.36	11282.00	11282.00	54.92
M1k.3	1000	11617.00	11617.00	53.94	11617.00	11617.00	53.91
M3k.0	3162	3289.00	3289.00	129.47	3289.00	3289.00	130.00
M3k.1	3162	3034.00	3034.00	128.01	3034.00	3034.00	127.84
M10k.0	10000	1189.00	1189.00	536.46	1189.00	1189.00	534.94

TABLE 9. JM-Instances of E and M types. All solutions are optimal and their values are equal to the BBSSP bound

bound is guaranteed to be strictly lower than the BTSP optimal objective function value. Table 10 summarizes results on specially structured hard instances with 500 nodes. All solutions obtained were optimal. Table 11 report similar results on 2500 nodes. We obtained optimal solutions for 20 out of 27 problems. The running time for these hard instances are higher compared to other classes of problems. We have also tested hard instance with 100 nodes (table not included for these results) and obtained optimal solutions for all. The optimality of these instances were verified by using a modified version of our algorithm with an exact TSP solver.

beta	gamma	r	Seed	Obj. Value	Optimal?	BBSSP LB	BBSSP Time	LK Calls	Avg. LK Time	Total Time
500	5000	125	6125	525	Yes	29	0.07	57	108.05	6190.58
500	5000	250	6250	17	Yes	17	0.08	1	5.86	5.94
500	5000	375	6375	524	Yes	35	0.08	50	109.19	5491.03
500	7500	125	8625	538	Yes	31	0.08	58	97.84	5709.70
500	7500	250	8750	22	Yes	22	0.08	1	5.68	5.77
500	7500	375	8875	537	Yes	36	0.08	58	90.37	5272.00
500	10000	125	11125	547	Yes	53	0.09	52	90.77	4749.93
500	10000	250	11250	20	Yes	20	0.08	1	5.88	5.98
500	10000	375	11375	551	Yes	30	0.10	60	90.71	5471.06
1000	5000	125	6625	1022	Yes	56	0.08	65	100.55	6573.59
1000	5000	250	6750	26	Yes	26	0.08	1	5.50	5.60
1000	5000	375	6875	1019	Yes	73	0.08	56	107.78	6061.48
1000	7500	125	9125	1031	Yes	59	0.08	65	102.46	6697.39
1000	7500	250	9250	40	Yes	40	0.08	1	5.91	5.99
1000	7500	375	9375	1034	Yes	55	0.08	65	97.96	6396.32
1000	10000	125	11625	1052	Yes	70	0.08	64	88.71	5721.51
1000	10000	250	11750	34	Yes	34	0.09	1	5.88	5.98
1000	10000	375	11875	1047	Yes	99	0.08	64	82.80	5336.26
2500	5000	125	8125	2514	Yes	175	0.08	72	95.30	6897.77
2500	5000	250	8250	88	Yes	88	0.07	1	6.10	6.18
2500	5000	375	8375	2513	Yes	156	0.06	78	90.44	7088.92
2500	7500	125	10625	2530	Yes	174	0.08	63	101.75	6446.26
2500	7500	250	10750	106	Yes	106	0.08	1	5.69	5.79
2500	7500	375	10875	2528	Yes	170	0.08	71	91.07	6493.24
2500	10000	125	13125	2534	Yes	171	0.08	79	86.61	6874.99
2500	10000	250	13250	83	Yes	83	0.08	1	6.02	6.11
2500	10000	375	13375	2536	Yes	161	0.08	64	101.73	6539.55

TABLE 10. Hard instances with $n = 500$

beta	gamma	r	Seed	Obj. Value	Optimal?	BBSSP LB	BBSSP Time	LK Calls	Avg. LK Time	Total Time
500	5000	625	8625	505	???	7	4.16	64	377.10	24305.76
500	5000	1250	9250	4	Yes	4	3.80	1	27.82	32.01
500	5000	1875	9875	505	Yes	9	3.90	54	400.87	21989.61
500	7500	625	11125	508	Yes	8	4.35	64	361.42	23334.92
500	7500	1250	11750	4	Yes	4	3.97	1	28.18	32.55
500	7500	1875	12375	507	???	8	4.20	58	389.32	22762.23
500	10000	625	13625	510	???	6	4.91	58	408.11	23854.55
500	10000	1250	14250	4	Yes	4	4.13	1	27.94	32.44
500	10000	1875	14875	510	Yes	7	4.25	64	367.58	23708.87
1000	5000	625	9125	1005	Yes	16	4.63	71	352.41	25242.82
1000	5000	1250	9750	6	Yes	6	4.27	1	31.18	35.85
1000	5000	1875	10375	1005	Yes	16	4.37	71	351.78	25255.14
1000	7500	625	11625	1007	Yes	14	4.08	70	362.80	25724.98
1000	7500	1250	12250	8	Yes	8	4.03	1	30.41	34.84
1000	7500	1875	12875	1007	???	16	4.13	65	375.00	24531.80
1000	10000	625	14125	1009	???	17	4.54	65	364.93	23909.13
1000	10000	1250	14750	6	Yes	6	4.58	1	30.31	35.28
1000	10000	1875	15375	1009	???	17	4.30	65	369.04	24152.90
2500	5000	625	10625	2503	Yes	39	4.31	79	340.14	27607.44
2500	5000	1250	11250	21	Yes	21	4.26	1	28.52	33.17
2500	5000	1875	11875	2503	Yes	40	4.34	78	339.14	26995.48
2500	7500	625	13125	2506	Yes	46	4.61	72	356.50	25895.66
2500	7500	1250	13750	22	Yes	22	4.24	1	31.88	36.52
2500	7500	1875	14375	2506	Yes	35	4.35	78	352.15	27706.35
2500	10000	625	15625	2509	Yes	38	4.59	72	367.34	26657.82
2500	10000	1250	16250	17	Yes	17	4.18	1	32.69	37.25
2500	10000	1875	16875	2508	???	37	4.28	72	368.48	26699.86

TABLE 11. Hard instances with $n = 2500$

As the final experiment, we have used the parameter values #CS1 = 3, #CS2 = 3, $a = 1$, $b = n$, $\theta = 5$ and $\tau = 50$ and tested all available standard problems (subject to memory limitations). The outcome of these experiments are summarized in Tables 12,13, 14, 15, 16, 17. Our algorithm produced optimal solutions for all these problems. Using the large amount of data generated in the experiments, we tried a linear and quadratic fit to estimate the running time of our algorithm as a function of the problem size. The quadratic fit approximates the running time very well. (See Figure 2.) This is consistent with the known experimental results on the average complexity of $O(n^{2.2})$ for the LK-heuristic [12, 16].

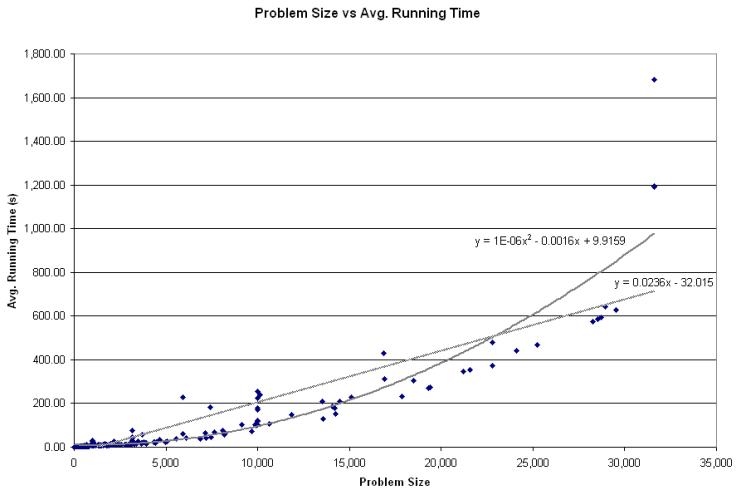

FIGURE 2. Problem size vs Avg. running time

4. Conclusion

In this paper we have developed powerful heuristic algorithms for solving the BTSP. Results of systematic and detailed computational experiments are also reported. No such study is available in the literature so far for the BTSP. Our algorithms use powerful TSP heuristics as a subroutine. We obtained optimal solutions for all the TSPLIB problems, JM-instances, National TSP instances, and VLSI instances. Almost all the solutions produced also have a proof of optimality. We also obtained optimal solutions for several hard instances we have created. All these impressive experimental results were achieved in very reasonable computational time. We believe the algorithms we have developed in this paper and our experimental analysis provide substantial insight into the structure of the BTSP and offer effective solution methods.

Problem	Size	Best obj. value	BBSSP Lower bound	BBSSP Avg. time	Avg LK calls	Avg. LK time	Avg. Total time
bbz25234	25234	15	15	259.39	1.00	154.36	467.37
bch2762	2762	15	15	2.50	1.00	8.69	11.83
bck2217	2217	16	16	1.61	1.00	8.25	10.27
bcl380	380	16	16	0.05	1.00	2.09	2.15
beg3293	3293	24	24	4.01	1.00	10.08	15.00
bgb4355	4355	13	13	6.90	1.00	14.52	23.01
bgd4396	4396	28	28	7.17	1.00	12.49	21.27
bgf4475	4475	25	25	7.35	1.00	12.26	21.28
bnd7168	7168	15	15	18.86	1.00	18.72	41.90
boa28924	28924	18	18	377.03	1.00	193.97	641.48
bva2144	2144	15	15	1.69	1.00	7.86	9.93
dbj2924	2924	16	16	3.16	1.00	9.11	12.99
dca1389	1389	18	18	0.72	1.00	7.51	8.39
dcb2086	2086	21	21	1.42	1.00	8.22	10.00
dcc1911	1911	16	16	1.34	1.00	7.80	9.45
dea2382	2382	25	25	2.35	1.00	8.93	11.75
dga9698	9698	15	15	34.51	1.00	28.36	70.74
dhb3386	3386	17	17	4.23	1.00	9.69	14.87
dja1436	1436	15	15	0.51	1.00	7.03	7.72
djb2036	2036	15	15	1.36	1.00	7.84	9.54
djc1785	1785	24	24	1.06	1.00	8.31	9.63
dka1376	1376	15	15	0.63	1.00	7.38	8.16
dkc3938	3938	16	16	6.28	1.00	9.96	17.53
dkd1973	1973	14	14	1.42	1.00	7.86	9.61
dke3097	3097	19	19	3.52	1.00	9.44	13.75
dkf3954	3954	20	20	5.72	1.00	10.54	17.57
dkg813	813	18	18	0.22	1.00	5.19	5.46
dlb3694	3694	19	19	5.02	1.60	48.48	55.45
fdp3256	3256	20	20	3.88	1.00	11.70	16.46
fea5557	5557	15	15	11.18	1.00	22.56	36.32
fjr3672	3672	21	21	6.09	1.00	10.45	17.65
fjs3649	3649	21	21	5.48	1.00	10.42	17.01
fma21553	21553	16	16	206.98	1.00	108.81	354.83
fnb1615	1615	45	45	0.86	1.00	10.38	11.46
fnc19402	19402	18	18	152.32	1.00	90.24	274.10
fqm5087	5087	16	16	10.44	1.00	12.60	25.19
fra1488	1488	13	13	0.81	1.00	7.23	8.23
frh19289	19289	28	28	151.02	1.00	89.00	271.25
frv4410	4410	15	15	6.33	1.00	11.05	19.00
fyg28534	28534	15	15	330.94	1.00	187.94	587.37
icw1483	1483	21	21	0.74	1.00	8.15	9.07
icx28698	28698	18	18	335.63	1.00	190.00	594.93
ida8197	8197	14	14	26.95	1.00	22.03	55.20
ido21215	21215	16	16	182.31	1.10	121.90	345.75
ird29514	29514	16	16	354.49	1.00	201.55	629.35
irw2802	2802	15	15	2.57	1.00	9.30	12.52
irx28268	28268	15	15	323.91	1.00	183.93	575.07

TABLE 12. VLSI Instances bbz25234-irx28268

Problem	Size	Best obj. value	BBSSP Lower bound	BBSSP Avg. time	Avg LK calls	Avg. LK time	Avg. Total time
lap7454	7454	16	16	20.44	1.00	20.47	45.54
ley2323	2323	27	27	2.24	1.00	8.77	11.47
lim963	963	13	13	0.26	1.00	6.65	6.98
lsb22777	22777	23	23	210.71	1.00	119.88	374.12
lsm2854	2854	19	19	2.66	1.00	9.50	12.85
lsn3119	3119	16	16	3.58	1.00	9.50	13.89
lta3140	3140	17	17	3.64	1.00	9.49	13.95
ltb3729	3729	13	13	5.06	1.00	11.75	17.97
mlt2597	2597	21	21	2.48	1.00	8.94	11.98
pbd984	984	13	13	0.32	1.00	6.77	7.17
pbk411	411	14	14	0.05	1.00	2.48	2.54
pbl395	395	13	13	0.04	1.00	2.26	2.31
pbm436	436	15	15	0.06	1.00	2.63	2.70
pbn423	423	14	14	0.05	1.00	2.40	2.46
pds2566	2566	15	15	2.14	1.00	8.45	11.14
pia3056	3056	15	15	3.35	1.00	9.44	13.57
pjh17845	17845	14	14	128.50	1.00	76.89	231.96
pka379	379	11	11	0.05	1.00	1.69	1.75
pma343	343	15	15	0.04	1.00	1.72	1.77
rbu737	737	13	13	0.18	1.00	4.39	4.62
rbv1583	1583	18	18	0.72	1.00	7.75	8.68
rbw2481	2481	27	27	2.27	1.00	9.18	11.96
rbx711	711	15	15	0.13	1.00	4.18	4.35
rby1599	1599	18	18	0.74	1.00	8.63	9.58
xia16928	16928	15	15	115.89	1.70	153.71	310.45
xit1083	1083	11	11	0.39	1.00	5.98	6.46
xmc10150	10150	17	17	41.64	3.10	165.65	239.83
xpr2308	2308	14	14	1.77	1.00	7.88	10.09
xqc2175	2175	15	15	1.54	1.00	8.25	10.18
xqd4966	4966	16	16	9.89	1.00	12.05	23.99
xqe3891	3891	20	20	5.54	1.00	10.34	17.15
xqf131	131	23	23	0.00	1.00	0.57	0.57
xqg237	237	13	13	0.02	1.00	1.02	1.05
xql662	662	26	26	0.15	1.00	4.40	4.59
xrb14233	14233	14	14	82.20	1.00	53.88	153.11
xrh24104	24104	17	17	258.39	1.00	136.06	443.29
xsc6880	6880	18	18	17.24	1.00	17.31	38.49
xua3937	3937	15	15	5.71	1.00	9.91	16.91
xva2993	2993	12	12	3.29	1.00	8.67	12.71
xvb13584	13584	15	15	67.32	1.00	48.01	130.84

TABLE 13. VLSI Instances lap7454-xvb13584

Problem	Size	Best obj. value	BBSSP Lower bound	BBSSP Avg. time	Avg LK calls	Avg. LK time	Avg. Total time
ar9152	9152	4871	4871	59.16	1.00	35.93	102.02
ca4663	4663	13768	13768	16.27	1.00	17.22	35.18
dj89	89	437	437	0.00	1.00	0.28	0.29
eg7146	7146	2150	2150	28.43	1.00	34.21	65.86
ei8246	8246	124	124	36.16	1.00	23.86	65.69
fi10639	10639	768	768	59.86	1.00	37.33	106.56
gr9882	9882	1399	1399	56.73	1.00	32.92	97.15
ho14473	14473	440	440	118.28	1.00	72.54	208.38
it16862	16862	1499	1499	163.35	1.70	229.81	431.88
ja9847	9847	6413	6413	61.65	1.00	32.95	101.17
kz9976	9976	1602	1602	61.80	1.00	33.84	103.94
lu980	980	44	44	0.33	1.00	5.57	5.97
mo14185	14185	939	939	108.14	1.00	55.67	180.20
mu1979	1979	1153	1153	2.22	1.00	11.58	14.10
nu3496	3496	650	650	6.68	1.00	17.14	24.84
pm8079	8079	331	331	36.15	1.00	33.93	75.56
qa194	194	370	370	0.02	1.00	1.11	1.13
rw1621	1621	150	150	1.23	1.00	10.01	11.46
sw24978	24978	1068	1044	357.33	27.40	5803.98	8270.68
tz6117	6117	486	486	21.28	1.00	15.89	40.26
uy734	734	389	389	0.27	1.00	4.73	5.04
vm22775	22775	388	388	293.00	1.00	141.35	477.76
wi29	29	2250	2250	0.00	1.00	0.02	0.02
ym7663	7663	3113	3113	37.08	1.00	25.83	66.67
zi929	929	887	887	0.37	1.00	7.68	8.12

TABLE 14. National TSP instances

Problem	Size	Best obj. value	BBSSP Lower bound	BBSSP Avg. time	Avg LK calls	Avg. LK time	Avg. Total time
a280	280	20	20	0.02	1.00	0.89	0.92
ali535	535	3889	3889	2.84	1.00	4.37	7.55
att48	48	519	519	0.00	1.00	0.08	0.08
att532	532	229	229	0.27	1.00	3.53	3.85
bayg29	29	111	111	0.00	1.00	0.03	0.03
bays29	29	154	154	0.00	1.00	0.03	0.03
berlin52	52	475	475	0.00	1.00	0.12	0.13
bier127	127	7486	7486	0.01	1.00	0.77	0.78
brazil58	58	2149	2149	0.00	1.00	0.18	0.18
brd14051	14051	1306	1306	113.44	1.00	57.98	187.73
brg180	180	30	30	0.00	1.00	0.99	1.00
burma14	14	418	418	0.00	1.00	0.00	0.01
ch130	130	142	142	0.01	1.00	0.41	0.42
ch150	150	93	93	0.01	1.00	0.37	0.38
d1291	1291	1289	1289	0.70	1.00	11.68	12.49
d15112	15112	1370	1370	133.18	1.00	75.59	227.86
d1655	1655	1476	1476	0.84	1.00	12.42	13.44
d18512	18512	476	476	184.11	1.00	91.34	304.16
d198	198	1380	1380	0.02	1.00	1.40	1.42
d2103	2103	1133	1133	2.02	1.00	12.30	14.65
d493	493	2008	2008	0.11	1.00	5.22	5.34
d657	657	1368	1368	0.21	1.00	6.68	6.92
dantzig42	42	35	35	0.00	1.00	0.05	0.05
dsj1000	1000	295939	295939	1.01	1.00	10.85	11.96
eil101	101	13	13	0.00	1.00	0.25	0.25
eil51	51	13	13	0.00	1.00	0.05	0.05
eil76	76	16	16	0.00	1.00	0.22	0.22
fl1400	1400	530	530	0.86	1.00	9.57	10.58
fl1577	1577	431	431	1.08	1.00	9.66	10.94
fl3795	3795	528	528	6.99	1.00	13.91	22.01
fl417	417	472	472	0.07	1.10	5.58	5.67
fnl4461	4461	132	132	9.84	1.00	12.09	23.59
fri26	26	93	93	0.00	1.00	0.03	0.03
gil262	262	23	23	0.02	1.00	0.93	0.96
gr120	120	220	220	0.00	1.00	0.48	0.49
gr137	137	2132	2132	0.16	1.00	0.51	0.69
gr17	17	282	282	0.00	1.00	0.01	0.01
gr202	202	2230	2230	0.36	1.00	1.57	1.97
gr21	21	355	355	0.00	1.00	0.02	0.02
gr229	229	4027	4027	0.47	1.00	1.31	1.85
gr24	24	108	108	0.00	1.00	0.02	0.02
gr431	431	4027	4027	1.81	1.00	3.44	5.46
gr48	48	227	227	0.00	1.00	0.07	0.07
gr666	666	4264	4264	4.62	1.00	5.17	10.33
gr96	96	2807	2807	0.07	1.00	0.35	0.43
hk48	48	534	534	0.00	1.00	0.06	0.07
kroA100	100	475	475	0.01	1.00	0.19	0.20
kroA150	150	392	392	0.01	1.00	0.32	0.34
kroA200	200	408	408	0.02	1.00	0.65	0.68
kroB100	100	530	530	0.00	1.00	0.24	0.25
kroB150	150	436	436	0.01	1.00	0.39	0.40
kroB200	200	344	344	0.02	1.00	0.52	0.55
kroC100	100	498	498	0.00	1.00	0.18	0.19
kroD100	100	491	491	0.01	1.00	0.18	0.19
kroE100	100	490	490	0.00	1.00	0.18	0.19

TABLE 15. TSPLIB Instances a280-kroE100

Problem	Size	Best obj. value	BBSSP Lower bound	BBSSP Avg. time	Avg LK calls	Avg. LK time	Avg. Total time
lin105	105	487	487	0.01	1.00	0.28	0.28
lin318	318	487	487	0.04	1.00	1.36	1.41
nrw1379	1379	105	105	0.94	1.00	7.34	8.44
p654	654	1223	1223	0.20	1.00	5.17	5.40
pa561	561	16	16	0.05	1.00	3.38	3.44
pcb1173	1173	243	243	0.67	1.00	7.52	8.30
pcb3038	3038	198	198	4.44	1.00	8.39	13.58
pcb442	442	500	500	0.10	1.00	2.83	2.94
pla7397	7397	81438	81438	55.21	2.18	113.82	181.79
pr1002	1002	2129	2129	0.41	1.00	6.57	7.07
pr107	107	7050	7050	0.00	1.00	0.41	0.42
pr124	124	3302	3302	0.01	1.00	0.47	0.48
pr136	136	2976	2976	0.01	1.00	0.53	0.54
pr144	144	2570	2570	0.01	1.00	0.50	0.51
pr152	152	5553	5553	0.01	1.00	0.66	0.67
pr226	226	3250	3250	0.02	1.00	0.91	0.94
pr2392	2392	481	481	2.71	1.00	8.84	12.03
pr264	264	4701	4701	0.03	1.00	1.79	1.83
pr299	299	498	498	0.05	1.00	2.35	2.41
pr439	439	2384	2384	0.10	1.00	2.95	3.07
pr76	76	3946	3946	0.00	1.00	0.20	0.20
rat195	195	21	21	0.01	1.00	0.50	0.51
rat575	575	23	23	0.12	1.00	2.67	2.82
rat783	783	26	26	0.25	1.00	5.16	5.46
rat99	99	20	20	0.00	1.00	0.09	0.10
rd100	100	221	221	0.00	1.00	0.29	0.30
rd400	400	104	104	0.07	1.00	1.86	1.94
rl11849	11849	842	842	81.20	1.00	56.66	149.59
rl1304	1304	1535	1535	1.05	1.00	8.15	9.35
rl1323	1323	2489	2489	1.03	1.00	9.70	10.87
rl1889	1889	896	896	1.91	1.00	7.52	9.72
rl5915	5915	602	602	21.19	4.20	188.47	229.82
rl5934	5934	896	896	20.09	1.00	37.30	60.29
si1032	1032	362	362	0.13	1.00	21.67	21.84
si175	175	177	177	0.00	1.00	0.67	0.68
si535	535	227	227	0.04	1.00	4.30	4.35
st70	70	24	24	0.00	1.00	0.15	0.15
swiss42	42	67	67	0.00	1.00	0.06	0.06
ts225	225	1000	500	0.02	10.40	2.92	2.99
tsp225	225	36	36	0.02	1.00	0.72	0.75
u1060	1060	2378	2378	0.59	1.00	7.72	8.39
u1432	1432	300	300	0.84	1.00	6.20	7.21
u159	159	800	800	0.01	1.00	0.51	0.52
u1817	1817	234	234	1.46	1.00	6.88	8.61
u2152	2152	105	105	2.05	1.00	23.22	25.66
u2319	2319	224	224	2.20	1.00	7.38	10.03
u574	574	345	345	0.16	1.00	3.54	3.73
u724	724	170	170	0.24	1.00	3.72	4.00
ulysses16	16	1504	1504	0.00	1.00	0.01	0.01
ulysses22	22	1504	1504	0.00	1.00	0.02	0.03
usa13509	13509	16754	16754	142.76	1.00	53.03	211.14
vm1084	1084	998	998	0.73	1.00	6.08	6.91
vm1748	1748	1017	1017	1.76	1.00	7.45	9.47

TABLE 16. TSPLIB Instances lin105-vm1748

Problem	Size	Best obj. value	BBSSP Lower bound	BBSSP Avg. time	Avg LK calls	Avg. LK time	Avg. Total time
C10k.0	10000	161062	161062	86.80	1.60	125.16	225.32
C10k.1	10000	94139	94139	80.19	1.00	90.09	178.63
C10k.2	10000	121209	121209	87.36	1.10	76.53	173.08
C1k.0	1000	290552	290552	0.80	1.00	12.56	13.44
C1k.1	1000	335184	335184	0.83	1.00	11.20	12.12
C1k.2	1000	225295	225295	0.71	1.00	9.39	10.18
C1k.3	1000	416768	416768	0.76	1.20	26.80	27.66
C1k.4	1000	318930	318930	0.90	1.00	9.88	10.86
C1k.5	1000	260389	260389	0.89	1.10	29.26	30.25
C1k.6	1000	175740	175740	0.80	1.00	10.06	10.94
C1k.7	1000	301366	301366	0.87	1.00	11.32	12.28
C1k.8	1000	246519	246519	0.84	1.00	9.83	10.75
C1k.9	1000	208091	208091	0.82	1.00	9.32	10.22
C31k.0	31623	84302	84302	888.35	1.67	652.06	1680.74
C3k.0	3162	252245	252245	8.55	1.00	22.83	32.21
C3k.1	3162	167466	167466	8.56	1.50	64.86	74.67
C3k.2	3162	194007	194007	8.32	1.00	18.28	27.43
C3k.3	3162	180852	180852	8.48	1.10	32.54	41.94
C3k.4	3162	180583	180583	8.60	1.00	20.19	29.63
E10k.0	10000	20174	20174	81.09	1.00	29.60	119.03
E10k.1	10000	22883	22883	81.23	1.00	31.47	121.10
E10k.2	10000	20208	20208	80.94	1.00	29.72	119.04
E1k.0	1000	64739	64739	0.74	1.00	5.26	6.09
E1k.1	1000	67476	67476	0.78	1.00	5.64	6.50
E1k.2	1000	88522	88522	0.76	1.00	7.24	8.08
E1k.3	1000	59220	59220	0.78	1.00	4.87	5.73
E1k.4	1000	68259	68259	0.78	1.00	5.68	6.54
E1k.5	1000	61406	61406	0.78	1.00	4.98	5.85
E1k.6	1000	68777	68777	0.74	1.00	5.87	6.70
E1k.7	1000	70389	70389	0.76	1.00	5.93	6.77
E1k.8	1000	57597	57597	0.83	1.00	5.00	5.92
E1k.9	1000	68420	68420	0.75	1.00	5.75	6.58
E31k.0	31623	12419	12419	863.27	1.00	243.78	1192.12
E31k.1	31623	15169	15169	862.53	1.00	248.05	1194.92
E3k.0	3162	39854	39854	8.13	1.00	8.71	17.68
E3k.1	3162	37500	37500	8.15	1.00	7.99	16.97
E3k.2	3162	35145	35145	8.14	1.00	7.59	16.57
E3k.3	3162	44428	44428	8.15	1.00	9.43	18.42
E3k.4	3162	36621	36621	8.14	1.00	7.98	16.96
M10k.0	10000	1189	1189	115.83	1.00	129.45	255.15
M1k.0	1000	9328	9328	0.60	1.00	17.13	17.77
M1k.1	1000	8856	8856	0.64	1.00	16.74	17.42
M1k.2	1000	11282	11282	0.62	1.00	19.46	20.11
M1k.3	1000	11617	11617	0.59	1.00	19.28	19.91
M3k.0	3162	3289	3289	9.31	1.00	37.77	47.56
M3k.1	3162	3034	3034	9.27	1.00	36.82	46.57

TABLE 17. JM Random Instances

As indicated earlier, the algorithms can easily be used to solve the MSTSP. Also the algorithms can be used to test (heuristically) the existence of a Hamiltonian cycle in a graph. It is also possible to solve the asymmetric version of the BTSP using the algorithms with appropriate modifications by replacing the LK heuristic with a good ATSP heuristic. However, additional experimentation is needed to assess the quality of the resulting solutions. For experimental results on ATSP heuristics, we refer to the book chapter [15].

Acknowledgement: We are thankful to Subin Punnen for his assistance in preparing the tables and charts.

References

[1] D. Applegate, R. Bixby, V. Chvtal, W. Cook, and M. Mevenkamp. Concorde TSP Solver. Last Updated Jan. 2005.
http://www.tsp.gatech.edu/concorde.html

[2] E. M. Arkin, Y. Chiang, J. S. B. Mitchell, S. S. Skiena, and T. Yang, On the maximum scatter traveling salesman problem, *SIAM Journal of Computing* 29 (1999) 515-544.

[3] S. Arora, Approximation algorithms for geometric TSP. In *The Travelling Salesman Problem and Its Variants* G. Gutin and A. P. Punnen (eds) Kluwer Academic Publishers, Secaucus, NJ, USA, 2002. Ch. 5, p 207-222.

[4] M. O. Ball and M. J. Magazine, Sequencing of insertions in printed circuit board assembly, *Operations Research* 36 (1988) 192-201.

[5] G. Carpaneto, S. Martello, and P. Toth. An algorithm for the bottleneck traveling salesman problem. *Operations Research*, 32: 380 389, 1984.

[6] Doroshko, N. N.; Sarvanov, V. I. The minimax traveling salesman problem and Hamiltonian cycles in powers of graphs. (Russian) *Vestsi Akad. Navuk BSSR Ser. Fi z.-Mat. Navuk* 1981, no. 6, 119-120, 143

[7] J. Edmonds and F. R. Fulkerson, Bottleneck estrema, *Journal of Combinatorial Theory* 8 (1970) 299-306.

[8] R. S. Garfinkel and K. C. Gilbert. The bottleneck traveling salesman problem: Algorithms and probabilistic analysis. *Journal of the Association of Computing Machinary* 25 (1978) 435-448.

[9] Gabovic, E.; Ciz, A.; Jalas, A. The bottleneck travelling salesman problem. (Russian) *Trudy Vy cisl. Centra Tartu. Gos. Univ.* 22 (1971), 3-24

[10] G. Gutin and A. P. Punnen *The Travelling Salesman Problem and Its Variants* G. Gutin and A. P. Punnen (eds) Kluwer Academic Publishers, Secaucus, NJ, USA, 2002.

[11] P. C. Gilmore and R. E. Gomory. Sequencing a one state-variable machine: A solvable case of the traveling salesman problem. *Operations research*, 12: 655 679, 1964.

[12] K. Helsgaun, "An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic", *European Journal of Operational research* 12 (2000) 106-130.

[13] D.S. Hochbaum and D. B. Shmoys, A unified approach to approximation algorithms for bottleneck problems, *Journal of the Association of Computing machinary* 33 (1986) 533-550.

[14] D. S. Johnson and L. A. McGeoch. Benchmark Code and Instance Generation Codes. Last Updated May 2002.
http://www.research.att.com/d̃sj/chtsp/download.html

[15] D. S. Johnson, G. Gutin, L. A. McGeoch, A. Yeo, W. Zhang, and A. Zverovitch, Experimental analysis of heuristics for the ATSP. In *The Travelling Salesman Problem*

and Its Variants G. Gutin and A. P. Punnen (eds) Kluwer Academic Publishers, Secaucus, NJ, USA, 2002. Ch. 10, p 445-489.

[16] D. S. Johnson and L. A. McGeoch, Experimental Analysis of Heuristics for the STSP. In *The Travelling Salesman Problem and Its Variants* G. Gutin and A. P. Punnen (eds) Kluwer Academic Publishers, Secaucus, NJ, USA, 2002. Ch. 9, p 369-444.

[17] S. Kabadi Polynomially solvable cases of the TSP. In *The Travelling Salesman Problem and Its Variants* G. Gutin and A. P. Punnen (eds) Kluwer Academic Publishers, Secaucus, NJ, USA, 2002. Ch. 11, p 489-584.

[18] S. Kabadi and A. P. Punnen. The Bottleneck TSP. In *The Travelling Salesman Problem and Its Variants* G. Gutin and A. P. Punnen (eds) Kluwer Academic Publishers, Secaucus, NJ, USA, 2002. Ch. 15, p 697-736.

[19] M. Y. Kao and M. Sanghi, An approximation algorithm for a bottleneck traveling salesman problem, Research Report, Department of Computer Science, Northwestern University.

[20] Kljaus, P. S. A special case of the bottleneck traveling salesman problem. (Russian) *Vesci Akad. Navuk BSSR Ser. Fi z.-Mat. Navuk* 1975, no. 1, 61-65, 141.

[21] E. L. Lawler, J. K. Lenstra, A. H. G. Rinooy Kan, and D. B. Shmoys (eds) *Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, Wiley, Chichester, 1985.

[22] S. Lin and B.W. Kernighan. An effective heuristic algorithm for the traveling salesman problem. *Operations Research* 21:972-989, 1973

[23] G. S. Manku. A linear time algorithm for the bottleneck biconnected spanning subgraph problem. *Information Processing Letters*, 59: 1 7, 1996.

[24] Dave Mount, Articulation points and Biconnected Components, Oct. 6 1998. http://www.cs.umd.edu/šamir/451/bc.ps

[25] D. Naddef. Polyhedral Theory and Branch-and-Cut Algorithms for the Symmetric TSP. In *The Travelling Salesman Problem and Its Variants*, Secaucus, NJ, USA: Kluwer Academic Publishers, 2002. Ch. 2, p 29-116.

[26] R. G. Parker and R. L. Rardin. Guaranteed performance heuristics for the bottleneck traveling salesperson problem. *Operations Research letters*, 12: 269 272, 1982.

[27] J. M. Philips, A. P. Punnen, and S. N. Kabadi, A linear time algorithm for the bottleneck traveling salesman problem on a Halin graph, *Informations Processing Letters* 67 (1998) 105-110.

[28] A. P. Punnen. Computational Complexity. In The Travelling Salesman Problem and Its Variants. Secaucus, NJ, USA: Kluwer Academic Publishers, 2002. Appendix B, p 754.

[29] A. P. Punnen and K. P. K. Nair. A fast and simple algorithm for the bottleneck biconnected spanning subgraph problem. *Information Processing Letters*, 50: 283 286, 1994.

[30] R. Ramakrishnan, Prabha Sharma, and A.P. Punnen, New heuristics for the bottleneck TSP. Opsearch 46 (2009) 275 - 288.

[31] S. S. Reddi and R. Ramamoorthy. On the flow-shop sequencing problem with no wait in process, *Operational Research Quarterly* 23 (1972) 1-9.

[32] C. Rego and F. Glover, Local search and metaheuristics, In *The Travelling Salesman Problem and Its Variants*, Secaucus, NJ, USA: Kluwer Academic Publishers, 2002. Ch. 8, p 309-367.

[33] G. Reinelt. TSPLIB. http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/

[34] Sarvanov V I, A minimax traveling salesman problem on a plane: complexity of an approximate solution. (Russian) *Dokl. Akad. Nauk Belarusi* 39 (1995), no. 6, 1619

[35] Sergeev, S. I.; Chernyshenko, A. V. Algorithms for solving a minimax traveling salesman problem. II. A dual approach. *Automat. Remote Control* 56 (1995), no. 8, part 2, 1155–1168 (1996)

[36] Sergeev, S. I. Algorithms for solving a minimax traveling salesman problem. I. An approach based on dynamic programming. *Automation and Remote Control* 56 (1995), no. 7, part 2, 1027–1032
[37] The world TSP page, http://www.tsp.gatech.edu/world/countries.html
[38] E. A. Timofeev. Minmax di-associated subgraphs and the bottleneck traveling salesman problem. *Cybernetics*, 4: 75 79, 1979.
[39] UNB Advanced Computational Research Laboratory. URL: http://acrl.cs.unb.ca
[40] Vairaktarakis, George L. On Gilmore-Gomory's open question for the bottleneck TSP. *Operations Research letters* 31 (2003), no. 6, 483–491
[41] J. A. A. van der Veen, An O(n) algorithm to solve the Bottleneck Traveling Salesman Problem restricted to ordered product matrices, *Discrete Applied Mathematics* 47 (1993) 57-75.

Faculty of Computer Science, University of New Brunswick, Fredericton, New Brunswick, Canada
E-mail address: jal21@sfu.ca

Department of Mathematics, Simon Fraser University Surrey, Central City, 250-13450 102nd AV, Surrey, British Columbia,V3T 0A3, Canada
E-mail address: apunnen@sfu.ca

Faculty of Computer Science, University of New Brunswick, Fredericton, New Brunswick, Canada
E-mail address: aubanel@unb.ca