Computers & Operations Research 43 (2014) 20–35

Contents lists available at ScienceDirect

Computers & Operations Research

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

The asymmetric bottleneck traveling salesman problem: Algorithms, complexity and empirical analysis ☆

John LaRusic, Abraham P. Punnen *

Department of Mathematics, Simon Fraser University Surrey, 250-13450 102nd Ave., Surrey, BC, Canada V3T 0A3

ARTICLE INFO

Available online 19 August 2013

Keywords:
Traveling salesman problem
Bottleneck problems
Experimental analysis
Heuristics
Complexity
Approximation algorithms

ABSTRACT

We consider the asymmetric bottleneck traveling salesman problem on a complete directed graph on n nodes. Various lower bound algorithms are proposed and the relative strengths of each of these bounds are examined using theoretical and experimental analysis. A polynomial time $\lceil n/2 \rceil -$approximation algorithm is presented when the edge-weights satisfy the triangle inequality. We also present a very efficient heuristic algorithm that produced provably optimal solutions for 270 out of 331 benchmark test instances. Our algorithms are applicable to the maxmin version of the problem, known as the maximum scatter TSP. Extensive experimental results on these instances are also given.

1. Introduction

Let $G=(V,E)$ be a directed or undirected graph with $n=|V|$ and $m=|E|$. For each edge $(i,j) \in E$, a nonnegative cost c_{ij} is prescribed. Without loss of generality, we assume that G is complete. The $n \times n$ matrix $C=(c_{ij})_{n \times n}$ is called the cost matrix. Let $\Pi(G)$ be the collection of all (directed) Hamiltonian cycles in G. Then the bottleneck traveling salesman problem (BTSP) [20] is to find a Hamiltonian cycle (tour) in G whose largest edge cost is as small as possible, i.e.

$$\text{Minimize} \quad \max\{c_{ij} : (i,j) \in H\}$$
$$\text{subject to} \quad H \in \Pi(G). \tag{1}$$

Akin to the traveling salesman problem (TSP), BTSP instances are classified as either symmetric (i.e. $c_{ij}=c_{ji}$ for all $i,j \in V$) or asymmetric (i.e. $c_{ij} \neq c_{ji}$ for some $i,j \in V$).

BTSP is a special case of the minmax combinatorial optimization problem [37]. For a complete discussion on the complexity of the BTSP we refer to the book chapter by Kabadi and Punnen [26]. In particular, the BTSP is NP-hard, and, unless P=NP, no polynomial time $\epsilon-$approximation algorithm exists for the problem for any $\epsilon > 1$ [14,33,43]. Much like the TSP, polynomial time approximation algorithms with guaranteed performance ratios exist for BTSP on specially structured problem data [6,14,22,26,33]. Moreover, several special cases of the problem can be solved to optimality in polynomial time [26].

Garfinkel and Gilbert discussed a branch and bound based exact algorithm to solve the BTSP and reported computational results with a construction heuristic on randomly generated problems of sizes up to 100 nodes [18]. Timofeev [47] reported experimental results on problems of similar size but with a heuristic algorithm. Sergeev proposed a dynamic programming approach [45] while Carpento et al. reported experimental results with a branch and bound algorithm on problems of size up to 200 nodes [11]. Ramakrishnan et al. presented experimental results with a threshold heuristic on 72 symmetric TSPLIB problems of size up to 783 cities [40] and Ahmad [1] reported experimental results using algorithms based on lexicographic search for symmetric TSPLIB instances with less than 300 cities. In a small computational study with less than 100 cities, Ahmad [2] reported experimental results on asymmetric BTSP instance using a lexicographic search based algorithm. Very recently, LaRusic et al. [29] reported extensive experimental results on the symmetric version of BTSP on almost all available test problems (TSPLIB, Johnson–McGeoch random instances, VLSI and National TSP instances up to 31,623 nodes) and obtained optimal solutions for most of these instances.

In this paper, we focus on the asymmetric version of the BTSP which is not thoroughly investigated in the literature. The best known performance ratio of a polynomial time approximation algorithm for the symmetric TSP with cost matrix satisfying triangle inequality is $\frac{3}{2}$ [12] whereas for the asymmetric TSP it is $O(\log n)$ [17]. Reducing this gap is a well known open problem. In the case of BTSP, it is well known that the symmetric version can be approximated with a performance ratio of 2 whenever the edge weights satisfy the triangle inequality [14,26,33] and this is the best possible bound for a polynomial time algorithm (unless P=NP) for this class of cost matrices. For the asymmetric BTSP, no polynomial time approximation algorithm with bounded

☆This work was supported by NSERC discovery grants awarded to Abraham P. Punnen.

* Corresponding author. Tel.: +1 7787827611.
E-mail addresses: jal21@sfu.ca (J. LaRusic), apunnen@sfu.ca (A.P. Punnen).

0305-0548/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cor.2013.08.005

performance ratio is known even with triangle inequality assumption on the cost matrix. We give a polynomial time approximation algorithm for asymmetric BTSP with performance ratio $\lceil n/2 \rceil$ whenever the edge costs satisfy the triangle inequality, and generalize this result to the case where edge costs satisfying the $\tau-$triangle inequality.

Further, extending the algorithms for the symmetric version reported in [29], we develop a binary search based heuristic for the asymmetric BTSP and report results of extensive computational experiments on all available benchmark test instances for the asymmetric TSP. To the best of our knowledge, no such extensive computational study on asymmetric BTSP is available in the literature. Our algorithm produced optimal solutions for 270 out of 331 problems considered and this is achieved within a very reasonable computational time. We establish optimality certificate in the majority of instances by developing various lower bounding schemes that produce very tight bounds. Extensive theoretical and experimental comparisons of these lower bounds are also given. The optimality of the remaining problems is established by an exact optimization scheme which is obtained by modifying our heuristic algorithm.

The maxmin version of the BTSP is called the *maximum scatter traveling salesman problem* (MSTSP) [5] which is defined as follows:

$$\begin{aligned} &\text{Maximize} \quad \min\{c_{ij} : (i,j) \in H\} \\ &\text{subject to} \quad H \in \Pi(G). \end{aligned} \tag{2}$$

Arkin et al. [5] showed that the symmetric version of MSTSP is NP-Complete, and no constant-factor approximation algorithm exists for the problem unless P=NP. They also provided a 2-approximation algorithm for the MSTSP with a symmetric cost matrix satisfying the triangle inequality and discussed applications of the model in sequencing rivet operations when fastening sheets of metal together in the aircraft industry among others. Kabadi and Punnen [26] obtained a $2\tau-$approximation algorithm for the MSTSP whenever the cost matrix satisfies the $\tau-$triangle inequality and this is the best possible bound for such cost matrices.

The MSTSP can be formulated as a BTSP using the transformation $d_{ij} = M - c_{ij}$ where $D = (d_{ij})_{n\times n}$ is the cost matrix for the equivalent BTSP and M is a sufficiently large number. While this transformation preserves optimality, it does not preserve $\epsilon-$optimality. However, we show that the heuristic developed for the BTSP works reasonably well in practice for the MSTSP under this transformation.

The paper is organized as follows. Section 2 discusses approximation algorithms for the BTSP. In Section 3 we consider lower bounds for the asymmetric BTSP and Section 4 discusses our primary heuristic algorithm, which can easily be modified into an exact BTSP solver. Extensive computational results are reported and discussed in Section 5. Section 6 presents computational results on MSTSP, and concluding remarks are given in Section 7.

For any directed graph G $\delta^+(v)$ and $\delta^-(v)$, respectively, denote the in-degree and the out-degree of vertex v. Since we assume that G is a complete digraph, an instance of BTSP is completely defined by the cost matrix C. For that reason, we use the terminology BTSP on G and BTSP on C interchangeably. Also, for simplicity, a tour in G with cost matrix C is sometimes referred to as a tour in C. A lower bound for a problem means a lower bound for the optimal objective function value of the problem. Finally, for any spanning subgraph S of G, we denote $C_{\max}(S) = \max\{c_{ij} : (i,j) \in S\}$.

2. Approximation algorithms

Approximation algorithms for TSP is a thoroughly investigated research area and the behavior of its symmetric and asymmetric versions are quiet different in terms of approximability. When the edge costs satisfy the triangle inequality, the symmetric version has a $\frac{3}{2}-$approximation algorithm [12] while the best known performance ratio for the asymmetric version is $O(\log n)$ [17,27,28]. The behavior of the BTSP in terms of approximability appears even more intriguing. The symmetric version can be approximated within a factor of 2 whenever the cost matrix satisfies the triangle inequality and this is the best possible performance bound (unless P=NP). For the asymmetric version no $\epsilon-$approximation algorithm is reported in the literature for any $\epsilon > 1$ even if the edge costs satisfy the triangle inequality. It is easy to see that no polynomial time approximation algorithm with a data independent performance ratio exists for BTSP (unless P=NP) on an arbitrary cost matrix [33]. Thus we restrict our attention in this section to asymmetric instances where the edge weights satisfy the $\tau-$triangle inequality. Note that a cost matrix $C = (c_{ij})_{n\times n}$ satisfies *$\tau-$triangle inequality* if $c_{ij} \le \tau(c_{ik} + c_{kj})$ for all $i,j,k \in V$.

The tth *power of a graph* (not necessarily complete) G is the graph $G^t = (V, E^t)$, where $(u,v) \in E^t$ whenever a path from u to v exists in G with at most t edges.

Theorem 1. *Let C be the cost matrix associated with a complete digraph G satisfying $\tau-$triangle inequality for some $\tau > \frac{1}{2}$ and let H^0 be an optimal solution to the BTSP on G. Let S be a spanning subgraph of G such that $C_{\max}(S) \le C_{\max}(H^0)$. If the graph S^t, $1 \le t < n$ and integer t, contains a Hamiltonian cycle H, then*

$$\frac{C_{\max}(H)}{C_{\max}(H^0)} \le \begin{cases} t & \text{if } \tau = 1 \\ \dfrac{\tau}{\tau-1}(2\tau^{t-1} - \tau^{t-2} - 1) & \text{if } \tau > 1 \\ \dfrac{\tau}{\tau-1}(\tau^{t-1} + \tau - 2) & \text{if } \tau < 1 \end{cases}$$

Theorem 1 above was originally proved by Kabadi and Punnen [26] for the symmetric BTSP case. However, the proof is almost identical for the asymmetric version and hence we skip the detailed proof.

Our approximation algorithm was inspired by the 2-approximation algorithm for BTSP on a complete undirected graphs with edge-costs satisfying the triangle inequality discussed in [33,26,14,22]. A formal description of our approximation algorithm for BTSP on a complete directed graph G is given below.

Algorithm Approx-BTSP:

- Step 1: Compute a bottleneck strongly connected spanning subgraph S of G.
- Step 2: Find S^t for $t = \lceil n/2 \rceil$.
- Step 3: Output any hamiltonian cycle in S^t.

To establish the complexity and performance ratio of algorithm Approx-BTSP we use the following well known theorem of Ghouilà-Houri [19].

Theorem 2 (*Ghouilà-Houri [19]*). *If G is a directed graph on n vertices and $\min\{\delta^+(v), \delta^-(v)\} \ge n/2$ for every vertex $v \in G$, then G is Hamiltonian.*

Theorem 3. *Algorithm Approx-BTSP runs in polynomial time and guarantees an $\epsilon-$optimal solution for the asymmetric BTSP whenever the edge-costs satisfy the $\tau-$triangle inequality, where*

$$\epsilon = \begin{cases} \left\lceil \dfrac{n}{2} \right\rceil & \text{if } \tau = 1, \\ \dfrac{\tau}{\tau-1}(2\tau^{\lceil n/2 \rceil - 1} - \tau^{\lceil n/2 \rceil - 2} - 1) & \text{if } \tau > 1, \\ \dfrac{\tau}{\tau-1}(\tau^{\lceil n/2 \rceil - 1} + \tau - 2) & \text{if } \tau < 1. \end{cases}$$

Proof. Let H^0 be an optimal solution to the BTSP on G. Since S is a strongly connected spanning subgraph of G, we have $C_{\max}(S) \le C_{\max}(H^0)$. Thus by Theorem 1, the performance ratio holds. We now show that the algorithm is polynomially bounded. Step 1 can be computed in $O(n^2)$ time [35]. $S^{\lceil n/2 \rceil}$ [32] can be obtained in $O(n^3)$ time using an all-pair shortest path algorithm with unit edge costs. Schaar and Wojda [44] guarantees that $S^{\lceil n/2 \rceil}$ is Hamiltonian and satisfies the condition of Ghouilà-Houri's theorem. Bondy and Thomassen [9] gave an $O(n^4)$ algorithm for finding Hamiltonian cycles in graphs satisfying the conditions of Theorem 2 [9]. Thus a Hamiltonian cycle in $S^{\lceil n/2 \rceil}$ can be obtained in polynomial time. □

3. Lower bounds for the asymmetric BTSP

It is well known that an asymmetric TSP instance on n nodes can be formulated as a symmetric TSP on $2n$ nodes [25]. Using a somewhat similar transformation, Ramakrishnan et al. [40] showed that an asymmetric BTSP instance on n nodes can be formulated as symmetric BTSP instance on $2n$ nodes with an additional constraint. We consider a variation of the transformation of Ramakrishnan et al. [40] where the additional constraint is treated implicitly. This allows us to use known lower bounds for the symmetric BTSP on asymmetric instances, as well as later in the paper allows us to use symmetric TSP heuristics and solvers in the construction of an asymmetric BTSP heuristic.

Consider an asymmetric BTSP instance on a directed graph $G=(V,E)$ with n vertices and cost matrix C. Construct the complete undirected graph $\overline{G}=(\overline{V},\overline{E})$ where $\overline{V}=\{1,2,\ldots,n,n+1,n+2,\ldots,2n\}$. For each edge (i,j) of G create an edge $(i,j+n)$ in $\overline{G}$ with cost c_{ij}. Also introduce edges $(i,i+n)$ for each $i \in V$ with cost $-\infty$. The remaining edges are of weight ∞. Thus the cost matrix $\overline{C}=(\overline{c}_{ij})_{2n\times 2n}$ of $\overline{G}$ is given by

$$\overline{c}_{ij}=\begin{cases}-\infty & \text{if } i=j+n \text{ or } j=i+n\\ c_{i,j-n} & \text{if } i\le n \text{ and } j>n\\ c_{i-n,j} & \text{if } i>n \text{ and} j\le n\\ \infty & \text{otherwise}.\end{cases} \tag{3}$$

The edges of cost $-\infty$ are called *fixed edges* in the sense that they must be included in any symmetric BTSP solution on $\overline{G}$ to make the transformation valid. Ramakrishnan et al. used an additional constraint to enforce these fixed edges into a solution [40]. We discuss how to force these edges into the solution later. Any tour in the directed graph G with asymmetric cost matrix C corresponds to a unique tour in the directed graph G with the symmetric cost matrix $\overline{C}$ where all edges of cost $-\infty$ are included, and vice versa. For $i,j \in \{1,\ldots,n\}$, $i \neq j$, let the edge (i,j) in G be represented by the edge $(i,j+n)$ in $\overline{G}$. The fixed edges $(i,i+n)$ in $\overline{G}$ represent moving between the 'out-edges' and 'in-edges' of node i in G. All other edges in $\overline{G}$ are of infinite cost so as to effectively exclude them from consideration as they do not correspond to valid tour in G.

Consider a tour $\pi=(\pi_1,\pi_2,\pi_3,\ldots,\pi_n)$ of the asymmetric instance. This corresponds to the tour $\overline{\pi}=(\pi_1,\pi_{2+n},\pi_2,\pi_{3+n},\pi_3,\ldots,\pi_n,\pi_{1+n})$ in the symmetric instance (and vice versa). Therefore, we have a one-to-one mapping between tours of the asymmetric instance and tours of the symmetric instance containing all the fixed edges and excluding any edges of weight ∞. Moreover, the BTSP objective values of π and $\overline{\pi}$ are identical. This transformation is effectively used in our heuristics for the BTSP discussed in Section 4.

Let us now discuss some polynomial schemes to compute good quality lower bounds for the optimal objective function of the BTSP. Some of the lower bounds we discuss here are well known, but we present them for the purpose of comparison and completion.

3.1. 2-Max bound (2MB)

This simple bound, described in [11,40], finds the smallest in-edge and out-edge costs incident on every node and selects the largest of all these values over all nodes. It is clearly a lower bound for the BTSP and is computed in $O(m)$ time.

3.2. Bottleneck assignment problem (BAP) bound

The optimal objective function value of the bottleneck assignment problem with cost matrix C is a lower bound for the BTSP [11,18,39] on C. The BAP finds a permutation ϕ of $\{1,\ldots,n\}$ such that $\max_{1\le i\le n}\{c_{i\phi(i)}\}$ is minimized, i.e.

$$\begin{aligned}&\text{minimize} && \max_{i=1,\ldots,n}\{c_{i\phi(i)}\}\\ &\text{subject to} && \phi \in \Phi(n)\end{aligned}$$

where $\Phi(n)$ is the set of all permutations of $\{1,2,\ldots,n\}$.

The BAP can be solved $O(m\sqrt{n}\log k)$ time [3] using the binary search version of the threshold algorithm, where k is the number of distinct costs in C. This is the implementation we used in our computational experiments. The best known algorithm for solving the BAP is a combination of three algorithms as described in [10].

3.3. Bottleneck biconnected spanning subgraph problem (BBSSP)

The first algorithm to solve this problem was published by Timofeev [47] and later an almost identical algorithm was proposed independently by Parker and Rardin [33] and Sarvanov [43] to solve this problem and its optimal objective function value gives a good lower bound for the symmetric BTSP (see [29]). This simple algorithm was used in our computational experiments which runs in $O(m\log n)$ time. The $O(m+n\log n)$ algorithm by Punnen and Nair [38] and an $O(m)$ algorithm by Manku [31] however have better worst case complexity.

The BBSSP bound can be applied on an asymmetric BTSP instance after constructing the equivalent symmetric instance $\overline{G}$ with cost matrix $\overline{C}$ as formulated by Eq. (3). The BBSSP on $\overline{C}$ gives a lower bound on the optimal objective function value of the BTSP on C. We denote this lower bound by BBSSP($\overline{C}$).

One can also compute another lower bound by solving the BBSSP on the symmetric relaxation $\hat{C}$ of the asymmetric cost matrix C where $\hat{C}=(\hat{c}_{ij})_{n\times n}$ is defined as

$$\hat{c}_{ij}=\min\{c_{ij},c_{ji}\}. \tag{4}$$

Let BBSSP($\hat{C}$) be the optimal objective function value of this bottleneck problem. It is easy to verify that the BBSSP($\hat{C}$) is a lower bound on the optimal objective function value of BTSP on C.

3.4. Bottleneck strongly connected spanning subgraph problem (BSCSSP) bound

Since a directed Hamiltonian tour in G is strongly connected, the optimal objective function value of BSCSSP on G with cost matrix C gives a lower bound for the BTSP on G with cost matrix C. Punnen [35] proposed an $O(\min\{m+n\log n, m\log^* n\})$ algorithm to solve this problem, where $\log^* n$ is the iterative logarithm of n, as well as a simpler $O(m\log k)$ implementation. For our experiments, we use the simpler $O(m\log k)$ implementation.

3.5. Bidirectional bottleneck path (BBP) bound

Carpaneto et al. [11] considered a lower bound for the asymmetric BTSP using two bottleneck path computations as follows: for any node $i \in V$, find the tree of bottleneck paths T_1 from i to every other node in V, as well as the tree of bottleneck paths T_2 to i from every other nodes in V. The maximum edge-weight in T_1 and T_2, i.e. $\max\{c_{ij} : (i,j) \in T_1 \cup T_2\}$, is a lower bound for the asymmetric BTSP and this can be identified efficiently [36]. It is easy to see that this bound is identical to the BSCSSP bound. We summarize this observation in the following lemma for future reference.

Lemma 4. *The BBP bound and the BSCSSP bounds are identical.*

3.6. Strengthening the lower bounds

Many of the lower bounds discussed so far can be strengthened by repeated applications of the following scheme on a sequence of related graphs. For $i \in V$ and $(i,j) \in E,\ (k,i) \in E$ consider the graph G^i_{jk} obtained by deleting all arcs incident on node i except (i,j) and (k,i). Let β^i_{jk} be a lower bound for BTSP on G^i_{jk}. Define
$\beta^i = \min\{\beta^i_{jk} : j,k \in V\backslash\{i\}, j \neq k\}$.

Theorem 5. *$\beta = \max_{i \in V}\beta^i$ is a lower bound for the BTSP.*

Proof. We first show that for any $i \in V$, β^i is a lower bound for the BTSP on G. Let F^i_{jk} be the family of all hamiltonian cycles in G^i_{jk}. These are precisely all the tours in G using edges (i,j) and (k,i). Then

$$\Pi(G) = \bigcup_{j \in V, j \neq i} \bigcup_{k \in V, k \neq i,j} F^i_{jk} \tag{5}$$

By definition

$$\beta^i_{jk} \le \max\{c_{pq} : (p,q) \in H\} \quad \text{for all } H \in F^i_{jk} \tag{6}$$

Thus, in view of (5) we have

$$\beta^i \le \max\{c_{pq} : (p,q) \in H\} \quad \text{for all } H \in \Pi(G) \tag{7}$$

Since (7) is true for all $i \in V$ we conclude

$$\beta = \max_{i \in V}\beta^i \le \max\{c_{pq} : (p,q) \in H\} \quad \text{for all } H \in \Pi(G) \qquad \square$$

Using Theorem 5, one may strengthen any of the lower bounds discussed so far. For example, consider solving the BAP lower bound on G^i_{jk}, where arcs not in G^i_{jk} are given a sufficiently large cost so as to exclude them from consideration. Computing β^i requires solving the BAP $O(n^3)$ times (once for each $i,j,k \in V$), and results in an overall complexity of $O(n^{5.5})$ time. Similar techniques strengthen the BSCSSP or the BBSSP bounds in complexity $O(n^5)$. However, a careful implementation exploiting the special properties of the lower bounds achieves significant computational advantage. We illustrate this by showing that if β^i_{jk} is used as the BSCSSP lower bound (equivalently the BBP bound) on G^i_{jk}, then β can be identified in $O(n^{3.792})$ time. We call the resulting strengthened lower bound the enhanced BBP bound or simply the EBBP bound.

The algorithm for computing the EBBP bound works as follows. For each node i, construct the graph $G^i = G\backslash\{i\}$ and solve the all-pair bottleneck path problem on G^i. Let P^i be the resulting matrix of all-pair bottleneck path distances in G^i. Using P^i the bottleneck values of paths from node i to all other nodes in G^i_{jk} is easily identified by scanning row j of P^i and comparing with c_{ij}. Likewise, the values of bottleneck paths from all nodes in G^i_{jk} to node i are identified by scanning column k of P^i and comparing with c_{ki}. Thus β^i_{jk} and hence β^i can be identified in $O(n^{2.792})$ time [48]. The value β is the largest value of β^i obtained. A formal description of the algorithm is given below.

Algorithm 1. *EBBP*(G, C).

Input: A graph $G = (V, E)$ with cost matrix C.
Output: A lower bound on the BTSP objective value.
for $i \in V$ **do**
  $G^i \leftarrow (V\backslash\{i\}, E\backslash\{(u,v) \in E : u = i \text{ or } v = i\})$; /* remove i from G */
  $P^i \leftarrow$ all − pairs − bottleneck − paths(G^i);
  /* P^i is a matrix of bottleneck distances, i.e. row j of P^i gives bottleneck path distances from $j \in V\backslash\{i\}$ to all other nodes in G^i. */
  for $(i,j) \in E$ **do**
    $\alpha^i_j \leftarrow \max\{P^i_{jl} : l \in V\backslash\{i,j\}\}$; /* max bottleneck edge from j */
    for $(k,i) \in E$ **do**
      $\gamma^i_k \leftarrow \max\{P^i_{lk} : l \in V\backslash\{i,k\}\}$; /* max bottleneck edge to k */
      $\beta^i_{jk} \leftarrow \max\{\alpha^i_j, \gamma^i_k, c_{ij}, c_{ki}\}$;
    end
  end
  $\beta^i \leftarrow \min\{\beta^i_{jk} : j,k \in V\backslash\{i\}\}$;
end
$\beta \leftarrow \max\{\beta^i : i \in V\}$;
return β;

Theorem 6. *Algorithm* EBBP(G, C) *correctly identifies the lower bound β in $O(n^{3.792})$ time when β^i_{jk} is the BBP lower bound in G^i_{jk}.*

Proof. Let T_1 be the tree of bottleneck paths from node i to all other nodes in G^i_{jk}. Similarly, let T_2 be the tree of bottleneck paths from all nodes in G^i_{jk} to node i. Clearly

$$\beta^i_{jk} = \max\{c_{pq} : (p,q) \in T_1 \cup T_2\}. \tag{8}$$

Note that P^i is the all-pair bottleneck path matrix on $G^i = G\backslash\{i\}$. Then the jth row of P^i gives bottleneck distances from node j to all other nodes in G^i. Now

$$\begin{aligned}\alpha^i_j &= \max\{P^i_{jl} : l \in V\backslash\{i,j\}\}\\ &= \max\{c_{pq} : (p,q) \in T_1 - (i,j)\}\end{aligned}$$

Similarly, the kth column of P^i gives the bottleneck distances from each node l of G^i to node k. Thus

$$\gamma_k^i = \max\{P_{lk}^i : l \in V \backslash \{i,k\}\}$$
$$= \max\{c_{pq} : (p,q) \in T_2 - (i,k)\}$$

Hence, from Eq. (8) we have

$$\beta_{jk}^i = \max\{\alpha_j^i, \gamma_k^i, c_{ij}, c_{ki}\}$$

Thus the algorithm correctly computes β_{jk}^i, β^i and hence β.

To analyze the complexity, note that P^i can be identified in $O(n^{2.792})$ time [48] for each $i \in V$. All other computations for fixed i takes $O(n^2)$ time. Since these computations are repeated for each $i \in V$, the overall complexity of the algorithm is $O(n^{3.792})$. □

In our computational testing of this algorithm we used a variation of the Floyd–Warshal algorithm for all-pairs shortest paths (see [3]) which has a worst case complexity of $O(n^3)$ to compute the matrix P^i. Using this algorithm in algorithm EBBPB(G, C) results in a moderately higher complexity of $O(n^4)$, but with the advantage of easy implementation.

3.7. Analysis of the lower bounds

Let us now examine the relative strengths of the lower bounds discussed. Two lower bounds A and B are said to be *non-dominated* if there exists instances where $A > B$ and there exists instances where $B > A$. The lower bound A *dominates* the lower bound B if $A \geq B$ for all instances of the asymmetric BTSP.

Theorem 7. *The BSCSSP lower bound (equivalently the BBP lower bound) dominates the* BBSSP($\overline{C}$) *bound.*

Proof. Let δ be the optimal objective function value of the BSCSSP bound on a graph $G = (V, E)$ with cost matrix C. Thus the directed graph $\vec{G}^{\delta} = (V, E^{\delta})$ where $E^{\delta} = \{(i,j) \in E : c_{ij} \leq \delta\}$ is an optimal solution to the BSCSSP bound. Consider the undirected graph constructed from $\vec{G}^{\delta}$ using the $2n$-node symmetric transformation of Eq. (3). Discard all arcs of cost ∞ from this graph and call the resulting graph $\overline{G}$. Note that $\overline{G}$ is bipartite with the partite vertex sets $V_1 = \{1, 2, \ldots, n\}$ and $V_2 = \{n+1, n+2, \ldots, 2n\}$ and has no arcs of cost more than δ. It suffices to show that $\overline{G}$ is biconnected.

$\overline{G}$ must contain a cut-vertex r if it is not biconnected. Suppose $r \in V_1$. Let G_1 and G_2 be two connected components of $\overline{G} \backslash \{r\}$. Then there exist vertices $u+n \in G_1$ and $v+n \in G_2$ such that edges $(r, u+n)$ and $(r, v+n)$ are in $\overline{G}$. By construction of $\overline{G}$, $\vec{G}^{\delta}$ must contain edges (r,u) and (r,v). Since $\vec{G}^{\delta}$ is strongly connected, it must contain a path from u to r and a path from v to r. Thus there must exist a path in $\overline{G}$ from u to $r+n$ that does not contain r and a path from v to $r+n$ that does not contain r. Since r is a cut-vertex, both u and v must be in the same component. Since $u+n$ and $v+n$ are in different components, u, v are in the same component, and $(u, u+n)$ and $(v, v+n)$ are the edges of $\overline{G}$, this shows that a cut vertex cannot belong to the set V_1. The same logic shows that a cut vertex cannot belong to the set V_2, thus $\overline{G}$ must be biconnected. □

Theorem 8. *The EBBP lower bound dominates the 2MB, BSCSSP, BBP, BBSSP($\overline{C}$), and BBSSP($\hat{C}$) lower bounds.*

Proof. Let S be an optimal solution to the BSCSSP bound. Since each node of S will have at least one in-edge and one out-edge, the BSCSSP bound is at least as good as the 2MB. Since the EBBP bound is obtained by additional restriction on the BBP bound, it is as good as the BBP bound. Since the BSCSSP bound and the BBP bound are equivalent, the EBBP bound is at least as good as the BSCSSP bound. Thus, in view of Theorem 7, the EBBP bound is at least as good as the BBSSP($\overline{C}$). It remains to show that EBBP lower bound dominates the BBSSP($\hat{C}$) lower bound.

We prove this using the method of contradiction. If possible let there exists an instance of BTSP, say on a graph $G = (V, E)$ with cost matrix C, such that the EBBP bound, say β, is strictly less than the BBSSP($\hat{C}$) bound. Consider that the spanning subgraph G^{β} consists of all arcs in G with cost no more than β. By Lemma 4 G^{β} is strongly connected. Since β is less than the BBSSP($\hat{C}$) bound, G^{δ} must have a cut-vertex, say i. Thus $G^{\delta} \backslash \{i\}$ will have at least two components. For $(i,j) \in E$ and $(k,i) \in E$ let G_{jk}^i be the subgraph of G obtained by deleting all edges incident on i except (i,j) and (k,i) and let β_{jk}^i be the BBP (equivalently BSCSSP) lower bound on G^ijk. Let $\beta^i = \min\{\beta_{jk}^i : j,k \in V \backslash \{i\}, j \neq k\} = \beta_{uv}^i$, say. Note that by definition $\beta = \max\{\beta^i : i \in V\}$ and hence $\beta \geq \beta_{uv}^i$. Thus G_{uv}^i is a strongly connected spanning subgraph of G^{β}. Let w be a node which is not in the same connected component of $G^{\beta} \backslash \{i\}$ as the node u. Then there is no path from u to w in G_{uv}^i. This contradicts the strong connectivity of G_{uv}^i and hence the result. □

Theorem 9. *The BAP, BBSSP($\overline{C}$), BSCSSP, BBP, and EBBP bounds dominate the 2MB.*

Proof. Let $\sigma = (\sigma(1), \sigma(2), \ldots, \sigma(n))$ be an optimal solution to BAP with objective function value z. Note that σ generates a cycle cover Q of G such that $\max\{c_{ij} : (i,j) \in Q\} = z$. Since Q contains all nodes of G and each node has an incoming arc and an outgoing arc, 2MB cannot be larger than z.

For a strongly connected graph, each vertex has at least one incoming arc and at least one outgoing arc, the BSCSSP bound and BBP bound are no worse than the 2MB. By Theorem 8 EBBP bound dominates the 2MB.

Let B be an optimal solution to the BBSSP on $\overline{G}$ with cost matrix $\overline{C}$. (See the construction of $\overline{G}$ in Section 3.) Without loss of generality assume that B contains all edges of cost $-\infty$. Thus each vertex $j+n$, $j = 1, \ldots, n$ in B has an edge $(i, j+n), i \neq j$ incident on it representing the incoming arc (i,j) at vertex j in G which also represents the outgoing arc at vertex i in G. Since B is biconnected,

Table 1
Comparison of lower bounds.

	2MB	BAP	BBSSP($\overline{C}$)	BBSSP($\hat{C}$)	BSCSSP	BBP	EBBP	Complexity
2MB	=	▲	▲	✓	▲	▲	▲	$O(n^2)$
BAP	◀	=	✓	✓	✓	✓	✓	$O(n^{2.5})$
BBSSP($\overline{C}$)	◀	✓	=	✓	▲	▲	▲	$O(n^2)$
BBSSP($\hat{C}$)	✓	✓	✓	=	✓	✓	▲	$O(n^2)$
BSCSSP	◀	✓	◀	✓	=	=	▲	$O(n^2)$
BBP	◀	✓	◀	✓	=	=	▲	$O(n^2)$
EBBP	◀	✓	◀	◀	◀	◀	=	$O(n^{3.792})$

Table 2
Asymmetric BTSP lower bound summary on problem groups. The number listed under each bound name is the number of problems that bound, which gave the tightest lower bound in that problem group.

Problem set (# of problems)	2MB	BAP	BBSSP($\overline{C}$)	BBSSP($\hat{C}$)	BSCSSP	EBBP
coin (6)	0	0	6	1	1	6
crane (6)	4	4	0	4	6	6
disk (6)	6	6	0	5	6	6
rtilt (6)	2	2	0	2	6	6
shop (6)	5	5	0	5	6	6
stilt (6)	2	3	0	2	5	6
super (6)	6	6	5	6	6	6
balas (5)	0	0	0	0	5	5
tsplib: no ftv, ≤ 100 nodes (6)	1	1	2	2	5	6
tsplib: no ftv, >100 nodes (4)	0	4	0	0	0	0
tsplib: ftv, ≤ 100 nodes (9)	7	7	1	7	8	9
tsplib: ftv, >100 nodes (8)	2	8	0	2	2	3
ftv180 (1)	0	1	0	0	0	0
uk66 (1)	0	0	1	0	0	1
ran500 (5)	5	5	0	5	5	5
ran1000 (5)	5	5	0	5	5	5
Total (86)	45	57	15	46	66	76

Table 3
Asymmetric BTSP initial experimental results on selected problems.

Problem	Size	Opt. sol.	p	q	Best gap (%)	Avg gap (%)	Worst gap (%)	Avg time (s)
oin100.2	100	207	10	0	0.00	0.97	2.42	15.58
			5	5	0.00	0.48	1.45	16.95
rtilt100.0	100	260,342	10	0	10.46	13.64	18.12	47.34
			5	5	9.43	12.88	18.16	47.55
rtilt100.1	100	291,040	10	0	0.00	5.29	10.44	29.31
			5	5	0.00	9.65	17.33	41.22
rtilt100.2	100	227,248	10	0	20.70	28.57	36.99	51.01
			5	5	22.29	28.76	33.63	48.99
rtilt100.3	100	236,920	10	0	16.30	30.93	44.80	52.03
			5	5	22.79	29.48	44.78	54.63
rtilt100.4	100	294,367	10	0	5.78	8.32	12.01	49.96
			5	5	7.99	9.19	11.96	58.21
rtilt316.10	100	152,510	10	0	71.68	104.74	127.82	384.90
			5	5	96.33	101.68	110.68	490.10
shop100.0	100	2232	10	0	0.00	0.16	1.16	8.06
			5	5	0.00	0.06	0.58	5.57
shop316.10	316	2311	10	0	0.00	0.66	2.29	47.75
			5	5	0.00	1.11	2.34	92.42
stilt100.0	100	382,208	10	0	3.72	11.20	23.61	43.25
			5	5	7.43	11.20	20.96	45.81
stilt100.2	100	377,720	10	0	6.71	9.78	14.50	48.29
			5	5	5.56	11.80	19.12	48.93
stilt100.3	100	401,976	10	0	1.88	9.81	22.74	44.22
			5	5	2.24	7.04	19.50	42.77
stilt100.4	100	347,440	10	0	19.19	29.28	38.72	49.34
			5	5	24.40	29.43	38.10	52.61
stilt316.10	316	$226,504^{*}$	10	0	77.64	92.67	110.02	333.25
			5	5	83.18	91.39	99.89	352.84
ftv55	55	64	10	0	0.00	0.78	4.69	2.00
			5	5	0.00	0.63	3.13	2.35
ftv110	111	39	10	0	0.00	5.90	10.26	15.10
			5	5	0.00	7.95	10.26	19.03
ftv120	121	39	10	0	10.26	10.26	10.26	25.65
			5	5	0.00	5.38	10.26	16.85
ftv130	131	39	10	0	0.00	77.69	141.03	34.21
			5	5	0.00	27.18	135.90	21.42
ftv140	141	41	10	0	109.76	124.88	129.27	65.83
			5	5	0.00	86.10	129.27	53.34
ftv150	151	37	10	0	2.70	59.73	151.35	48.11
			5	5	0.00	57.30	140.54	51.01
rbg323	323	12	10	0	16.67	26.67	50.00	86.75
			5	5	16.67	20.00	50.00	94.78
rbg358	358	14	10	0	0.00	5.71	7.14	71.85
			5	5	0.00	4.29	7.14	75.06
rbg403	403	20	10	0	5.00	7.00	15.00	94.72
			5	5	5.00	5.50	10.00	110.60

the degree of node i, $i=1,\dots,2n$ is at least 2, the BBSSP($\overline{C}$) is no worse than the 2MB. □

Theorem 10. *The following pairs of bounds are non-dominated*:

(a) *The BAP bound and any of the bounds BBSSP($\hat{C}$), BBSSP($\overline{C}$), BSCSSP, BBP, or EBBP.*
(b) *The BSCSSP or BBP bound and the BBSSP($\hat{C}$) bound.*
(c) *The BBSSP($\hat{C}$) bound and the BBSSP($\overline{C}$) bound.*
(d) *The 2MB and the BBSSP($\hat{C}$) bounds.*

Proof. Consider the following cost matrices:

$$C_1=\begin{bmatrix} - & 3 & 9 & 1 & 1\\ 2 & - & 1 & 9 & 9\\ 9 & 9 & - & 1 & 9\\ 9 & 9 & 9 & - & 1\\ 1 & 1 & 9 & 9 & - \end{bmatrix},\quad C_2=\begin{bmatrix} - & 2 & 9 & 9 & 1\\ 2 & - & 1 & 9 & 9\\ 9 & 9 & - & 1 & 9\\ 9 & 1 & 9 & - & 3\\ 1 & 9 & 9 & 9 & - \end{bmatrix},$$

$$C_3=\begin{bmatrix} - & 1 & 3 & 3 & 3\\ 3 & - & 1 & 2 & 3\\ 1 & 3 & - & 3 & 1\\ 3 & 3 & 1 & - & 3\\ 2 & 3 & 3 & 1 & - \end{bmatrix},\quad C_4=\begin{bmatrix} - & 3 & 3 & 1 & 1\\ 3 & - & 3 & 1 & 3\\ 2 & 1 & - & 1 & 3\\ 3 & 3 & 3 & - & 3\\ 3 & 3 & 1 & 1 & - \end{bmatrix}.$$

The table below provides the lower bound values for each of these four cost matrices.

Bound	C_1	C_2	C_3	C_4
2MB	1	1	1	3
BAP	3	1	2	3
BBSSP($\hat{C}$)	1	2	2	1
BBSSP($\overline{C}$)	1	2	1	3
BSCSSP	1	2	1	3
EBBP	2	3	2	3
BTSP	3	3	3	3

Table 4
Complete asymmetric BTSP results (Part 1/2). Lower Bound (LB) algorithms: (a) 2MB, (b) BAP, (c) BBSSP($\hat{C}$), (d) BBSSP($\overline{C}$), (e) BSCSSP, (f) EBBP. Best/average/worst results reported from 10 trials for each problem.

Problem	Size	Tight LBs	Best LB	Best LB time	Best LB sol.	Opt. sol.	Best gap (%)	Avg. gap (%)	Worst gap (%)	Avg. Bin. steps	Avg. # LK calls	Avg. time (s)
coin100.0	100	c f	BBSSP($\hat{C}$)	0.00	253	253	0.00	0.00	0.00	0.00	2.50	0.97
coin100.1	100	c f	BBSSP($\hat{C}$)	0.00	219	219	0.00	0.00	0.00	0.00	1.60	0.42
coin100.2	100	c f	BBSSP($\hat{C}$)	0.00	203	207	0.00	0.48	1.45	10.80	46.20	16.95
coin100.3	100	c f	BBSSP($\hat{C}$)	0.02	232	232	0.00	0.00	0.00	0.00	2.40	0.85
coin100.4	100	c d e f	BBSSP($\hat{C}$)	0.00	214	214	0.00	0.00	0.00	0.00	1.00	0.11
coin316.10	316	c f	BBSSP($\hat{C}$)	0.05	227	227	0.00	0.00	0.00	0.00	3.60	6.51
crane100.0	100	a b d e f	2MB	0.00	173,390	173,390	0.00	0.00	0.00	0.00	1.00	0.12
crane100.1	100	a b d e f	2MB	0.00	152,923	152,923	17.80	17.80	17.80	14.00	60.00	25.34
crane100.2	100	a b d e f	2MB	0.00	214,843	214,843	0.00	0.00	0.00	0.00	1.00	0.11
crane100.3	100	e f	BSCSSP	0.00	145,622	145,622	0.00	0.00	0.00	0.00	1.90	0.65
crane100.4	100	e f	BSCSSP	0.02	171,484	171,484	0.00	0.00	0.00	0.00	1.00	0.20
crane316.10	316	a b d e f	2MB	0.00	119,345	120,333	0.00	0.00	0.00	16.00	62.00	128.23
disk100.0	100	a b d e f	2MB	0.00	508,034	508,034	0.00	0.00	0.00	0.00	1.00	0.12
disk100.1	100	a b d e f	2MB	0.00	473,495	473,495	0.00	0.00	0.00	0.00	1.00	0.11
disk100.2	100	a b d e f	2MB	0.00	382,677	382,677	1.08	1.08	1.08	13.00	32.00	10.78
disk100.3	100	a b d e f	2MB	0.00	453,657	453,657	0.00	0.00	0.00	0.00	1.00	0.13
disk100.4	100	a b d e f	2MB	0.00	415,696	415,696	0.00	0.00	0.00	0.00	1.00	0.20
disk316.10	316	a b e f	2MB	0.01	309,801	309,801	0.00	0.00	0.00	0.00	1.00	0.85
rtilt100.0	100	e f	BSCSSP	0.00	260,342	260,342	9.43	12.88	18.16	12.80	79.30	47.55
rtilt100.1	100	a b d e f	2MB	0.00	291,040	291,040	0.00	9.65	17.33	11.50	68.60	41.22
rtilt100.2	100	e f	BSCSSP	0.00	227,248	227,248	22.29	28.76	33.63	13.00	79.30	48.99
rtilt100.3	100	e f	BSCSSP	0.00	236,920	236,920	22.79	29.48	44.78	12.80	81.80	54.63
rtilt100.4	100	e f	BSCSSP	0.00	294,367	294,367	7.99	9.19	11.96	13.00	83.40	58.21
rtilt316.10	316	a b d e f	2MB	0.01	152,510	152,510	96.33	101.68	110.68	16.00	131.50	490.10
shop100.0	100	a b d e f	2MB	0.00	2232	2232	0.00	0.06	0.58	1.10	9.00	5.57
shop100.1	100	e f	BSCSSP	0.01	2608	2608	0.00	0.00	0.00	0.00	1.00	0.11
shop100.2	100	a b d e f	2MB	0.00	3620	3620	0.00	0.00	0.00	0.00	1.00	0.10
shop100.3	100	a b d e f	2MB	0.00	2526	2526	0.00	0.00	0.00	0.00	1.00	0.39
shop100.4	100	a b d e f	2MB	0.00	2792	2792	0.00	0.00	0.00	0.00	1.00	0.10
shop316.10	316	a b d e f	2MB	0.00	2311	2311	0.00	1.11	2.34	6.50	30.70	92.42
stilt100.0	100	e f	BSCSSP	0.00	382,208	382,208	7.43	11.20	20.96	12.80	75.20	45.81
stilt100.1	100	a b d e f	2MB	0.00	491,416	491,416	0.00	0.00	0.00	0.00	4.10	2.57
stilt100.2	100	e f	BSCSSP	0.00	377,720	377,720	5.56	11.80	19.12	12.80	80.10	48.93
stilt100.3	100	a b d e f	2MB	0.00	401,976	401,976	2.24	7.04	19.50	12.90	69.80	42.77
stilt100.4	100	e f	BSCSSP	0.00	347,440	347,440	24.40	29.43	38.10	12.80	85.00	52.61
stilt316.10	316	bf	BAP	0.80	226,504	?	83.18	91.39	99.89	16.40	102.10	352.84
super100.0	100	a b c d e f	2MB	0.00	10	10	0.00	0.00	0.00	0.00	1.00	0.09
super100.1	100	a b d e f	2MB	0.00	11	11	0.00	0.00	0.00	0.00	1.00	0.09
super100.2	100	a b c d e f	2MB	0.00	10	10	0.00	0.00	0.00	0.00	1.00	0.09
super100.3	100	a b c d e f	2MB	0.00	10	10	0.00	0.00	0.00	0.00	1.00	0.09
super100.4	100	a b c d e f	2MB	0.00	10	10	0.00	0.00	0.00	0.00	1.00	0.09
super316.10	316	a b c d e f	2MB	0.00	9	9	0.00	0.00	0.00	0.00	1.00	0.48
ftv180	181	b	BAP	0.03	35	37	0.00	0.00	0.00	8.00	27.50	25.08
uk66	66	c f	BBSSP($\hat{C}$)	0.00	170	170	0.00	0.00	0.00	0.00	1.00	0.05

Statement (a) is true by cost matrices C_1 and C_2. The remaining statements are true by cost matrices C_3 and C_4. □

Table 1 summarizes relative strengths of the lower bounds discussed. A '◂' in the table indicates that the bound representing the row dominates the bound representing the column. A '▴' in the table indicates that the bound representing the column dominates the bound representing the row. An '=' sign indicates that the two bounds are equivalent and a '✓' indicates that they are non-dominated.

4. A heuristic algorithm for the asymmetric BTSP

We now discuss our main heuristic algorithm for the asymmetric BTSP. The algorithm is similar to the one developed for the symmetric version in [29] with appropriate amendments to handle the asymmetry and specialized lower bounding schemes. The notable difference between theoretical approximability of the symmetric and asymmetric versions of the BTSP necessitates a systematic experimental analysis using asymmetric instances to understand the practical level of difficulty in solving the asymmetric BTSP in comparison to the symmetric BTSP.

Our heuristic is an approximate version of the threshold algorithm studied for various bottleneck problems [15]. The main ingredient of this algorithm is a feasibility test: "*Given an integer δ and a complete directed graph G, determine if G has a hamiltonian tour with bottleneck value no more than δ and, if yes, produce such a hamiltonian tour in G.*" Obviously, this is an NP-hard problem, so we explore ways to solve this approximately.

Consider the cost matrix $C^\delta = (c_{ij}^\delta)_{n\times n}$ where

$$c_{ij}^\delta = \begin{cases} 0 & \text{if } c_{ij} \le \delta, \\ c_{ij} & \text{otherwise.} \end{cases} \tag{9}$$

Solve a standard asymmetric TSP on G with cost matrix C^δ. The feasibility test has an 'yes' answer if and only if the optimal objective function value of the TSP is zero. By solving the TSP using a heuristic, we can answer the feasibility test in an approximate way. There are several ways to improve the accuracy of this approximation. One way is to employ different TSP heuristics on cost matrix C^δ. A more reasonable way is to apply the best known TSP heuristic on different, but equivalent, cost matrices generated using built-in randomness.

To achieve this goal, we construct a new cost matrix as follows. Let $z_1 < z_2 < \cdots < z_k$ be an ascending arrangement of all distinct costs in C. Generate positive random integers $r_1 < r_2 < \cdots < r_k$ in an

Table 5

Complete asymmetric BTSP results (Part 2/2). Lower Bound (LB) algorithms: (a) 2MB, (b) BAP, (c) BBSSP ($\hat{C}$), (d) BBSSP ($\overline{C}$), (e) BSCSSP, (f) EBBP. Best/average/worst results reported from 10 trials for each problem.

Problem	Size	Tight LBs	Best LB	Best LB time	Best LB sol.	Opt. sol.	Best gap (%)	Avg. gap (%)	Worst gap (%)	Avg. bin. steps	Avg. # LK calls	Avg. time (s)
balas84	84	e f	BSCSSP	0.00	18	18	0.00	0.00	0.00	0.00	1.00	0.07
balas108	108	e f	BSCSSP	0.00	13	13	0.00	0.00	0.00	0.00	1.00	0.09
balas120	120	e f	BSCSSP	0.00	22	22	0.00	0.00	0.00	0.00	1.00	0.11
balas160	160	e f	BSCSSP	0.00	13	13	0.00	0.00	0.00	0.00	1.00	0.17
balas200	200	e f	BSCSSP	0.02	13	13	0.00	0.00	0.00	0.00	1.00	0.39
ran500.0	500	a b d e f	2MB	0.02	24	24	0.00	0.00	0.00	0.00	1.00	1.23
ran500.1	500	a b d e f	2MB	0.02	22	22	0.00	0.00	0.00	0.00	1.10	3.73
ran500.2	500	a b d e f	2MB	0.02	23	23	0.00	0.00	0.00	0.00	1.00	2.73
ran500.3	500	a b d e f	2MB	0.02	23	23	0.00	0.00	0.00	0.00	1.00	2.43
ran500.4	500	a b d e f	2MB	0.01	28	28	0.00	0.00	0.00	0.00	1.00	1.89
ran1000.0	1000	a b d e f	2MB	0.03	18	18	0.00	0.00	0.00	0.00	1.00	4.61
ran1000.1	1000	a b d e f	2MB	0.03	17	17	0.00	0.00	0.00	0.00	1.40	7.52
ran1000.2	1000	a b d e f	2MB	0.03	17	17	0.00	0.00	0.00	0.00	1.30	9.46
ran1000.3	1000	a b d e f	2MB	0.03	19	19	0.00	0.00	0.00	0.00	1.30	8.25
ran1000.4	1000	a b d e f	2MB	0.03	19	19	0.00	0.00	0.00	0.00	1.10	7.61
br17	17	c d e f	BBSSP($\hat{C}$)	0.00	8	8	0.00	0.00	0.00	0.00	1.00	0.01
ft53	53	e f	BSCSSP	0.02	977	977	0.00	0.00	0.00	0.00	1.00	0.04
ft70	70	e f	BSCSSP	0.00	1398	1398	0.00	0.00	0.00	0.00	1.00	0.05
ftv33	34	a b d e f	2MB	0.00	113	113	0.00	0.00	0.00	0.00	1.00	0.02
ftv35	36	a b d e f	2MB	0.00	113	113	0.00	0.00	0.00	0.00	1.00	0.02
ftv38	39	a b d e f	2MB	0.00	113	113	0.00	0.00	0.00	0.00	1.00	0.03
ftv44	45	a b d e f	2MB	0.00	113	113	0.00	0.00	0.00	0.00	1.00	0.03
ftv47	48	a b d e f	2MB	0.00	104	104	0.00	0.00	0.00	0.00	1.00	0.03
ftv55	56	c f	BBSSP($\hat{C}$)	0.02	64	64	0.00	0.63	3.13	2.40	11.20	2.35
ftv64	65	a b d e f	2MB	0.00	104	104	0.00	0.00	0.00	0.00	1.00	0.05
ftv70	71	a b d e f	2MB	0.00	104	104	0.00	0.00	0.00	0.00	1.00	0.06
ftv90	91	e f	BSCSSP	0.00	48	48	0.00	0.00	0.00	0.00	2.70	1.21
ftv100	101	a b d e f	2MB	0.00	53	53	0.00	0.00	0.00	0.00	1.20	0.66
ftv110	111	b	BAP	0.02	39	39	0.00	7.95	10.26	7.20	33.40	19.03
ftv120	121	b	BAP	0.02	39	39	0.00	5.38	10.26	4.80	26.00	16.85
ftv130	131	b f	BAP	0.03	39	39	0.00	27.18	135.90	3.20	24.40	21.42
ftv140	141	a b d e f	2MB	0.00	41	41	0.00	86.10	129.27	5.60	44.20	53.34
ftv150	151	b	BAP	0.02	35	37	0.00	57.30	140.54	8.10	48.30	51.01
ftv160	161	b	BAP	0.02	35	37	0.00	0.00	0.00	8.00	31.90	26.14
ftv170	171	b	BAP	0.03	35	37	0.00	0.00	0.00	8.00	29.40	24.24
kro124p	100	a b d e f	2MB	0.00	607	607	0.00	0.00	0.00	0.00	1.20	0.24
p43	43	e f	BSCSSP	0.00	5008	5008	0.00	0.00	0.00	0.00	1.00	0.02
rbg323	323	b	BAP	0.25	12	12	16.67	20.00	50.00	4.10	26.00	94.78
rbg358	358	b	BAP	0.13	14	14	0.00	4.29	7.14	4.00	20.10	75.06
rbg403	403	b	BAP	0.73	20	20	5.00	5.50	10.00	4.00	27.80	110.60
rbg443	443	b	BAP	0.94	20	20	15.00	15.00	15.00	4.00	32.00	138.43
ry48p	48	c f	BBSSP($\hat{C}$)	0.00	550	577	0.00	0.00	0.00	10.00	38.00	5.17

interval [a, b]. Define the cost matrix $C^{\delta,r} = (c_{ij}^{\delta,r})_{n\times n}$ where

$$c_{ij}^{\delta,r} = \begin{cases} 0 & \text{if } c_{ij} \le \delta, \\ z_l + r_l & \text{otherwise, where } c_{ij} = z_l. \end{cases} \quad (10)$$

Note that the TSP with cost matrix C^δ has an optimal tour of cost zero if and only if the TSP with cost matrix $C^{\delta,r}$ has an optimal tour of cost zero. If a tour of non-zero value is constructed by the TSP heuristic, a new matrix $C^{\delta,r}$ can be generated (we call this a 'shake' operation) and the TSP heuristic can be applied on the new cost matrix. The process can be repeated a prescribed number of times, or "shakes". If after each shake the TSP heuristic outputs a non-zero tour length, we can conclude with high probability that the answer to the feasibility test is 'no'. The construction of $C^{\delta,r}$ is designed to discourage using edges of large cost in the solution produced by the TSP heuristic. Each time we solve a TSP using a heuristic, the bottleneck value of the resulting tour is also noted and upon termination, the best such tour, say H^*, is returned. A formal description of this 'feasibility test' procedure is summarized in Algorithm 2 which we call procedure '*IsFeasible*'. In all our experiments we selected the interval [a, b] as [1, n^2].

Algorithm 2. *IsFeasible*$(n, C, \delta, \alpha, p, q)$.

Input: A problem on n nodes with cost matrix C, integer δ, TSP solver/heuristic α, and integers p and q, which represent the number of iterations with cost matrix C^δ and $C^{\delta,r}$, respectively.
Output: The 3-tuple $(feasible, tour, max\_cost)$ where *feasible* is a Boolean value that indicates if a Hamiltonian cycle was found using only costs less than or equal to δ, *tour* is the feasible/best tour found, and *max_cost* is the largest cost in *tour*.

```
minmax_cost ← ∞;
best_tour ← ∅;
for i = 1, …, p do
    (length, tour) ← α(n, C^δ);
    max_cost ← max{c_ij : (i,j) ∈ tour};
    if length = 0 then
        return (TRUE, tour, max_cost);
    else
        if max_cost < minmax_cost then
            minmax_cost ← max_cost;
            best_tour ← tour;
        end
    end
end
for i = 1, …, q do
    Let r ← {r_1, r_2, …, r_k} be a list of random integers
    such that 1 ≤ r_1 < r_2 < ⋯ < r_k ≤ n^2;
    (length, tour) ← α(n, C^δ, r);
    max_cost ← max{c_ij : (i,j) ∈ tour};
    if length = 0 then
        return (TRUE, tour, max_cost);
    else
        if max_cost < minmax_cost then
            minmax_cost ← max_cost;
            best_tour ← tour;
        end
    end
end
return (FALSE, best_tour, minmax_cost);
```

Let $U = \max\{c_{ij} : (i,j) \in H^*\}$ where H^* is a heuristic solution to the asymmetric BTSP with objective function value U. We try to improve this value using a binary search over the edge costs in the range [L, U] where L is a BTSP lower bound. The algorithm *IsFeasible* is used to guide the binary search. A formal description of this scheme is summarized in Algorithm 3.

Algorithm 3. *BTSPThreshold*(n, C, l, α, p, q).

Input: A problem on n nodes with cost matrix C, a lower bound l, TSP solver/heuristic α, and integers p and q, which represent the number of iterations with cost matrix C^δ and $C^{\delta,r}$, respectively.
Output: The (optimal/heuristic conclusion) on the BTSP objective value and tour.

```
(feasible, best_tour, max_cost) ← IsFeasible(n, C, l, α, p, q);
if feasible then return (l, best_tour);
;
Let z_1 < z_2 < ⋯ < z_k be a list of the unique ordered costs from C
    in non-increasing order;
low ← l; high ← k;
while low ≠ high do
    med ← ((high − low)/2) + low;
    (feasible, tour, max_cost) ← IsFeasible(n, C, z_med, α, p, q);
    if feasible then
        high ← median;
        best_tour ← tour;
    else
        low ← median + 1;
    end
end
return (z_low, best_tour);
```

In our implementation, the asymmetric TSP heuristic α used within procedure *IsFeasible* is Concorde's implementation of the Lin–Kernighan algorithm [4,21,30] after converting the asymmetric instance into a symmetric instance using the transformation described in Section 3. The Lin–Kernighan heuristic should naturally force the fixed edges of cost $-\infty$ into the tour with no additional constraint necessary, but we make sure that these edges are present to ensure any tour found in the symmetric instance is valid for the asymmetric instance. Instead of using this transformation, one could use any asymmetric TSP heuristic but we found it more convenient and effective to use the LK-algorithm on the transformed problem.

5. Computational experiments

The lower bounding schemes discussed in Section 3 and the heuristic algorithms discussed in Section 4 were coded in C with the GNU C compiler and tested on a PC with 3.40 GHz Pentium 4 CPU and 2 GB of RAM running Microsoft Windows XP SP2 operating system and Cygwin NT 5.1. All reported running times are in CPU seconds rounded to two decimal places and include input and output times.

The test bed primarily consists of all available benchmark asymmetric TSP instances studied in the literature. We first selected 86 problems from the test instances. These are primarily the problems considered in [16]. Details of these instances are summarized below.

(a) 42 instances by Cirasella, Johnson, McGeoch, and Zhang that simulate real-world applications in various fields, as described

Table 6
Asymmetric MSTSP upper bound summary on problem groups. The number listed under each bound name is the number of problems that bound, which gave the tightest upper bound in that problem group.

Problem set (# of problems)	2MB	BAP	BBSSP($\overline{C}$)	BBSSP($\hat{C}$)	BSCSSP	EBBP
coin (6)	0	5	1	0	0	2
crane (6)	1	6	0	1	1	2
disk (6)	1	6	1	1	1	2
rtilt (6)	5	6	0	5	5	6
shop (6)	5	6	0	5	5	5
stilt (6)	4	6	0	4	4	6
super (6)	6	6	5	6	6	6
balas (5)	5	5	0	5	5	5
tsplib: no ftv, ≤ 100 nodes (6)	1	6	1	1	1	2
tsplib: no ftv, >100 nodes (4)	0	4	0	0	0	0
tsplib: ftv, ≤ 100 nodes (9)	1	9	0	1	1	1
tsplib: ftv, >100 nodes (8)	0	8	0	0	0	0
ftv180 (1)	0	1	0	0	0	0
uk66 (1)	0	1	0	0	0	0
ran500 (5)	5	5	0	5	5	5
ran1000 (5)	5	5	0	5	5	5
Total (86)	39	85	8	39	39	47

Table 7
Complete asymmetric MTSP results (Part 1/2). Upper Bound (UB) algorithms: (a) 2MB, (b) BAP, (c) BBSSP ($\hat{C}$), d) BBSSP ($\overline{C}$), (e) BSCSSP, (f) EBBP. Best/average/worst results reported from 10 trials for each problem.

Problem	Size	Tight UBs	Best UB	Best UB Time	Best UB sol.	Opt. sol.	Best gap (%)	Avg. gap (%)	Worst gap (%)	Avg. bin. steps	Avg. # LK calls	Avg. time (s)
coin100.0	100	b	BAP	0.03	896	891	1.35	1.63	2.13	10.00	57.30	40.99
coin100.1	100	b	BAP	0.03	858	858	0.00	0.59	1.05	5.60	24.80	14.51
coin100.2	100	b f	BAP	0.02	974	974	1.85	2.46	2.77	10.00	56.30	34.58
coin100.3	100	b	BAP	0.03	890	890	2.92	3.31	3.93	9.90	52.20	34.12
coin100.4	100	c f	BBSSP ($\hat{C}$)	0.00	903	903	0.00	0.12	0.78	6.00	19.40	10.90
coin316.10	316	b	BAP	0.59	1684	1684	2.61	3.05	3.56	10.80	63.60	293.33
crane100.0	100	b	BAP	0.03	647,354	647,354	0.00	0.00	0.00	0.00	1.10	0.30
crane100.1	100	a b d e f	2MB	0.00	627,751	627,751	0.00	0.00	0.00	0.00	1.00	0.11
crane100.2	100	b	BAP	0.00	645,372	645,372	0.00	0.00	0.00	0.00	1.00	0.22
crane100.3	100	b	BAP	0.02	629,932	629,932	0.07	0.32	0.48	12.90	55.50	36.57
crane100.4	100	b f	BAP	0.02	619,054	619,054	0.00	0.00	0.00	0.00	1.10	0.23
crane316.10	316	b	BAP	0.28	664,713	664,713	1.43	2.25	2.83	16.20	90.40	385.49
disk100.0	100	b	BAP	0.03	4,767,083	4,767,083	10.11	14.76	16.51	12.60	94.30	78.04
disk100.1	100	b	BAP	0.03	4,882,684	4,882,684	0.28	0.30	0.36	13.00	48.50	31.31
disk100.2	100	b c f	BBSSP ($\hat{C}$)	0.00	4,959,789	4,959,789	0.00	0.00	0.00	0.00	1.80	1.24
disk100.3	100	b	BAP	0.03	4,663,663	4,663,663	0.95	2.03	2.52	12.90	74.70	57.06
disk100.4	100	a b d e f	2MB	0.00	4,849,971	4,849,971	1.61	4.32	17.20	12.70	72.70	57.94
disk316.10	316	b	BAP	0.91	4,947,068	4,947,068	7.66	11.57	19.41	16.20	107.00	416.43
rtilt100.0	100	a b d e f	2MB	0.00	529,202	529,202	0.00	0.00	0.00	0.00	1.00	0.24
rtilt100.1	100	a b d e f	2MB	0.00	546,234	546,234	0.00	0.00	0.00	0.00	1.00	0.13
rtilt100.2	100	b f	BAP	0.02	494,714	494,714	0.00	0.00	0.00	0.00	1.00	0.16
rtilt100.3	100	a b d e f	2MB	0.00	500,897	500,897	0.00	0.00	0.00	0.00	1.00	0.13
rtilt100.4	100	a b d e f	2MB	0.00	517,383	517,383	0.00	0.00	0.00	0.00	1.00	0.12
rtilt316.10	316	a b d e f	2MB	0.00	509,367	509,367	0.00	0.00	0.00	0.00	1.00	1.06
shop100.0	100	a b d e f	2MB	0.00	1939	1939	0.00	0.00	0.00	2.20	9.90	6.12
shop100.1	100	a b d e f	2MB	0.00	1786	1786	0.00	0.00	0.00	0.00	1.00	0.10
shop100.2	100	b	BAP	0.03	2466	2466	5.15	6.25	8.72	10.70	71.40	57.33
shop100.3	100	a b d e f	2MB	0.00	2044	2044	0.00	0.00	0.00	0.00	1.00	0.37
shop100.4	100	a b d e f	2MB	0.00	2104	2104	0.00	0.00	0.00	1.10	6.80	4.07
shop316.10	316	a b d e f	2MB	0.01	1966	1966	0.00	0.00	0.00	0.00	1.10	1.06
stilt100.0	100	a b d e f	2MB	0.00	1,011,282	1,011,282	0.00	0.00	0.00	0.00	1.00	0.11
stilt100.1	100	a b d e f	2MB	0.00	1,071,396	1,071,396	0.00	0.00	0.00	0.00	1.00	0.14
stilt100.2	100	a b d e f	2MB	0.00	957,076	957,076	0.00	0.00	0.00	0.00	1.00	0.11
stilt100.3	100	b f	BAP	0.03	997,468	997,468	0.00	0.00	0.00	0.00	1.00	0.12
stilt100.4	100	a b d e f	2MB	0.00	985,154	985,154	0.00	0.00	0.00	0.00	1.00	0.12
stilt316.10	316	b f	BAP	0.52	990,472	990,472	0.00	0.00	0.00	0.00	1.00	0.85
super100.0	100	a b c d e f	2MB	0.00	16	16	0.00	0.00	0.00	0.00	1.00	0.09
super100.1	100	a b c d e f	2MB	0.00	16	16	0.00	0.00	0.00	0.00	1.00	0.09
super100.2	100	a b d e f	2MB	0.00	16	16	0.00	0.00	0.00	0.00	1.00	0.09
super100.3	100	a b c d e f	2MB	0.00	16	16	0.00	0.00	0.00	0.00	1.00	0.09
super100.4	100	a b c d e f	2MB	0.00	16	16	0.00	0.00	0.00	0.00	1.00	0.10
super316.10	316	a b c d e f	2MB	0.00	17	17	0.00	0.00	0.00	0.00	1.00	0.49
ftv180	181	b	BAP	0.03	180	180	0.00	0.00	0.00	0.00	1.00	0.34
uk66	66	b	BAP	0.00	609	604	0.33	0.66	0.99	9.00	39.80	12.52

in [13]. There are seven groups of problems, each including five instances of 100 nodes and a single instance of 316 nodes:
- coin: Pay phone coin collection instances.
- crane: Random Euclidean stacker crane instances.
- disk: Disk drive instances.
- rtilt: Tilted drilling machine instances with additive norm.
- shop: No-wait flow shop instances.
- stilt: Tilted drilling machine instances with sup norm.
- super: Approximate shortest common superstring instances.

(b) 5 scheduling instances generated by Balas that simulate an application in a Dupont chemical plant. There are five problems in total of sizes 84, 108, 120, 160, and 200 nodes [7].

(c) All 27 TSPLIB instances are maintained by Reinelt [42]. We subdivide them into the problems labeled “ftv” and those that are not, as well as subdivide them into problems with 100 nodes or less and problems with more than 100 nodes.

(d) 2 real-world instances (ftv180 and uk66) and 5 random instances with integer costs uniformly generated in the range [1, 1000] of 500 nodes and 1000 nodes each. These were created by Fischetti et al. [16].

We also considered the remaining 288 benchmark instances available in the literature. The summary of the results on these problems is given in the Appendix. The computational results are presented in two sets. First, we compared various lower bounding schemes discussed in Section 3 and summarized the results in Table 2. The 86 problem instances are grouped into the 16 groups as described earlier for a compact presentation of the results. The number given for each bound and problem group indicates the number of problems in that group for which the lower bound achieved the tightest result. With the exception of the EBBP bound, all other bounds generally take less than 1 s to run, even on problems of 1000 vertices. As expected, the EBBP bound, being an $O(n^4)$ algorithm, is expensive to calculate, and did not provide a tighter lower bound than the best of the other lower bounds we tested for the asymmetric BTSP. However, if we are looking to choose a single lower bounding scheme, the EBBP performed consistently better.

Next, we tested Algorithm 3 on the same problem set with two experiments to observe the effects of “shaking” (i.e. using cost matrix $C^{\delta,r}$ versus cost matrix C^{δ}). For one experiment, we set $p=10$ and $q=0$, corresponding to 10 attempts with cost matrix C^{δ} and 0 attempts with cost matrix $C^{\delta,r}$, respectively; the second we set $p=5$ and $q=5$. The idea was to see if splitting the effort between both cost matrix formulations would be more beneficial than simply using cost matrix C^{δ}. We make no claim that these

Table 8
Complete asymmetric MSTSP results (Part 2/2). Upper Bound (UB) algorithms: (a) 2MB, (b) BAP, (c) BBSSP ($\hat{C}$), (d) BBSSP ($\overline{C}$), (e) BSCSSP, (f) EBBP. Best/average/worst results reported from 10 trials for each problem.

Problem	Size	Tight UBs	Best UB	Best UB time	Best UB sol.	Opt. sol.	Best gap (%)	Avg. gap (%)	Worst gap (%)	Avg. bin. steps	Avg. # LK calls	Avg. time (s)
balas84	84	a b d e f	2MB	0.00	29	29	0.00	0.00	0.00	0.00	1.00	0.06
balas108	108	a b d e f	2MB	0.00	24	24	0.00	0.00	0.00	0.00	1.00	0.09
balas120	120	a b d e f	2MB	0.00	29	29	0.00	0.00	0.00	0.00	1.60	1.04
balas160	160	a b d e f	2MB	0.00	31	31	0.00	0.00	0.00	0.00	1.00	0.18
balas200	200	a b d e f	2MB	0.00	32	32	0.00	0.00	0.00	0.00	1.00	0.23
ran500.0	500	a b d e f	2MB	0.02	1000	1000	0.00	0.00	0.00	0.00	1.10	2.06
ran500.1	500	a b d e f	2MB	0.02	997	997	0.00	0.00	0.00	0.00	1.00	1.28
ran500.2	500	a b d e f	2MB	0.02	997	997	0.00	0.00	0.00	0.00	1.00	1.43
ran500.3	500	a b d e f	2MB	0.02	998	998	0.00	0.00	0.00	0.00	1.00	1.92
ran500.4	500	a b d e f	2MB	0.00	996	996	0.00	0.00	0.00	0.00	1.00	1.08
ran1000.0	1000	a b d e f	2MB	0.06	1000	1000	0.00	0.00	0.00	0.00	1.30	9.79
ran1000.1	1000	a b d e f	2MB	0.05	1005	1005	0.00	0.00	0.00	0.00	1.10	7.55
ran1000.2	1000	a b d e f	2MB	0.06	1004	1004	0.00	0.00	0.00	0.00	1.20	5.82
ran1000.3	1000	a b d e f	2MB	0.05	1004	1004	0.00	0.00	0.00	0.00	1.00	2.60
ran1000.4	1000	a b d e f	2MB	0.05	1006	1006	0.00	0.00	0.00	2.00	6.50	50.92
br17	17	b	BAP	0.00	5	5	0.00	0.00	0.00	0.00	1.00	0.02
ft53	53	b f	BAP	0.00	379	379	0.00	0.00	0.00	0.00	1.00	0.06
ft70	70	b	BAP	0.00	976	976	0.20	0.30	0.31	9.00	30.20	9.81
ftv33	34	b	BAP	0.00	143	143	0.00	0.00	0.00	0.00	1.00	0.02
ftv35	36	b	BAP	0.00	154	154	0.00	0.00	0.00	0.00	1.00	0.02
ftv38	39	b	BAP	0.02	154	154	0.00	0.00	0.00	0.00	1.00	0.03
ftv44	45	a b d e f	2MB	0.00	162	162	0.00	0.00	0.00	0.00	1.00	0.03
ftv47	48	b	BAP	0.00	168	168	0.00	0.00	0.00	0.00	1.00	0.03
ftv55	56	b	BAP	0.00	154	154	0.00	0.00	0.00	0.00	1.00	0.05
ftv64	65	b	BAP	0.00	160	160	0.00	0.00	0.00	0.00	1.00	0.05
ftv70	71	b	BAP	0.00	161	161	0.00	0.00	0.00	0.00	1.00	0.09
ftv90	91	b	BAP	0.02	148	148	2.70	3.04	4.05	7.00	38.30	22.67
ftv100	101	b	BAP	0.03	155	155	0.65	1.81	2.58	7.10	35.70	23.17
ftv110	111	b	BAP	0.02	165	165	0.00	1.03	1.82	6.90	27.40	21.22
ftv120	121	b	BAP	0.02	165	165	0.00	0.00	0.00	0.00	1.20	0.87
ftv130	131	b	BAP	0.02	172	172	0.00	0.00	0.00	0.00	1.00	0.17
ftv140	141	b	BAP	0.02	172	172	0.00	0.00	0.00	0.00	1.10	0.46
ftv150	151	b	BAP	0.03	178	178	0.00	0.00	0.00	0.00	1.00	0.23
ftv160	161	b	BAP	0.02	178	178	0.00	0.00	0.00	0.00	1.00	0.19
ftv170	171	b	BAP	0.02	180	180	0.00	0.00	0.00	0.00	1.00	0.94
kro124p	100	a b c d e f	2MB	0.00	2347	2347	0.00	0.00	0.00	0.00	1.00	0.15
p43	43	b	BAP	0.02	17	17	0.00	1.76	5.88	1.60	11.30	2.12
rbg323	323	b	BAP	0.17	23	23	8.70	8.70	8.70	4.00	24.20	78.19
rbg358	358	b	BAP	0.34	21	21	0.00	0.00	0.00	0.00	1.70	4.49
rbg403	403	b	BAP	0.41	19	19	0.00	0.53	5.26	2.00	11.90	43.47
rbg443	443	b	BAP	0.75	18	18	0.00	0.00	0.00	0.00	2.10	8.90
ry48p	48	b	BAP	0.00	1232	1232	2.27	3.20	4.30	9.80	49.90	11.43

Table 9
Heuristic results on random asymmetric matrices (amat), pay phone collection instances (coin), random Euclidean stacker crane instances (crane), and disk drive instances (disk).

Problem	Lower bound	Best obj.	Avg. time (s)	Opt.?	Problem	Lower bound	Best obj.	Avg. time (s)	Opt.?
amat100.0	50,981	50,981	0.13	Yes	amat316.12	26,551	26,551	0.88	Yes
amat100.1	50,821	50,821	0.16	Yes	amat316.13	21,695	21,695	1.04	Yes
amat100.2	63,674	63,674	0.13	Yes	amat316.14	17,539	17,539	2.64	Yes
amat100.3	62,994	62,994	0.17	Yes	amat316.15	19,952	19,952	2.07	Yes
amat100.4	55,534	55,534	0.15	Yes	amat316.16	20,779	20,779	1.18	Yes
amat100.5	44,548	44,548	0.15	Yes	amat316.17	26,165	26,165	0.95	Yes
amat100.6	72,650	72,650	0.15	Yes	amat316.18	17,207	17,207	1.71	Yes
amat100.7	59,291	59,291	0.15	Yes	amat316.19	43,608	43,608	2.36	Yes
amat100.8	55,462	55,462	0.13	Yes	amat1000.20	9978	9978	15.35	Yes
amat100.9	49,928	49,928	0.13	Yes	amat1000.21	10,624	10,624	15.81	Yes
amat316.10	21,896	21,896	0.98	Yes	amat1000.22	6715	6715	29.15	Yes
amat316.11	20,451	20,451	1.25	Yes	amat3162.30	2985	47,237	5760.72	
coin100.0	253	253	14.49	Yes	coin316.12	225	225	72.60	Yes
coin100.1	219	219	18.85	Yes	coin316.13	245	245	66.07	Yes
coin100.2	203	207	16.18		coin316.14	278	278	88.84	Yes
coin100.3	232	232	20.84	Yes	coin316.15	282	282	56.75	Yes
coin100.4	214	214	0.16	Yes	coin316.16	243	243	47.92	Yes
coin100.5	244	244	18.60	Yes	coin316.17	277	277	88.05	Yes
coin100.6	239	239	9.68	Yes	coin316.18	277	277	66.54	Yes
coin100.7	265	265	20.47	Yes	coin316.19	259	259	0.53	Yes
coin100.8	198	201	15.54		coin1000.20	278	278	341.00	Yes
coin100.9	243	243	21.38	Yes	coin1000.21	327	327	422.52	Yes
coin316.10	227	227	82.78	Yes	coin1000.22	245	249	279.33	
coin316.11	238	238	87.09	Yes	coin3162.30	260	649	4075.57	
crane100.0	173,390	173,390	0.12	Yes	crane316.12	132,750	132,750	2.34	Yes
crane100.1	152,923	180,146	30.65		crane316.13	102,898	102,898	0.95	Yes
crane100.2	214,843	214,843	0.14	Yes	crane316.14	145,963	145,963	0.73	Yes
crane100.3	145,622	145,622	0.28	Yes	crane316.15	128,548	128,548	192.00	Yes
crane100.4	171,484	171,484	0.26	Yes	crane316.16	111,939	111,939	1.71	Yes
crane100.5	205,739	205,739	0.12	Yes	crane316.17	102,571	102,571	1.42	Yes
crane100.6	185,071	185,071	0.17	Yes	crane316.18	106,821	106,821	1.92	Yes
crane100.7	200,173	200,173	0.12	Yes	crane316.19	133,399	133,399	0.86	Yes
crane100.8	205,258	205,258	0.18	Yes	crane1000.20	56,343	56,343	19.44	Yes
crane100.9	192,987	192,987	23.40	Yes	crane1000.21	61,720	64,450	722.66	
crane316.10	119,345	120,333	136.68		crane1000.22	58,091	58,466	489.04	
crane316.11	108,045	108,045	1.08	Yes	crane3162.30	41,751	76,894	6234.88	
disk100.0	508,034	508,034	0.12	Yes	disk316.12	273,516	273,516	1.41	Yes
disk100.1	473,495	473,495	0.10	Yes	disk316.13	236,394	236,394	32.21	Yes
disk100.2	382,677	386,809	10.39		disk316.14	226,920	226,920	2.64	Yes
disk100.3	453,657	453,657	0.15	Yes	disk316.15	271,724	271,724	1.50	Yes
disk100.4	415,696	415,696	0.19	Yes	disk316.16	249,590	249,590	7.44	Yes
disk100.5	553,069	553,069	0.21	Yes	disk316.17	305,813	305,813	1.01	Yes
disk100.6	432,563	432,563	0.13	Yes	disk316.18	246,356	246,356	1.37	Yes
disk100.7	550,543	550,543	0.16	Yes	disk316.19	320,680	320,680	1.16	Yes
disk100.8	396,159	396,159	0.40	Yes	disk1000.20	190,741	190,741	975.74	Yes
disk100.9	426,697	426,697	0.22	Yes	disk1000.21	190,665	190,665	746.68	Yes
disk316.10	309,801	309,801	0.82	Yes	disk1000.22	171,509	195,825	1537.19	
disk316.11	259,308	259,308	1.33	Yes	disk3162.30	114,028	275,891	5957.56	

values are the best choices for p and q, but these choices appear to be reasonable for the problems in our test set.

In both experiments, we set l equal to the strongest lower bound found. Our TSP heuristic α was Concorde's implementation of the Lin–Kernighan algorithm [4] applied on the symmetric instance obtained using the transformation discussed in Section 3 on cost matrix C^{δ} and $C^{\delta,r}$. We call Concorde's Lin–Kernighan algorithm with the default parameters and five random restarts.

Optimality is generally verified by the existence of a tight lower bound. For problems where the best found solution value was not equal to a lower bound optimality could not be verified directly. To test the quality of such solutions we used the exact version of Algorithm 3 where Concorde's exact TSP solver is used in place of α. We were unable to verify the optimality of 'stilt316.10' with Concorde due to an integer overflow error, so instead we compare heuristic results against the best lower bound.

Our heuristic algorithm 3 worked extremely well in terms of solution quality and running time. Out of the 86 instances, the algorithm consistently found the optimal solution to 60 of the instances. We focused primarily on 23 selected problems in Table 3. These are problems where a consistent solution was not found for each of the 10 trials. We present the best, average, and worst solution gaps found from the optimal solution value over 10 trials, i.e. if b is the (best/average/worst) bottleneck solution found by our algorithm and b^* is the optimal solution value then

$$\text{gap}\% = (b - b^*)/b^* \times 100.$$

The average time reported includes output times, which were negligible (generally much less than 0.10 s).

We noticed that performing shake operations ($p=5$, $q=5$) generally produced solutions with a lower average and lower worst gaps from the optimal solution. This indicates that shake operations are a promising idea for helping the Lin–Kernighan algorithm to find good tours. Although the results are generally excellent, our heuristic performs poorly on the 'rtilt' and 'stilt' class of problems. These problems have many distinct, large integer

Table 10
Heuristic results on random two-dimensional rectilinear instances (rect), tilted drilling machine instances, additive norm (rtilt), no-wait flowshop instances (shop), and random symmetric matrices (smat problems).

Problem	Lower bound	Best obj.	Avg. time (s)	Opt.?	Problem	Lower bound	Best obj.	Avg. time (s)	Opt.?
rect100.0	223,717	223,717	12.40	Yes	rect316.12	143,209	143,209	134.99	Yes
rect100.1	296,349	296,349	28.72	Yes	rect316.13	116,995	116,995	104.04	Yes
rect100.2	198,319	198,319	22.94	Yes	rect316.14	140,437	140,437	158.32	Yes
rect100.3	252,297	252,297	24.45	Yes	rect316.15	144,858	144,858	118.90	Yes
rect100.4	246,494	246,494	27.80	Yes	rect316.16	142,029	142,029	110.67	Yes
rect100.5	270,406	270,406	22.01	Yes	rect316.17	121,054	121,054	110.10	Yes
rect100.6	226,069	226,069	18.40	Yes	rect316.18	166,616	166,616	217.73	Yes
rect100.7	245,457	245,457	21.60	Yes	rect316.19	142,134	142,134	151.49	Yes
rect100.8	259,347	259,347	27.80	Yes	rect1000.20	88,925	88,925	521.58	Yes
rect100.9	199,114	200,000	22.88		rect1000.21	82,241	82,241	543.19	Yes
rect316.10	160,358	160,358	168.87	Yes	rect1000.22	73,643	73,643	398.07	Yes
rect316.11	133,083	133,083	121.94	Yes	rect3162.30	47,063	123,706	5241.82	
rtilt100.0	260,342	286,962	38.31		rtilt316.12	162,936	272,157	337.92	
rtilt100.1	291,040	291,040	33.51	Yes	rtilt316.13	141,912	249,325	335.80	
rtilt100.2	227,248	270,913	43.60		rtilt316.14	157,594	281,467	361.11	
rtilt100.3	236,920	288,191	47.37		rtilt316.15	181,676	231,232	395.26	
rtilt100.4	294,367	307,304	45.41		rtilt316.16	173,991	273,055	358.35	
rtilt100.5	238,332	280,052	42.26		rtilt316.17	128,217	317,338	276.98	
rtilt100.6	276,224	276,947	33.44		rtilt316.18	147,764	263,831	334.85	
rtilt100.7	361,152	361,152	0.33	Yes	rtilt316.19	133,664	270,363	387.19	
rtilt100.8	368,500	368,500	1.24	Yes	rtilt1000.20	86,549	287,949	1130.05	
rtilt100.9	198,376	307,809	50.76		rtilt1000.21	86,828	323,513	1329.22	
rtilt316.10	152,510	261,362	431.86		rtilt1000.22	92,692	319,739	1218.79	
rtilt316.11	139,280	240,201	392.59		rtilt3162.30	47,152	391,531	3980.61	
shop100.0	2232	2232	8.89	Yes	shop316.12	2541	2541	0.40	Yes
shop100.1	2608	2608	0.16	Yes	shop316.13	2970	2970	0.42	Yes
shop100.2	3620	3620	0.10	Yes	shop316.14	2235	2235	4.38	Yes
shop100.3	2526	2526	0.40	Yes	shop316.15	2459	2459	0.70	Yes
shop100.4	2792	2792	0.10	Yes	shop316.16	2806	2806	0.43	Yes
shop100.5	2679	2679	0.09	Yes	shop316.17	2298	2298	5.34	Yes
shop100.6	2314	2314	1.01	Yes	shop316.18	2204	2204	4.23	Yes
shop100.7	2665	2665	0.09	Yes	shop316.19	2811	2811	0.44	Yes
shop100.8	2621	2621	0.11	Yes	shop1000.20	2041	2190	719.28	
shop100.9	2276	2276	7.46	Yes	shop1000.21	2709	2709	1.83	Yes
shop316.10	2311	2311	119.55	Yes	shop1000.22	2057	2184	730.85	
shop316.11	2907	2907	0.42	Yes	shop3162.30	2407	2407	127.45	Yes
smat100.0	70,175	70,175	24.42	Yes	smat316.12	34,266	34,266	152.54	Yes
smat100.1	69,636	69,636	20.17	Yes	smat316.13	27,251	27,251	130.37	Yes
smat100.2	59,186	59,186	28.64	Yes	smat316.14	23,203	23,203	119.50	Yes
smat100.3	73,104	73,104	24.16	Yes	smat316.15	24,252	24,252	152.07	Yes
smat100.4	58,962	58,962	18.51	Yes	smat316.16	28,690	28,690	180.38	Yes
smat100.5	89,297	89,297	30.22	Yes	smat316.17	27,313	27,313	157.42	Yes
smat100.6	57,248	61,904	18.74		smat316.18	29,449	29,449	144.71	Yes
smat100.7	67,624	67,624	28.50	Yes	smat316.19	25,414	25,414	117.88	Yes
smat100.8	66,698	66,698	22.76	Yes	smat1000.20	10,371	10,371	826.19	Yes
smat100.9	71,811	71,811	25.26	Yes	smat1000.21	9,360	9,360	636.71	Yes
smat316.10	21,672	21,672	147.15	Yes	smat1000.22	8,817	8,817	572.56	Yes
smat316.11	23,998	23,998	147.35	Yes	smat3162.30	2,959	49,035	6963.29	

costs, and appear structurally quite difficult for the Lin–Kernighan algorithm.

We present in detail the results for all 86 problems in Tables 4 and 5 for $p=5$ and $q=5$. We also identify the lower bounds that found a tight optimal solution, as well as the best lower bound for each problem (ties are broken by shortest running time). The columns "Avg. bin. steps" and "Avg. # of LK calls" give the average number of binary search steps and calls to Concorde's Lin–Kernighan algorithm, respectively. The results are generally quite good and computing times are reasonable.

6. The asymmetric maximum scatter traveling salesman problem

The maxmin version of the BTSP is called the maximum scatter traveling salesman problem (MSTSP). In Section 1 we discussed this problem and observed that the MSTSP on cost matrix $C=(c_{ij})_{n\times n}$ can be reduced to a BTSP on cost matrix $\tilde{C}=(\tilde{c}_{ij})_{n\times n}$ using the transformation $\tilde{c}_{ij}=M-c_{ij}$ where $M=\max\{c_{ij}:(i,j)\in E\}$. Although this transformation preserves optimality, it does not preserve theoretical approximation ratios. In this section we explore the impact of this transformation on heuristics using experimental analysis.

As in our presentation of results for the BTSP, Table 6 summarizes the relative strengths of the lower bound algorithms applied to cost matrix $\tilde{C}$ (which, in the MSTSP sense, are upper bounds to the optimal objective value on C) on the same problem test set used in the computational results in Section 5. These results are not unlike those we observed for the asymmetric BTSP, with the BAP, BBSSP($\hat{C}$), and BSCSSP bounds generally providing cheap but tight upper bounds to the MSTSP.

Tables 7 and 8 report results of Algorithm 3 for $p=5$ and $q=5$ on each problem over 10 trials. For 63 of the 86 problems, our algorithm had little trouble consistently finding an optimal tour. For the remaining 23 problems, our algorithm found a tour

Table 11
Heuristic results on tilted drilling machine instances, sup norm (`stilt`), approx. shortest common superstring instances (`super`), shortest-path closure of `amat` (`tmat`), and shortest-path closure of `smat` (`tsmat`).

Problem	Lower bound	Best obj.	Avg. time (s)	Opt.?	Problem	Lower bound	Best obj.	Avg. time (s)	Opt.?
stilt100.0	382,208	404,808	31.09		stilt316.12	291,320	432,260	386.30	
stilt100.1	491,416	491,416	3.34	Yes	stilt316.13	179,462	365,846	302.48	
stilt100.2	377,720	392,728	35.26		stilt316.14	198,152	430,308	281.13	
stilt100.3	401,976	410,408	35.13		stilt316.15	232,104	379,108	298.41	
stilt100.4	347,440	389,296	45.20		stilt316.16	290,738	370,896	372.58	
stilt100.5	456,788	456,788	1.64	Yes	stilt316.17	196,896	391,892	247.78	
stilt100.6	417,056	417,056	1.10	Yes	stilt316.18	244,688	418,088	355.59	
stilt100.7	494,360	494,360	2.12	Yes	stilt316.19	212,268	367,686	287.51	
stilt100.8	522,748	522,748	0.49	Yes	stilt1000.20	121,812	466,364	986.35	
stilt100.9	321,884	383,838	38.75		stilt1000.21	117,542	460,732	1295.38	
stilt316.10	226,504	401,968	336.37		stilt1000.22	127,000	456,270	1236.09	
stilt316.11	235,910	424,144	268.78		stilt3162.30	66,552	569,890	4127.88	
super100.0	10	10	0.09	Yes	super316.12	9	9	0.48	Yes
super100.1	11	11	0.09	Yes	super316.13	9	9	0.48	Yes
super100.2	10	10	0.09	Yes	super316.14	9	9	0.48	Yes
super100.3	10	10	0.09	Yes	super316.15	9	9	0.52	Yes
super100.4	10	10	0.09	Yes	super316.16	9	9	0.48	Yes
super100.5	10	10	0.09	Yes	super316.17	9	9	0.47	Yes
super100.6	10	10	0.09	Yes	super316.18	9	9	0.49	Yes
super100.7	10	10	0.09	Yes	super316.19	9	9	0.49	Yes
super100.8	10	10	0.09	Yes	super1000.20	8	8	12.07	Yes
super100.9	10	10	0.09	Yes	super1000.21	8	8	12.07	Yes
super316.10	9	9	0.49	Yes	super1000.22	8	8	10.63	Yes
super316.11	9	9	0.47	Yes	super3162.30	7	9	1905.87	
tmat100.0	50,981	50,981	0.11	Yes	tmat316.12	26,551	26,551	0.48	Yes
tmat100.1	50,821	50,821	0.12	Yes	tmat316.13	20,090	20,090	0.62	Yes
tmat100.2	63,674	63,674	0.11	Yes	tmat316.14	17,539	17,539	0.70	Yes
tmat100.3	62,994	62,994	0.13	Yes	tmat316.15	19,952	19,952	0.59	Yes
tmat100.4	55,534	55,534	0.15	Yes	tmat316.16	20,779	20,779	0.61	Yes
tmat100.5	35,793	35,793	0.12	Yes	tmat316.17	26,165	26,165	0.56	Yes
tmat100.6	72,650	72,650	0.22	Yes	tmat316.18	17,207	17,207	0.62	Yes
tmat100.7	59,291	59,291	0.11	Yes	tmat316.19	43,608	43,608	0.48	Yes
tmat100.8	55,462	55,462	0.14	Yes	tmat1000.20	9978	9978	2.02	Yes
tmat100.9	49,928	49,928	0.17	Yes	tmat1000.21	10,624	10,624	1.84	Yes
tmat316.10	21,896	21,896	0.58	Yes	tmat1000.22	6715	6715	2.11	Yes
tmat316.11	18,240	18,240	0.59	Yes	tmat3162.30	2985	2985	12.95	Yes
tsmat100.0	44,258	44,258	44.31	Yes	tsmat316.12	21,337	21,337	276.18	Yes
tsmat100.1	42,415	42,415	43.65	Yes	tsmat316.13	24,463	24,463	206.42	Yes
tsmat100.2	37,786	37,786	34.48	Yes	tsmat316.14	21,042	21,042	240.87	Yes
tsmat100.3	40,608	40,608	36.24	Yes	tsmat316.15	21,767	21,767	208.61	Yes
tsmat100.4	48,184	48,184	40.81	Yes	tsmat316.16	28,690	28,690	225.90	Yes
tsmat100.5	54,108	54,108	42.30	Yes	tsmat316.17	27,099	27,099	247.76	Yes
tsmat100.6	54,157	54,486	46.66		tsmat316.18	24,829	24,829	178.47	Yes
tsmat100.7	45,189	45,189	41.46	Yes	tsmat316.19	15,023	15,023	263.81	Yes
tsmat100.8	64,065	64,065	29.83	Yes	tsmat1000.20	8104	8104	291.85	Yes
tsmat100.9	61,244	61,244	43.66	Yes	tsmat1000.21	6823	6823	886.82	Yes
tsmat316.10	20,537	20,537	239.36	Yes	tsmat1000.22	7578	7578	658.42	Yes
tsmat316.11	22,643	22,643	287.11	Yes	tsmat3162.30	2544	2544	1885.18	Yes

generally within 3% of optimality. As in previous cases, if our algorithm did not find a tour whose largest cost is equal to the best upper bound obtained, we confirm optimality using the exact version of Algorithm 3 where Concorde's exact TSP solver is selected as solver α.

7. Conclusions and discussion

We developed algorithms for solving the asymmetric versions of BTSP and MSTSP and studied their efficacy using experimental and theoretical analysis. An $\lceil n/2 \rceil -$approximation algorithm for the asymmetric BTSP is given for instances where edge costs satisfy the triangle inequality and this is the first heuristic for the problem with bounded performance ratio. Several lower bound algorithms for BTSP are given and analyzed using theoretical and experimental comparisons. We also give effective heuristics to solve BTSP and MSTSP in practice. Results of systematic experiments are reported on a test bed consisting of benchmark problems.

It would be interesting to explore $\epsilon -$approximation algorithms for the asymmetric BTSP for constant ϵ on instances where the edge costs satisfy the triangle inequality. This question is relevant since the symmetric version has a 2-approximation algorithm when the edge costs satisfy the triangle inequality.

Appendix

Tables 9–11 present summarized results for the additional 288 benchmark instances. All results are the average of 10 trials with each problem and parameters settings are as discussed in the paper. Times reported are in CPU seconds (Table 12).

Table 12
Global statistics of experimental results on BTSP. The column "% optimal" shows that the percentage of problems in the class where optimality is guaranteed.

Problem prefix	Problem size			CPU Time			Number of problems	% Optimal
	Min	Max	Average	Min	Max	Average		
amat	100	430	3162	0.13	5760.72	243	24	95.8
balas	84	200	134	0.7	0.39	0.17	4	100
br	17	17	17	0.01	0.01	0.01	1	100
coin	100	430	3162	0.16	4075.57	247.15	24	83.3
crane	100	430	3162	0.12	6234.88	327.54	24	79.1
disk	100	430	3162	0.10	5957.56	386.67	24	87.5
ft	53	70	61	0.01	0.04	0.03	2	100
ftv	33	181	97	0.02	53.34	13.42	18	100
kro	124	124	124	0.24	0.24	0.24	1	100
p	43	43	43	0.02	0.02	0.02	1	100
ran	500	1000	750	1.23	9.46	4.94	10	100
rect	100	430	3162	12.4	5241.8	347.11	24	91.6
rgb	323	443	381	75.06	138.43	104.71	4	25
rtilt	100	430	3162	0.33	3980.61	483.61	24	12.5
ry	48	48	48	5.17	5.17	5.17	1	100
shop	100	430	3162	0.09	730.85	72.25	24	91.6
smat	100	430	3162	18.74	6963.29	445.39	24	91.6
stilt	100	430	3162	0.49	4127.88	457.36	24	20.8
super	100	430	3162	0.09	1905.87	81.09	24	95.8
tmat	100	430	3162	0.11	12.95	1.08	24	100
tsmat	100	430	3162	29.83	1885.18	27.84	24	95.8
uk	66	66	66	0.05	0.05	0.05	1	100

References

[1] Ahmed ZH. A lexisearch algorithm for the bottleneck travelling salesman problem. International Journal of Computer Science and Security 2010;3:569–77.
[2] Ahmed ZH. A data-guided lexisearch algorithm for the asymmetric traveling salesman problem. Mathematical Problems in Engineering, doi: http://dx.doi.org/10.1155/2011/750968.
[3] Ahuja RK, Magnanti TL, Orlin JB. Network flows: theory, algorithms, and applications. Prentice Hall; 1993.
[4] Applegate DL, Bixby RE, Chvátal V, Cook WJ. Concorde TSP solver; May 2005. ⟨http://www.tsp.gatech.edu/concorde⟩.
[5] Arkin EM, Chiang Y, Mitchell JSB, Skiena SS, Yang T. On the maximum scatter traveling salesman problem. SIAM Journal on Computing 1999;29 (2):515–44.
[6] Arora S. Approximation algorithms for geometric TSP. In: Gutin G, Punnen AP, editors. The traveling salesman problem and its variations. Combinatorial optimization. Secaucus, NJ, USA: Kluwer Academic Publishers; 2002. p. 207–22 [Chapter 5].
[7] Balas E. Personal communication by J. LaRusic; August 2008.
[9] Bondy J, Thomassen C. A short proof of Meyniel's theorem. Discrete Mathematics 1977;19:195–7.
[10] Burkard R, Dell'Amico M, Martello S. Assignment problems. Philadelphia: SIAM; 2008.
[11] Carpaneto G, Martello S, Toth P. An algorithm for the bottleneck traveling salesman problem. Operations Research 1984;32(2):380–9.
[12] Christofides N. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical Report 388, CMU; 1976.
[13] Cirasella J, Johnson DS, McGeoch LA, Zhang W. The asymmetric traveling salesman problem: algorithms, instance generators, and tests. Lecture notes in computer science, vol. 2153. Berlin, Heidelberg: Springer; 2001. p. 32–59.
[14] Doroshko NN, Sarvanov VI, Toth P. A minimax traveling salesman problem and Hamiltonian cycles in powers of graphs. Vestsi Akademii Navuk BSSR, Seriya Fizika-Matematichnykh Navuk 1981;6 [Russian].
[15] Edmonds J, Fulkerson F. Bottleneck extrema. Journal of Combinatorial Theory 1970;8:435–48.
[16] Fischetti M, Lodi A, Toth P. Exact methods for the asymmetric traveling salesman problem. In: Gutin G, Punnen AP, editors. The traveling salesman problem and its variations. Combinatorial optimization. Secaucus, NJ, USA: Kluwer Academic Publishers; 2002. p. 169–205 [Chapter 4].
[17] Frieze AM, Galbiati G, Maffioli F. On the worst-case performance of some algorithms for the asymmetric traveling salesman problem. Networks 1982;12 (1):23–39.
[18] Garfinkel RS, Gilbert KC. The bottleneck traveling salesman problem: algorithms and probabilistic analysis. Journal of the Association for Computing Machinery 1978;25(3):435–48.
[19] Ghouilà-Houri A. Une condition suffisante d'existence d'un circuit hamiltonien. Comptes Rendus de l'Académie des Sciences Paris 1960;25:495–7.
[20] Gutin G, Punnen AP, editors. The traveling salesman problem and its variations. Combinatorial optimization. Secaucus, NJ, USA: Kluwer Academic Publishers; 2002.
[21] Helsgaun K. An effective implementation of the Lin–Kernighan traveling salesman heuristic. European Journal of Operational Research 2000;126: 106–30.
[22] Hochbaum DS, Shmoys DB. A unified approach to approximation algorithms for bottleneck problems. Journal of the Association for Computing Machinery 1986;33:533–50.
[25] Jonker R, Volgenant T. Transforming asymmetric into symmetric traveling salesman problems. Operations Research Letters 1983;2:161–3.
[26] Kabadi S, Punnen AP. The Bottleneck TSP. In: Gutin G, Punnen AP, editors. The traveling salesman problem and its variations. Combinatorial optimization. Secaucus, NJ, USA: Kluwer Academic Publishers; 2002. p. 489–584 [Chapter 15].
[27] Kaplan H, Lewenstein M, Shafrir N, Sviridenko M. Approximation algorithms for asymmetric TSP by decomposing directed regular multigraphs. In: Proceedings of the 44th Annual Symposium on the Foundation of Computer Science; 2003. p. 56–7.
[28] Kleinberg J, Williamson D. A new $O(\log n)$-approximation algorithm for ATSP. In: Lecture Notes on *Approximation Algorithms*; Fall 1998.
[29] LaRusic J, Aubanel E, Punnen AP. Experimental analysis of heuristics for the bottleneck traveling salesman problem. Journal of heuristics 2012;18: 473–503.
[30] Lin S, Kernighan BW. An effective heuristic algorithm for the traveling salesman problem. Operations Research 1973;21:972–89.
[31] Manku GS. A linear time algorithm for the bottleneck biconnected spanning subgraph problem. Information Processing Letters 1996;59:1–7.
[32] Marczyk A. On hamiltonian powers of digraphs. Graphs and Combinatorics 2000;16:103–13.
[33] Parker RG, Rardin RL. Guaranteed performance heuristics for the bottleneck traveling salesman problem. Operations Research Letters 1982;12: 269–72.
[35] Punnen AP. Minmax strongly connected subgraphs with node penalties. Journal of Applied Mathematics and Decision Sciences 2005;2005(2): 107–11.
[36] Punnen AP. A fast algorithm for a class of bottleneck problems. Computing 1996;56:397–401.
[37] Punnen AP, Aneja YP. Minmax combinatorial optimization. European Journal of Operational Research 1995;81:634–43.
[38] Punnen AP, Nair KPK. A fast and simple algorithm for the bottleneck biconnected spanning subgraph problem. Information Processing Letters 1994;50:283–6.
[39] Punnen AP, Nair KPK. Improved complexity bound for the maximum cardinality bottleneck bipartite matching problem. Discrete Applied Mathematics 1994;55(1):91–3.

[40] Ramakrishnan R, Sharma P, Punnen AP. New heuristics for the bottleneck TSP. Opsearch 2009;46:275–88.

[42] Reinelt G. TSPLIB; August 2008. 〈 http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/〉.

[43] Sarvanov VI. A minimax traveling salesman problem on a plane: complexity of an approximate solution. Doklady Natsionalnoi Akademii Nauk Belarusi 1995;39(6):16–9 [Russian].

[44] Schaar G, Wojda AP. An upper bound for the Hamiltonicity exponent of finite digraphs (Note). Discrete Mathematics 1997;164:313–6.

[45] Sergeev SI. Algorithms for the minimax problem of the traveling salesman. I. An approach based on dynamic programming. Automation and Remote Control 1995;56(7):1027–32.

[47] Timofeev EA. Minimax 2-connected spanning subgraph and the bottleneck travelling salesman problem. Cybernetics 1980;15:516–21.

[48] Vassilevska V, Williams R, Yuster R. All-pairs bottleneck paths for general graphs in truly sub-cubic time. In: *Proceedings of the thirty-ninth annual ACM symposium on theory of computing*; 2007. p. 585–9.