Chapter 15

THE BOTTLENECK TSP

Santosh N. Kabadi
Faculty of Administration
University of New Brunswick-Frederiction
New Brunswick, Canada
kabadi@unb.ca

Abraham P. Punnen
Department of Mathematical Sciences
University of New Brunswick-Saint John
New Brunswick, Canada
punnen@unbsj.ca

1. Introduction

In this chapter we study the *bottleneck traveling salesman problem* (BTSP) introduced in Chapter 1. BTSP is a variation of the classical traveling salesman problem (TSP) that differs from the TSP only in the objective function. Let us first restate the problem.

Let $G = (N, E)$ be a (directed or undirected) graph on node set $N = \{1, 2, \ldots, n\}$ and let $\mathbb{F}$ be the family of all Hamiltonian cycles (tours) in G. For each edge $e \in E$, a cost c_e is prescribed. For any $\mathcal{H} \in \mathbb{F}$, let $c_{\max}(\mathcal{H}) = \max\{c_e : e \in \mathcal{H}\}$. Then the BTSP is to find a Hamiltonian cycle $\mathcal{H} \in \mathbb{F}$ such that $c_{\max}(\mathcal{H})$ is as small as possible.

Without loss of generality we replace G by K_n in the undirected case and $\overleftrightarrow{K}_n$ in the directed case by adding the missing edges with cost ∞. Thus, the edge costs will be given in the form of an $n \times n$ matrix C, called the cost matrix, where any non-diagonal entry c_{ij} corresponds to the cost c_e of the edge $e = (i, j)$. As indicated in Chapter 1and Chapter 11, we can also represent a tour in $\overleftrightarrow{K}_n$ as a cyclic permutation γ on $N = \{1, 2, \ldots, n\}$. Let $c_{\max}(\gamma) = \max\{c_{i,\gamma(i)} : i \in N\}$. Then the BTSP is to find a tour γ^* on N such that $c_{\max}(\gamma^*) = \min\{c_{\max}(\gamma) : \gamma$ is

G. Gutin et al. (eds.), *The Traveling Salesman Problem and Its Variations*

a tour on N}. When the underlying cost matrix needs to be emphasized, we sometimes denote the BTSP with cost matrix C as BTSP(C). The BTSP can also be formulated as an integer programming problem:

$$\begin{aligned}
\text{Minimize} \quad & \max\{c_{ij}x_{ij},\ 1 \le i,j \le n,\ i \ne j\} && (1)\\
\text{Subject to} \quad & && (2)\\
\sum_{i=1}^{n} x_{ij} &= 1, j \in N && (3)\\
\sum_{j=1}^{n} x_{ij} &= 1, i \in N && (4)\\
x_{ij} &= 0 \text{ or } 1 && (5)\\
\sum_{i\in S}\sum_{j\in \overline{S}} x_{ij} &\ge 1\ \forall S \subset N, && (6)
\end{aligned}$$

where $\overline{S} = N \setminus S$.

In fact each of the (mixed) integer programming formulations of TSP discussed in Chapter 1 leads to a corresponding (mixed) integer programming formulation of the BTSP.

To the best of our knowledge, the bottleneck TSP was introduced by Gilmore and Gomory [360] assuming special structure of elements of C. Garfinkel and Gilbert [349] considered the general BTSP model and discussed an application of the problem in the context of machine scheduling. Meaningful interpretations of the BTSP model and its variations can be given in the context of some route planning problems and transportation of goods perishable in time.

Another problem closely related to the BTSP is the *maximum scatter traveling salesman problem* (MSTSP) [33]. Here, each edge e in G is assigned a weight w_e and we seek a tour $\mathcal{H}$ in G such that the smallest edge weight in $\mathcal{H}$ is as large as possible. Let $w_{\min}(\mathcal{H}) = \min\{w_e : e \in \mathcal{H}\}$. Then, the MSTSP is to find a Hamiltonian cycle $\mathcal{H} \in \mathbb{F}$ such that $w_{\min}(\mathcal{H})$ is as large as possible.

Applications of MSTSP and its variations include medical image processing [666], obtaining revetting sequence in joining metals in the aircraft industry [745, 746] etc.

The problems MSTSP and BTSP are equivalent in the sense that an optimal solution to the MSTSP with weight matrix W can be obtained by solving the BTSP with cost matrix $C = -W$ and vice versa. If we require the edge costs/weights to be positive, a large enough constant could be added to each of the edge costs/weights. However, this trans-

formation may not be useful for some approximation algorithms that use special structure of C or W, since the addition of a constant and/or multiplication of edge costs/weights by -1 may violate key properties of the matrix that are used by these algorithms. Thus in such cases, the characteristics of the two problems are different and warrant separate treatment.

When the cost matrix C is symmetric, (equivalently the underlying graph is symmetric), we refer to the BTSP as *symmetric bottleneck traveling salesman problem* (SBTSP); otherwise it is referred to as *asymmetric bottleneck traveling salesman problem* (ABTSP). A special case of SBTSP, where the vertices of G correspond to points in the Euclidean plane and edge costs are Euclidean distances, is referred to as *Euclidean bottleneck traveling salesman problem*(EBTSP). Similarly we have symmetric, asymmetric, and Euclidean versions of the MSTSP.

It is well known that the Hamiltonian cycle problem on a grid graph is NP-complete [466]. As an immediate consequence we have that EBTSP is NP-hard and hence BTSP is NP-hard [466]. In fact BTSP is NP-hard even if we restrict the graph G to a grid graph or a planar bipartite graph in which degree of each node is 3 or less. As in the case of TSP, this follows immediately by a reduction from the Hamiltonian cycle problem on these graphs which is known to be NP-complete [466] (see Appendix B). Similarly, MSTSP can be shown to be NP-hard on grid graphs and planar bipartite graphs in which degree of each node is 3 or less. Fekete [284] recently proved that MSTSP under Euclidean distances in $\mathbb{R}^d$ is NP-hard for any fixed $d \geq 3$. More complexity results are discussed is Section 3.

2. Exact Algorithms

Recall that an exact algorithm for an optimization problem is guaranteed to produce an optimal solution for any instance of the problem or declare that a feasible solution does not exist. Since BTSP and MSTSP are NP-hard, such algorithms are generally of implicit enumeration type. Exact algorithms for MSTSP have not been discussed explicitly in literature. However, the transformation discussed in Section 1 can be used to solve MSTSP as BTSP.

2.1. BTSP as a TSP

In the preceding chapters, we have seen several interesting properties of and solution approaches for the TSP. Let us now examine how the TSP is related to the BTSP. We will first show that BTSP can be formulated

as a TSP in the sense that an optimal solution to this TSP gives an optimal solution to the BTSP.

Note that in solving BTSP to optimality, the numerical values of the edge costs are unimportant; only the ordering of edge costs matters. Let the edges of G be labeled as $\{e_1, e_2, \ldots, e_m\}$ such that $c_{e_1} \leq c_{e_2} \leq \cdots \leq c_{e_m}$. Let $\{d_e : e \in E\}$ be another set of costs for the edges of G. Let us denote by BTSP(C) and BTSP(D) the instances of BTSP with edge costs c_e's and d_e's respectively. Let L and U respectively be known lower and upper bounds on the optimal objective function value of BTSP(C).

Lemma 1 *If $d_{e_1} \leq d_{e_2} \leq \cdots \leq d_{e_m}$ with $d_{e_i} < d_{e_{i+1}}$ whenever $c_{e_i} < c_{e_{i+1}}$ for all i such that $L \leq c_{e_i} \leq U$, then every optimal solution to BTSP(D) is also an optimal solution to BTSP(C).*

The proof of the above lemma is straightforward.

Let $\alpha_1 < \alpha_2 < \cdots < \alpha_t$ be an ascending arrangement of distinct costs c_e of edges of G such that $L \leq c_e \leq U$. Define $F_r = \{\mathcal{H} \in \mathbb{F} : c_{\max}(\mathcal{H}) = \max\{c_e : e \in \mathcal{H}\} = \alpha_r\}$ for $r = 1, \ldots, t$, $F_{t+1} = \{\mathcal{H} \in \mathbb{F} : c_{\max}(\mathcal{H}) > \alpha_t\}$, and $U_r = \cup_{i=1}^{r} F_i$, $r = 1, \ldots, t+1$. Consider new edge weights $\{d_e : e \in E\}$ satisfying,

$$\min\{\sum_{e \in \mathcal{H}} d_e : \mathcal{H} \in F_r\} > \min\{\sum_{e \in \mathcal{H}} d_e : \mathcal{H} \in U_{r-1}\} \tag{7}$$

for all $2 \leq r \leq (t+1)$. Here, minimum over empty set is taken as $-\infty$. Let TSP(D) denote the TSP with edge weights d_e for $e \in E$.

Theorem 2 *Every optimal solution to TSP(D) is also an optimal solution to BTSP(C).*

Proof. Clearly, if k is the smallest index such that F_k is non-empty then any $\mathcal{H} \in F_k$ is an optimal solution to BTSP(C) with the optimal objective function value α_k. If $\mathcal{H}'$ is an optimal solution to TSP(D) and $\mathcal{H}' \in F_p$, then,

$$\sum_{e \in \mathcal{H}'} d_e = \min\{\sum_{e \in \mathcal{H}} d_e : \mathcal{H} \in F_p\} > \min\{\sum_{e \in \mathcal{H}} d_e : \mathcal{H} \in U_{p-1}\}.$$

Thus, U_{p-1} must be empty and hence $\mathcal{H}'$ is optimal to BTSP(C). ■

Corollary 3 *Let $b_1 = 0$ and for $j = 2, \ldots, t+1$, let $b_j = nb_{j-1} + 1$. Let the edge costs d_e's be defined as :*

$$d_e = \begin{cases} 0 & \text{if } c_e \leq \alpha_1 \\ b_j & \text{if } c_e = \alpha_j,\ j = 2, \ldots, t \\ b_{t+1} & \text{if } c_e > \alpha_t \end{cases}$$

Then every optimal solution to TSP(D) is also an optimal solution to BTSP(C).

Corollary 4 *Let p be the largest index such that $c_{e_p} \leq U$ and $m = |E|$. Define the edge costs $d_{e_j} = 2^{j-1}$ for $j = 1, 2, \ldots, p$ and $d_{e_j} = 2^{p+1}$ for $j = p+1, p+2, \ldots, m$. Then every optimal solution TSP(D) is also an optimal solution to BTSP(C).*

Let k_i be the number of edges e with $c_e = \alpha_i$, $i = 1, \ldots, t$.

Corollary 5 *Let $b_1 = 0$. For $j = 2, \ldots, t+1$, let u_j be the smallest positive integer such that $\sum_{i=u_j}^{j-1} k_i < n$. Define $b_j = \sum_{i=u_j}^{j-1} k_i b_i + (n - \sum_{i=u_j}^{j-1} k_i) b_{u_j - 1} + 1$. Let the edge costs d_e's be defined as :*

$$d_e = \begin{cases} 0 & \text{if } c_e \leq \alpha_1 \\ b_j & \text{if } c_e = \alpha_j,\ j = 2, \ldots, t \\ b_{t+1} & \text{if } c_e > \alpha_t \end{cases}$$

Then every optimal solution to TSP(D) is also an optimal solution to BTSP(C).

The proofs of corollaries 3, 4 and 5 follow from the fact that the special edge costs defined satisfy the condition (7).

Results analogous to Theorem 2 in the context of various bottleneck problems including the BTSP are well known [146, 410, 472, 691]. Although, Theorem 2 shows that BTSP can be solved as a TSP, it is not of much practical value since the costs d_e used in TSP(D) grow exponentially. However, when the number of distinct edge costs between given lower and upper bounds on the optimal objective function value is relatively small (say ≤ 10), then the edge costs defined in corollaries 3 and 5 are sometimes useful. This is exploited in the generalized threshold algorithm given in the next section.

A result similar to Theorem 2 can be obtained for the case of MSTSP showing that MSTSP can be solved as a MAX TSP or as a TSP.

2.2. Generalized Threshold Algorithm

The generalized threshold algorithm [691] is a modification of the well known threshold algorithm for solving bottleneck problems [269]. It solves BTSP as a sequence of TSP's with relatively smaller edge costs utilizing Corollary 3. Without loss of generality, we assume the edge costs are positive. Thus $L > 0$. Let $S_1, S_2, \ldots, S_r$ be an ordered partition of the index set $\{1, 2, \ldots, t\}$, (that is, $p \in S_i$, $q \in S_j$, $i < j$ implies $\alpha_p <$

α_q). We say that an edge e of G *corresponds to* S_i if $c_e = \alpha_p$ for some $p \in S_i$. For each edge e that corresponds to S_i, define $c'_e = i$, $1 \leq i \leq r$. Define $c'_e = 0$ if $c_e < L$ and $c'_e = r + 1$ if $c_e > U$. Let BTSP(C') represent the BTSP with edge costs $\{c'_e : e \in E\}$. Let $\mathcal{H}'$ be an optimal solution to BTSP(C') with optimal objective function value k. (Note that $k \in \{1, \ldots, r\}$.) Let $\mathcal{H}^*$ be an optimal solution to BTSP.

Theorem 6 *[691]* $\min_{i \in S_k}\{\alpha_i\} \leq c_{\max}(H^*) \leq \max_{i \in S_k}\{\alpha_i\}$.

Note that BTSP(C') is an approximation to the problem BTSP(C). If r is small (say ≤ 10), then BTSP(C') could be solved as a TSP with edge costs of moderate size using Corollary 3 or 5. Further, the solution to BTSP(C') provides new lower and/or upper bounds for the BTSP as guaranteed by Theorem 6. Using these new bounds, a 'better' approximation to BTSP can be constructed. Continuing this process, we eventually get an optimal solution to the BTSP. Our generalized threshold algorithm is precisely this. A formal description of the algorithm is given below.

The Generalized Threshold Algorithm

Step 1: Construct a lower bound L and an upper bound U for the optimal objective function value. Set $q = 1$, (q is the iteration counter).

Step 2: Let $\alpha_1 < \alpha_2 < \cdots < \alpha_{t_q}$ be an ascending arrangement of distinct edge costs c_e such that $L \leq c_e \leq U$.

Step 3: Construct the ordered partition $S_1, S_2, \ldots, S_{r(q)}$ of $\{1, 2, \ldots, t_q\}$. Let every edge e with cost $c_e < L$ correspond to S_0 and every edge e with $c_e > U$ correspond to $S_{r(q)+1}$. Let $c'_e = i$ for edges e that corresponds to S_i, $0 \leq i \leq r(q) + 1$.

Step 4: Solve the BTSP with costs c'_e. (This is done by solving an equivalent TSP indicated in Theorem 2.) Let $\mathcal{H}'$ be the optimal solution produced and let the optimal objective function value be k. (Note that $k \in \{1, \ldots, r(q)\}$.)

Step 5: If $|S_k| = 1$, then output $\mathcal{H}'$ and stop. Else update the lower and upper bounds
$L = \min_{i \in S_k}\{\alpha_i\}$ and
$U = \max_{e \in \mathcal{H}'}\{c_e\}$.
Set $q = q + 1$ and go to Step 2.

The validity of the generalized threshold algorithm follows from the preceding discussions. Note that the number of partitions, r (indicated as $r(q)$ in the algorithm), is a function of the iteration counter and can

be changed from iteration to iteration. For a large value of r, the TSP in Step 4 will have very large edge costs, leading to overflow errors. A reasonable choice of r is a number less than or equal to 10. The complexity of the algorithm depends on the number of iterations, which in turn depends on the way the partitions $S_1, S_2, \ldots, S_r$ are constructed. If $S_1, S_2, \ldots, S_{r-1}$ are selected as singletons and S_r contains all the remaining edges with cost c_e between L and U, the algorithm could take $O(n^2/r)$ iterations in worst case, since $t = O(n^2)$. We call this the *incremental search version* of the generalized threshold algorithm. If $|S_i|$ is approximately equal to $|S_j|$ for all $1 \leq i, j \leq r$, the number of iterations of the generalized threshold algorithm is $O(\log_r n)$. We call this *the r-section version* of the generalized threshold algorithm. If the lower bound L is expected to be tight, the incremental search version may work well in practice and often terminates in one iteration, although its worst case complexity is too high. If $r = 2$ (which is desirable for very large n), Step 4 of the generalized threshold algorithm can be implemented as testing hamiltonicity of an appropriate spanning subgraph of G. This special case of the generalized threshold algorithm is precisely the adaptation of the well known *threshold algorithm* to the BTSP.

The generalized threshold algorithm could take advantage of existing powerful TSP codes to get a reasonably fast algorithm for the BTSP without much programming effort. It is easy to develop a corresponding generalized threshold algorithm for the MSTSP which solves the problem as a sequence of MAX TSP's. We could also use the generalized threshold algorithm for BTSP to solve MSTSP using the transformation discussed in the Section 1.

2.3. Branch and Bound Algorithms

Branch and bound algorithms are classical approaches for solving 'hard' combinatorial optimization problems. The power of a branch and bound algorithm depends on the ability to generate good lower and upper bounds on the optimal objective function value and establishing an efficient branching strategy to generate the search tree. Garfinkel and Gilbert [349], Carpaneto et al [165], and Sergeev and Chernyshenko [757] developed specialized branch and bound algorithms to solve BTSP. Computational results based on problems of size less than or equal 200 are reported in [165, 349]. There is no recent experimental study published on the branch and bound algorithms for BTSP. The branching strategies used by Carpaneto et al [165] and Garfinkel and Gilbert [349] are similar to those studied for the case of TSP and will not be discussed here. For details we refer to the original papers. (See also Chapter 4.)

2.3.1 Lower Bounds. We now discuss some good lower bounds for the BTSP that can be obtained efficiently.

2-max Bound: This lower bound is very simple and easy to compute. It is valid for the symmetric version of BTSP only. For each node i of G, let $\Delta(i)$ be the set of edges incident on i and μ_i be the second smallest cost (counting multiplicity) of edges in $\Delta(i)$. Then $\max_{i \in N}\{\mu_i\}$ is a lower bound on the optimal objective function value of the BTSP. In a graph with m edges, this bound can be identified in $O(m)$ time. An asymmetric version of 2-max bound is introduced by Carpaneto et al [165].

Biconnected Spanning Subgraph Bound: We denote this lower bound as *BSS bound* and it is defined for the symmetric version of BTSP. Since every tour is biconnected, the optimum objective function value of a Bottleneck biconnected spanning subgraph problem (BBSSP) on the graph G is a lower bound for the BTSP. Several algorithms are available to solve the BBSSP. For example, in a graph with m edges and n nodes, Manku [578] proposed an $O(m)$ algorithm, Punnen and Nair [685] an $O(m + n \log n)$ algorithm, Timofeev [793] an $O(n^2)$ algorithm, and Parker and Rardin [661] an $O(n^2 \log n)$ algorithm to solve BBSSP.

Strongly Connected Spanning Subgraph Bound: We denote this bound as SCSS bound and is used for the asymmetric version of the BTSP. Since every directed Hamiltonian cycle is strongly connected, the optimal objective function value of a bottleneck strongly connected spanning subgraph problem (BSSSP) on the digraph G is a lower bound for the ABTSP on G. In a digraph with m arcs, BSSSP can be solved in $O(m)$ time [677].

Assignment Bound: The assignment problem is used to compute lower bounds for the traveling salesman problem. In the same way, the bottleneck assignment problem (BAP) can be used to compute a lower bound for the BTSP. Carpaneto et al [165] used the assignment bound, among other lower bounds, in their branch and bound algorithm for BTSP. They also provided a heuristic search scheme to find alternate optimal solutions for the BAP that correspond to cyclic permutations (tours). If the heuristic is successful in getting such a tour, it is indeed an optimal tour. BAP can be solved in $O(n^{2.5})$ time using an algorithm of Punnen and Nair [686]. For the case of sparse cost matrix (several edge costs are very large) an algorithm due to Gabow and Tarjan [343] runs faster. Other algorithms for BAP include Derigs and Zimmerman [253],

and Carpaneto and Toth [167].

2.4. Branch and Cut Algorithms

Branch and cut algorithms are the state-of-the-art exact algorithms for the TSP (see Chapters 4and 2) that could solve reasonably large TSPs to optimality. A branch and cut algorithm for the TSP is based on a partial linear programming representation of the TSP. The success of the algorithm depends on the ability to generate facets or high dimensional faces of the TSP polytope that are violated by a 'current' infeasible solution. The BTSP can be formulated as a bottleneck linear programming problem (BLP) [410]

$$\begin{aligned} &\text{Minimize} \quad \max\{c_e : x_e > 0\} \\ &\text{Subject to } X \in T_n, \end{aligned}$$

where T_n is the TSP polytope (symmetric or asymmetric depending on whether BTSP is symmetric or asymmetric) and entries of $X = (x_1, x_2, \ldots, x_m)$ correspond to edges of G. Since the objective function of the BLP is concave and quasi-convex, it can be shown that there exists an optimal solution to this BLP that is an extreme point of T_n and a local minimum is a global minimum. Thus branch and cut algorithms similar to those for TSP can be developed for the BTSP using cutting planes and a BLP solver. However, no implementation of such an algorithm is available in literature.

A dynamic programming approach for the BTSP was proposed by Sergeev [756]. As in the case of TSP, this algorithm is primarily of theoretical importance only.

3. Approximation Algorithms

In Chapters 5 and 6 we have studied approximation algorithms for the TSP where a *priori* mathematical guarantee could be obtained on the performance of an approximation algorithm. Chapters 8, 9, and 10 discussed implementation aspects of various practical approximation algorithms (heuristics) for TSP. The literature on approximation algorithms for BTSP is not as extensive as that of the TSP. In this section we study approximation algorithms for BTSP and MSTSP. We assume throughout this section that the edge costs are positive.

Recall that an algorithm yields a factor δ approximation for a minimization problem, if the algorithm is guaranteed to produce a solution whose objective function value is at most δ times the optimal objective function value. An algorithm yields a factor δ-approximation for a max-

imization problem if the algorithm is guaranteed to produce a solution whose objective function value is at least $1/\delta$ times the optimal objective function value.

Note that transformation used earlier between BTSP and MSTSP may not map a δ-approximate solution of BTSP to a δ-approximate solution of MSTSP. Thus, for worst case analysis of approximation algorithms we treat these two problems separately.

Theorem 7 *[661, 33] Unless P = NP, there is no polynomial time δ-approximation algorithm for the BTSP or MSTSP for any constant δ, $1 \leq \delta < \infty$.*

Proof. Let us first consider the case of symmetric BTSP. We prove the theorem by showing that any such approximation algorithm $\mathcal{A}$, if exists, can be used to solve the Hamiltonian cycle problem in polynomial time, implying P = NP. Suppose that we are given a graph G^* and we wish to test if G^* contains a Hamiltonian cycle. Convert the graph G^* to a complete graph K_n by adding the missing edges and assign a cost of $\delta + 1$ to each of these new edges. Assign a cost 1 to each of the original edges. Now we have an instance of SBTSP on K_n. If G^* contains a Hamiltonian cycle, the optimal objective function value, OPT, of our SBTSP is 1 and if the algorithm $\mathcal{A}$ is applied to this instance, it must produce a tour of value δ or less (in fact exactly equal to 1). If G^* has no Hamiltonian cycle, then OPT is $\delta + 1$. Thus the objective function value of the solution produced by $\mathcal{A}$ is less than or equal to δ precisely when G^* contains a Hamiltonian cycle. The result now follows from the NP-completeness of Hamiltonicity testing [347]. The proof for the case of ABTSP or MSTSP (symmetric and asymmetric) can be obtained in a similar way. ■

In view of Theorem 7, it is not very likely that we shall succeed in obtaining meaningful performance bound for polynomial time approximation algorithms for BTSP or MSTSP with arbitrary edge costs. However, by assuming special properties of edge costs, heuristics with guaranteed performance bounds can be obtained.

3.1. Worst Case Analysis of Heuristics for BTSP

Recall that for any $\tau \geq 1/2$, the edge costs c_e of a complete graph K_n satisfy the τ-triangle inequality [22], if for any three nodes i, j, k of K_n, $c_{ij} \leq \tau(c_{ik} + c_{kj})$. If $\tau = 1$, τ-triangle inequality reduces to the triangle inequality. $\tau = 1/2$ forces all the edges of K_n to be of same cost. If $\tau > 1$, τ-triangle inequality can be viewed as a relaxation of the triangle inequality, where as for $1/2 < \tau < 1$, it is a restriction on the

triangle inequality. The following lemma is an immediate consequence of the τ-triangle inequality.

Lemma 8 *Suppose the edge costs c_e of K_n satisfy the τ-triangle inequality for some $\tau \geq 1/2$. Let $P(i,j) = (e_1, e_2, \ldots, e_r)$ be a path in K_n joining vertices i and j. Then, $c_{ij} \leq \min\{S_1, S_2\}$, where $S_1 = \tau^{r-1}c_{e_r} + \sum_{k=1}^{r-1} \tau^k c_{e_k}$ and $S_2 = \tau^{r-1}c_{e_1} + \sum_{k=1}^{r-1} \tau^k c_{e_{r+1-k}}$.*

The concept called the *power of a graph* plays a central role in our approximation algorithms for SBTSP

Definition 9 *Let $G = (N, E)$ be a graph (not necessarily complete) and t be a positive integer. The t^{th} power of G is the graph $G^t = (N, E^t)$, where there is an edge $(u, v) \in E^t$ whenever there is a path from u to v in G with at most t edges.*

For any graph $G = (N, E)$ with edge costs c_e for $e \in E$, $c_{\max}(G)$ denotes $\max\{c_e : e \in E\}$. Similarly, $c_{\min}(G) = \min\{c_e : e \in E\}$. A similar notation will be used if G is replaced by a collection S of edges of G. We use the phrase $e \in G$ and $e \in E$ interchangeably.

G^2 is called the square of the graph G and G^3 is the cube of G.

Lemma 10 *Suppose the edge costs c_e of K_n satisfy the τ-triangle inequality for some $\tau \geq 1/2$. Let G be a subgraph of K_n. Then*

$$c_{\max}(G^t) \leq \begin{cases} t\ c_{\max}(G) & \text{if } \tau = 1 \\ \frac{\tau}{\tau-1}(2\tau^{t-1} - \tau^{t-2} - 1)c_{\max}(G) & \text{if } \tau > 1 \\ \frac{\tau}{\tau-1}(\tau^{t-1} + \tau - 2)c_{\max}(G) & \text{if } \tau < 1 \end{cases}$$

Proof. Let (i,j) be an arbitrary edge of G^t. By definition of G^t, there exists a path $P(i,j) = (e_1, e_2, \ldots, e_r)$ in G^t from i to j of length at most t. By Lemma 8,

$$c_{ij} \leq \tau^{r-1}c_{e_r} + \sum_{k=1}^{r-1} \tau^k c_{e_k} \leq t\ c_{\max}(G) \text{ for } \tau = 1.$$

Similarly, if $\tau > 1$, then by Lemma 8 we have,

$$\begin{aligned} c_{ij} &\leq \tau^{r-1} c_{e_r} + \sum_{k=1}^{r-1} \tau^k c_{e_k} \\ &\leq (\tau^{r-1} + \sum_{k=1}^{r-1} \tau^k) c_{\max}(G) \\ &\leq (\tau^{t-1} + \sum_{k=1}^{t-1} \tau^k) c_{\max}(G) \\ &= \frac{\tau}{\tau - 1}(2\tau^{t-1} - \tau^{t-2} - 1) c_{\max}(G) \end{aligned}$$

The case $\tau < 1$ can be proved in a similar way. ■

Theorem 11 *Suppose the edge costs c_e of K_n satisfy the τ-triangle inequality for some $\tau \geq 1/2$. Let S be a spanning subgraph of K_n such that $c_{\max}(S)$ is a lower bound for the optimal objective function value of BTSP and let $\mathcal{H}^*$ be an optimal solution to BTSP on K_n. If S^t, $1 \leq t < n$ and integer t, contains a Hamiltonian cycle $\mathcal{H}$, then*

$$c_{\max}(\mathcal{H}) \leq \begin{cases} t\ c_{\max}(\mathcal{H}^*) & \text{if } \tau = 1 \\ \frac{\tau}{\tau-1}(2\tau^{t-1} - \tau^{t-2} - 1) c_{\max}(\mathcal{H}^*) & \text{if } \tau > 1 \\ \frac{\tau}{\tau-1}(\tau^{t-1} + \tau - 2) c_{\max}(\mathcal{H}^*) & \text{if } \tau < 1 \end{cases}.$$

Proof. Assume that $\tau = 1$. By definition of power of a graph, $c_{\max}(\mathcal{H}) \leq \max(S^t) \leq t\ c_{\max}(S)$. The last inequality follows from Lemma 10. From the optimality of $\mathcal{H}^*$, $c_{\max}(S) \leq c_{\max}(H^*)$. The result now follows immediately. The case $1/2 \leq \tau < 1$ and $\tau > 1$ can be proved in a similar way. ■

Theorem 11 and Lemma 10 are generalizations of corresponding results by Hochbaum and Shmoys [449] proved for the case $\tau = 1$.

Let us now discuss a simple heuristic for BTSP called *the bottleneck double tree heuristic*, which is an adaptation of the double tree heuristic for the TSP.

Bottleneck Double Tree Heuristic

Step 1: Compute a bottleneck spanning tree T of K_n.

Step 2: Duplicate the edges of T to form an Eulerian multigraph T^*.

Step 3: Identify an Eulerian tour in T^*. Traverse T^* along this Eulerian tour and introduce shortcuts whenever a previously visited node is encountered, to produce a tour in K_n.

Note that Step 3 of the algorithm allows a lot of flexibility. It is possible to construct examples where the double tree heuristic produces the worst solution. The quality of the solution produced however depends on the order in which the edges are traversed in the Eulerian tour. The solution generated by different orders of traversal may be different. Among all such solutions, identifying the best one can be shown to be a NP-hard problem. It may be noted that the cube of any connected graph is Hamiltonian connected (and hence Hamiltonian) [448, 449]. It has been shown by Hobbs [448] that a tour can be generated in Step 3 of the algorithm by introducing shortcuts to only paths of T^* of length 3 or less and such a tour, say $\mathcal{H}'$, belongs to T^3. (See also [100].) By Theorem 11, if the edge costs satisfy τ-triangle inequality, then

$$c_{\max}(\mathcal{H}') \leq \begin{cases} 3\ c_{\max}(\mathcal{H}^*) & \text{if } \tau = 1 \\ \frac{\tau}{\tau-1}(2\tau^2 - \tau - 1)c_{\max}(\mathcal{H}^*) & \text{if } \tau > 1 \\ \frac{\tau}{\tau-1}(\tau^2 + \tau - 2)c_{\max}(\mathcal{H}^*) & \text{if } 1/2 \leq \tau < 1 \end{cases}$$

where $\mathcal{H}^*$ is an optimal solution to SBTSP. The short-cutting phase as described above can be done in $O(n)$ time, (see for example Hobbs [448]). Also, the bottleneck spanning tree in Step 1 can be obtained in $O(m)$ time, where m is the number of edges in G [153]. Thus we have the following theorem.

Theorem 12 *If the edge costs satisfy the τ-triangle inequality, then the double tree algorithm produces a solution to the SBTSP in $O(n^2)$ time with a performance bound δ, where*

$$\delta = \begin{cases} 3 & \text{if } \tau = 1 \\ \frac{\tau}{\tau-1}(2\tau^2 - \tau - 1) & \text{if } \tau > 1 \\ \frac{\tau}{\tau-1}(\tau^2 + \tau - 2) & \text{if } 1/2 \leq \tau < 1 \end{cases}$$

Let us now consider a general heuristic algorithm for BTSP based on the concept of power of a graph.

Algorithm Power(S,t)

Step 1: Construct a connected spanning subgraph S of K_n such that $c_{\max}(S)$ is a lower bound on the optimal objective function value of the BTSP.

Step 2: Find the smallest positive integer t such that S^t is Hamiltonian.

Step 3: Output any Hamiltonian cycle in S^t.

When S is a bottleneck spanning tree, and $t = 3$, Power(S,t) is identical to the double tree algorithm if Step 3 of power(S,t) is implemented

using Steps 2 and 3 of the double tree algorithm. The complexity and performance bound in this case are the same as those of the double tree algorithm. We now show that by investing the same amount of time a better performance bound can be achieved.

Recall that the objective function value of the bottleneck biconnected spanning subgraph problem (BBSSP) is a lower bound on the optimal objective function value of BTSP. Thus, we could use a bottleneck biconnected spanning subgraph for S in Step 1 of algorithm *Power(S,t)*. As we have seen earlier, BBSSP on K_n can be solved in $O(n^2)$ time, (see Timofeev [793], Parker and Rardin [661], Punnen and Nair [685], and Manku [578]).

Now, for a biconnected graph, S, what is the smallest value of t such that S^t is Hamiltonian? Fleischner [312] proved that this value of t is 2. We state this elegant result in the following theorem.

Theorem 13 *The square of every biconnected graph is Hamiltonian.*

We are now left with the task of generating a tour in the square of a biconnected graph. Lau [542, 543] suggested an $O(n^2)$ algorithm to accomplish this. (See also Rardin and Parker [697].) Thus, by Theorem 11, the performance bound δ of this algorithm is 2τ (using $t = 2$) for $\tau \geq 1/2$. We thus have the following theorem.

Theorem 14 *If the edge costs of K_n satisfy τ-triangle inequality, then a 2τ-approximate solution to the BTSP can be obtained in $O(n^2)$ time.*

Algorithm $Power(S, t)$ for $t = 2$, and 3, with $\tau = 1$ was published first by Doroshko and Sarvanov [260] in a Russian paper in 1981. The case $t = 2$, $\tau = 1$ was obtained independently by Parker and Rardin [661]. Hochbaum and Shmoys [449] considered the case for general t with $\tau = 1$. We now show that improving the performance bound of Theorem 14 for any polynomial time algorithm and any $\tau > 1/2$ amounts to P=NP. This result was proved by Doroshko and Sarvanov [260], Parker and Rardin [661], and Hochbaum and Shmoys [449] for the case $\tau = 1$.

Theorem 15 *Unless P = NP, there is no polynomial time $2\tau-\epsilon$ approximation algorithm for BTSP on K_n, with edge costs satisfying τ-triangle inequality, for any $\epsilon > 0$, $\tau > 1/2$.*

Proof. We prove the theorem by showing that for any $\epsilon > 0$, $\tau > 1/2$, a polynomial time $2\tau - \epsilon$ approximation algorithm $\mathcal{A}$ for the BTSP on K_n, with edge costs satisfying τ-triangle inequality, can be used to test Hamiltonicity of an arbitrary graph (digraph), establishing P=NP. Let

G be an arbitrary graph (digraph) with each edge of cost one. Convert G into a complete graph (digraph) by adding new edges of cost 2τ each. Clearly the costs of edges of this complete graph (digraph) satisfy τ-triangle inequality. It can be verified that $\mathcal{A}$ produces a tour $\mathcal{H}$ with worst case error bound $2\tau - \epsilon$ precisely when $\mathcal{H}$ is a tour in G. This completes the proof. ■

Bollobás, Fenner, and Frieze [128] discussed an approximation algorithm for BTSP along with its probabilistic analysis. This algorithm uses a variation of the algorithm HAM discussed in Chapter 7.

3.2. Approximation Algorithms for MSTSP

Let us now consider an approximation algorithm for the MSTSP. Although algorithms discussed in the previous subsection could be easily modified to generate corresponding algorithms for MSTSP, the performance guarantees do not translate in the same way.

The basic idea of this approximation algorithm is to solve a 'relaxation' of the MSTSP, which provides a lower bound on the optimal objective function value, and generate a Hamiltonian cycle from the optimal solution of this relaxation. Sufficient conditions for existence of a Hamiltonian cycle in a graph form the key to our algorithm. Some well known such conditions are summarized in the following theorem.

Theorem 16 *If a graph $G = (N, E)$ on n nodes satisfies any of the following properties $\mathcal{P}1, \mathcal{P}2, \ldots, \mathcal{P}6$, then G is Hamiltonian.*

$\mathcal{P}1$: *$deg(v) \geq \lceil n/2 \rceil$ for all $v \in N$ (Dirac [256]).*

$\mathcal{P}2$: *$deg(u) + deg(v) \geq n$ for all pairs of non-adjacent vertices u and v of G. (Ore [636]).*

$\mathcal{P}3$: *$i < n/2$ implies $deg(i) > i$ or $deg(n-1) > n - i$, where the nodes $1, 2, ..., n$ of G are labeled in the ascending sequence of their degrees. (Chvátal [194]).*

$\mathcal{P}4$: *$\kappa(G) \geq \alpha(G)$, where $\kappa(G)$ is the connectivity number and $\alpha(G)$ is the independence number of G. (Chvátal-Erdos [199]).*

$\mathcal{P}5$: *$\theta(G)n \geq \alpha(G)$ where $\theta(G) = \min\{t(S) : S \subset N, S \neq \emptyset\}$ with $t(S) = \frac{|\overline{S} \cap N(S)|}{|\overline{S}|}$ and $N(S)$ is the neighborhood of S (Lu [571]).*

$\mathcal{P}6$: *G has minimum degree k, $n \geq 6k$ and $|E| > \binom{n-k}{2} + k^2$ (Erdos [276]).*

Let $\mathcal{F}(\mathcal{P})$ be the family of all spanning subgraphs of K_n satisfying a prescribed property $\mathcal{P}$ which guarantees that elements of $\mathcal{F}(\mathcal{P})$ are Hamiltonian. Consider the *maxmin* problem (MMP($\mathcal{P}$))

$$\begin{array}{ll} \text{Maximize} & w_{\min}(S) \\ \text{Subject to} & S \in \mathcal{F}(\mathcal{P}) \end{array}$$

Let $S(\mathcal{P})$ be an optimal solution to MMP($\mathcal{P}$) with optimal objective function value $z(\mathcal{P})$ and $\mathcal{H}^*$ be an optimal solution to the MSTSP. Clearly $z(\mathcal{P}) \leq w_{\min}(\mathcal{H}^*)$. Consider the following approximation algorithm for MSTSP.

The $\mathcal{P}$-relaxation Algorithm

Step 1: Solve MMP($\mathcal{P}$). Let $S(\mathcal{P})$ be an optimal solution generated.

Step 2: Output any Hamiltonian cycle $\mathcal{H}(\mathcal{P})$ in $S(\mathcal{P})$.

The complexity of this algorithm depends on the ability to solve MMP($\mathcal{P}$) and to generate $\mathcal{H}(\mathcal{P})$ from $S(\mathcal{P})$, which in turn depends on the property $\mathcal{P}$. We will study the case when $\mathcal{P} = \mathcal{P}1$ or $\mathcal{P}2$. The $\mathcal{P}$-relaxation algorithm is based on the original ideas of Arkin et al [33] where the case $\mathcal{P} = \mathcal{P}1$ was considered.

Let us first consider the case $\mathcal{P} = \mathcal{P}2$. In this case, MMP($\mathcal{P}$) can be solved using the binary search version of the threshold algorithm [269] for bottleneck problems. The complexity of the algorithm, in this case, would be $O(n^2 \log n)$. Since $\mathcal{P}1$ implies $\mathcal{P}2$, the case $\mathcal{P} = \mathcal{P}1$ can be solved by the algorithm for $\mathcal{P} = \mathcal{P}2$. However, this special case can be solved more efficiently in $O(n^2)$ time as observed by Arkin et al [33]. For each node i of K_n, compute the $\lceil n/2 \rceil^{th}$ largest weight of edges that are adjacent i (counting multiplicity). This can be done in $O(n)$ time [5]. Let δ be the smallest of all these values. Remove all edges with weight less than δ from K_n. The resulting graph is an optimal solution to MMP($\mathcal{P}1$).

A Hamiltonian cycle $\mathcal{H}(\mathcal{P}1)$ in $S(\mathcal{P}1)$ can be generated in $O(n^2)$ time. (see Arkin et al[33].) This algorithm is based on a constructive proof of sufficiency of $\mathcal{P}1$ for the existence of a Hamiltonian cycle in a graph. The same algorithm works for generating a Hamiltonian cycle in $S(\mathcal{P}2)$.

The following lemma is useful in establishing a performance bound for the $\mathcal{P}$-relaxation algorithm.

Lemma 17 *[33] Let R be a subset of nodes of K_n such that $|R| > \lfloor n/2 \rfloor$. Then for any Hamiltonian cycle of K_n, there exists an edge joining two nodes of R.*

Let $e^* = (u, v)$ be an edge in $S(\mathcal{P}2)$ of smallest weight. Without loss of generality we assume $S(\mathcal{P}2)$ contains all edges of weight more than w_{e^*} and it is 'minimal' in the sense that removal of any more edges of weight w_{e^*} from $S(\mathcal{P}2)$ violates the condition of $\mathcal{P}2$. Hence, there exists a node i in $S(\mathcal{P}2)$ such that degree of i is less than or equal to $\lfloor n/2 \rfloor$. Consider the set

$$R = \{i\} \cup \{j : (i, j) \notin S(\mathcal{P}2)\}.$$

Since $deg(i) \leq \lfloor n/2 \rfloor$, we have $|R| > \lceil n/2 \rceil$. Let $\mathcal{H}^*$ be an optimal solution to MSTSP. By Lemma 17 $\mathcal{H}^*$ contains an edge f joining two nodes of R. Thus

$$w_{\min}(\mathcal{H}^*) \leq w_f.$$

Now, suppose the edge weights of K_n satisfy τ-triangle inequality for some $\tau \geq 1/2$. Since for any $j \in R, j \neq i$, weight of the edge (i, j) is no more than w_{e^*}, it follows that $w_f \leq 2w_{e^*}$. Hence, for any Hamiltonian cycle $\mathcal{H}(\mathcal{P}2)$ in $S(\mathcal{P}2)$,

$$w_{\min}(\mathcal{H}^*) = w_f \leq 2\tau w_{e^*} \leq 2\tau w_{\min}(\mathcal{H}(\mathcal{P}2)).$$

The same performance bound holds for the case $\mathcal{P} = \mathcal{P}1$ since $\mathcal{P}1$ implies $\mathcal{P}2$. The forgoing discussion is summarized as follows.

Theorem 18 *The $\mathcal{P}$-relaxation algorithm produces a solution to MSTSP on K_n with objective function value at least $\frac{1}{2\tau}$ times the optimal objective function value. When $\mathcal{P} = \mathcal{P}1$ (or $\mathcal{P}2$), the complexity of the algorithm is $O(n^2)$(or $O(n^2 \log n)$).*

It can be shown that the performance bound given above is the best possible for the MSTSP whenever the edge weights satisfy the τ-triangle inequality for $\tau > 1/2$.

3.3. Experimental Analysis of Heuristics

Unlike the TSP, not much literature exists on experimental analysis of heuristics for the BTSP. The algorithms discussed in the previous subsections, without further modifications, may not work well in practice although they are best possible from the worst case analysis point of view. Some quick heuristics for BTSP can be obtained by straightforward modifications of well known construction and/or improvement heuristics for the TSP, such as arbitrary insertion heuristic and its variations, nearest neighbor algorithm, patching algorithms (especially in the case of asymmetric version of BTSP), k-opt heuristic, etc. However, no computational results are reported in literature on any such adaptations. Because of the close similarity of these heuristics with their TSP

counterparts, we shall not elaborate on them. Similar heuristics can be constructed for the case of MSTSP. Garfinkel and Gilbert [349] proposed a heuristic for BTSP which is based on the threshold algorithm.

We now discuss a heuristic based on the generalized threshold algorithm, called the *approximate generalized threshold algorithm* which can be considered as an extension of the heuristic by Garfinkel and Gilbert [349]. This algorithm differs from the generalized threshold algorithm only in the way we solve the TSP in Step 4. In the approximation version, we simply use an approximation algorithm for TSP. Preliminary computational results discussed in [691], using Helsgaun's implementation [446] of the Lin-Kernighan Heuristic as the approximate TSP solver produced optimal solutions for many problems from the TSPLIB. For this implementation, the 2-max lower bound was used as the initial lower bound. For many Euclidean problems, this lower-bound turned out to be the optimal objective function value. The incremental search version produced optimal solutions for almost all these problems in one iteration. Various refinements and hybrids of the approximate generalized threshold algorithm are discussed in [691] along with computational results.

4. Polynomially Solvable Cases of BTSP

In this section we use the permutation definition of BTSP mentioned in Section 1. This will keep the section consistent with analogous results for TSP discussed in Chapter 11. Though the diagonal elements of the cost matrix C do not play any role in the definition of BTSP, various results in this section require specific values of the elements c_{ii}.

The following result will be useful in what follows.

Lemma 19 *[361] Let L be a lower bound on the optimal objective function value of BTSP(C). Define matrix C' as $c'_{ij} = \max\{c_{ij} - L, 0\}$. Then a tour γ is an optimal solution to BTSP(C) if and only if it is an optimal solution to BTSP(C').*

We start with a minor generalization of the classical result of Lawler [546] (see also [361]). We call a cost matrix C a *max-Lawler matrix* if and only if it satisfies the following conditions.

(i) For all $1 < i \leq j < k \leq n$, $\max\{c_{k1}, c_{ji}\} \geq \max\{c_{ki}, c_{j1}\}$.

(ii) For all $1 \leq i < j \leq k < n$, $\max\{c_{ni}, c_{kj}\} \leq \max\{c_{nj}, c_{ki}\}$.

It may be noted that every upper triangular matrix C is a max-Lawler matrix.

Theorem 20 *If C is a max-Lawler matrix then the problem BTSP(C) can be solved in $O(n^{2.5})$ time.*

Proof. A modification of the Lawler's algorithm for TSP with upper triangular cost matrix (see Chapter 11, Section 4.5), with the initial permutation π chosen as an optimal bottleneck assignment solution on C with $\pi(n) = 1$, produces an optimal solution to this subclass of BTSP. The proof is similar to the one in the case of TSP with upper triangular cost matrix. Since the bottleneck assignment problem can be solved in $O(n^{2.5})$ time [686], the result follows. ■

If C is an upper triangular matrix, then the optimal objective function value of BTSP(C) is clearly non-negative. Therefore, using Lemma 19, we can assume that C is a non-negative, upper triangular matrix with diagonal entries as zeros. This allows us to convert the problem of finding an optimal bottleneck assignment solution π with $\pi(n) = 1$, to the problem of finding an optimal bottleneck path from node 1 to node n in $\overleftrightarrow{K}_n$ [361] and the latter can be solved in $O(n^2)$ time.

4.1. Pyramidally Solvable Cases of BTSP

In this section, we use extensively various notations and definitions introduced in Chapter 11. An instance BTSP(C) is said to be *pyramidally solvable* if and only if there exists an optimal tour that is pyramidal. Given any cost matrix C, the problem of finding an optimal pyramidal tour for BTSP(C) can be solved in $O(n^2)$ time by transforming it into the problem of finding an optimal bottleneck path of a certain type in a directed, acyclic multigraph using idea similar to that used in the case of pyramidal TSP in Chapter 11. We give below the details.

Define a directed, acyclic multigraph $\hat{G} = (\hat{N}, E_u \cup E_l)$ where, $\hat{N} = \{1, 2, \ldots, n-1\}$ and for each $1 \leq i < j \leq n-1$, we have two arcs e^u_{ij} and e^l_{ij}, each from node i to node j, with $e^u_{ij} \in E_u$ and $e^l_{ij} \in E_l$. We call a path in $\hat{G}$ with arcs alternately in E_u and E_l an *alternating path* (*a-path*). We associate with each arc in $\hat{G}$ a set of arcs in $\overleftrightarrow{K}_n$ in such a way that every *a-path* in $\hat{G}$ from node 1 to node $n-1$ corresponds to a unique pyramidal tour on N and vice versa. We give below an example.

Let $n = 12$ and consider the pyramidal tour $\gamma = (1, 4, 5, 6, 9, 10, 12, 11, 8, 7, 3, 2, 1)$.

The fact that $\gamma(1) = 4$ implies $\gamma(3) = 2$ and $\gamma(2) = 1$. Hence, we associate with arc $e^u_{1,3}$ the set of arcs $\{(1,4), (3,2), (2,1)\}$ in $\overleftrightarrow{K}_{12}$ and we assign to $e^u_{1,3}$ weight $w^u_{1,3} = \max\{c_{1,4}, c_{3,2}, c_{2,1}\}$. Similarly, $\gamma(6) = 9$ implies that $\gamma(8) = 7$. So we associate with $e^u_{6,8}$ the set of arcs $\{(6,9), (8,7)\}$ and assign it weight $w^u_{6,8} = \max\{c_{6,9}, c_{8,7}\}$. Using the

same argument, we associate with arcs $e^u_{10,11}, e^l_{3,6}$ and $e^l_{8,10}$ arc sets $\{(10,12),(12,11)\}, \{(7,3),(4,5),(5,6)\}$ and $\{(11,8),(9,10)\}$, respectively, and assign them weights $w^u_{10,11} = \max\{c_{10,12}, c_{12,11}\}$, $w^l_{3,6} = \max\{c_{7,3}, c_{4,5}, c_{5,6}\}$ and $w^l_{8,10} = \max\{c_{11,8}, c_{9,10}\}$. Then γ corresponds to the *a-path* $e^u_{1,3} - e^l_{3,6} - e^u_{6,8} - e^l_{8,10} - e^u_{10,11}$ in $\hat{G}$. It is not difficult to verify that if we assign the following weights to the arcs of $\hat{G}$, then the bottleneck weight of each *a-path* equals the bottleneck cost of the corresponding pyramidal tour.

For each $1 \leq i < j \leq n-1$, assign to e^u_{ij} and e^l_{ij} weights

$$\begin{aligned} w^u_{ij} &= \max\{c_{i,j+1}, \max\{c_{k,k-1} : k \in U_{ij}\}\} \text{ and} \\ w^l_{ij} &= \max\{c_{j+1,i}, \max\{c_{k,k+1} : k \in L_{ij}\}\}, \end{aligned}$$

respectively. Here, for $1 < i < j < n-1$, $U_{ij} = \{i+2, \ldots, j\}$ and $L_{ij} = \{i+1, \ldots, j-1\}$; for $1 < j < n-1, U_{1j} = \{2, \ldots, j\}$ and $L_{1j} = \{1, \ldots, j-1\}$; for $1 < i < n-1, U_{i,n-1} = \{i+2, \ldots, n\}$ and $L_{i,n-1} = \{i+1, \ldots, n-1\}$ and $U_{1,n-1} = \{2, \ldots, n\}$ and $L_{1,n-1} = \{1, \ldots, n-1\}$.

Our problem is thus, equivalent to the problem of finding an optimal bottleneck *a-path* (an *a-path* with largest edge weight as small as possible) in $\hat{G}$ from node 1 to node $(n-1)$. The weights of all the arcs in $E_u \cup E_l$ can be calculated in $O(n^2)$ time and an optimal bottleneck *a-path* in $\hat{G}$ can be obtained in $O(n^2)$ time by simple modifications of standard shortest path algorithms.

We now present results on polynomially testable sufficiency conditions on C for BTSP(C) to be pyramidally solvable. Lemma 21 will be useful in proving the sufficiency conditions that follow. Here we use the concepts of upper minor $C^{(U,i,j,k)}$ and lower minor $C^{(L,i,j,k)}$ of a matrix C, hereditary matrix properties and a permutation $\pi^{(i)}$ corresponding to any permutation π on N and $i \in N$. These are defined in Chapter 11. Also, we associate with any permutation π on N a digraph $G_\pi = (N, E_\pi)$ where $E_\pi = \{(i, \pi(i)) : i \in N\}$. For any matrix property $\mathcal{P}$, let $MBVI(\mathcal{P})$ (minimum bottleneck violator index of $\mathcal{P}$) be the smallest integer n, such that there exists an $n \times n$ matrix C, satisfying property $\mathcal{P}$, for which BTSP(C) does not have any optimal tour that is pyramidal. Let $\mathcal{P}$ be a hereditary matrix property with $MBVI(\mathcal{P}) < \infty$. Let $n = MBVI(\mathcal{P})$.

Lemma 21 *Suppose an $n \times n$ cost matrix C satisfies property $\mathcal{P}$ and none of the optimal tours to BTSP(C) is pyramidal. Then the following statements are true.*

(i) *If for all* p, q, $3 \leq p < q - 1 \leq n - 1$, *each of the upper minors* $C^{(U,p+1,q,p)}$ *and* $C^{(U,q,p+1,p)}$ *satisfies property* $\mathcal{P}$, *then every optimal tour to BTSP(C) has node* $(n-1)$ *as one of its peaks.*

(ii) *If for all* p, q, $1 \leq q < p - 1 \leq n - 4$, *each of the lower minors* $C^{(L,p-1,q,p)}$ *and* $C^{(L,q,p-1,p)}$ *satisfies property* $\mathcal{P}$, *then every optimal tour to BTSP(C) has node* 2 *as one of its valleys.*

Proof. We prove the result by contradiction. Thus, if possible, let C be an $n \times n$ matrix satisfying $\mathcal{P}$ such that,

(i) none of the optimal tours for BTSP(C) is pyramidal;

(ii) for all p, q, $3 \leq p < q - 1 \leq n - 1$, each of the upper minors $C^{(U,p+1,q,p)}$ and $C^{(U,q,p+1,p)}$ satisfies $\mathcal{P}$; and

(iii) there exists an optimal tour γ to BTSP(C) with the second largest peak $p < n - 1$.

(The case when the lower minors rather than the upper minors of C satisfy property $\mathcal{P}$ will follow similarly.) Then G_γ contains a pyramidal subpath P_{uv} on node set $\{p+1, \ldots, n\}$, where either $u = p+1 < v \leq n$ or $v = p+1 < u \leq n$. Let γ' be the unique, non-trivial subtour of $\gamma^{(p+1)}$ on node set $\{1, 2, \ldots, p+1\}$ and let $C' = C^{(U,v,u,p)}$. By the choice of the matrix C, there exists an optimal tour ψ' for BTSP(C') that is pyramidal. If we replace node $(p+1)$ in $G_{\psi'}$ by the path P_{uv} in G_γ, the resultant digraph corresponds to a pyramidal tour ψ on N where

$$\psi(i) = \begin{cases} \psi'(i) & \text{for all } i \in \{1, 2, \ldots, p\} - \{\psi'^{-1}(p+1)\} \\ u & \text{for } i = \psi'^{-1}(p+1) \\ \psi'(p+1) & \text{for } i = v \\ \gamma(i) & \text{for all other } i \end{cases}$$

Let $a = \max\{c_{ij} : (i, j) \text{ is an arc in } P_{u,v}\}$. Then,

$$c_{\max}(\psi) = \max\{a, c'_{\max}(\psi')\} \text{ and } c_{\max}(\gamma) = \max\{a, c'_{\max}(\gamma')\}.$$

Hence, $c_{\max}(\psi) - c_{\max}(\gamma) \leq 0$. Thus, ψ is an optimal solution to BTSP(C), contradicting the choice of C. ■

As an immediate consequence of the above lemma we have the following.

Let $\mathcal{P}$ be a hereditary matrix property. Let $n = MBVI(\mathcal{P}) < \infty$.

Corollary 22 *Suppose property* $\mathcal{P}$ *is an upper hereditary (a lower hereditary) matrix property. Let* C *be an* $n \times n$ *matrix satisfying property* $\mathcal{P}$ *for which none of the optimal tours to BTSP(C) is pyramidal. Then*

every optimal tour to BTSP(C) has node $(n-1)$ as one of its peaks (node 2 as one of its valleys).

The property defined below is the 'max-version' of the property BK-II(k) discussed in Chapter 11.

Definition 23 *For any integer $k \geq 4$, an $n \times n$ matrix C satisfies property max-BK-II(k) if and only if the following two conditions hold.*

1 *For any p, q, $k-1 \leq p < q \leq n$, consider any two sets $\{i_1, i_2, \ldots, i_{(k-2)}\}$ and $\{j_1, j_2, \ldots, j_{(k-2)}\}$ of $(k-2)$ distinct integers each such that,*

   (i) $1 \leq i_1, i_2, \ldots, i_{(k-2)} < p$; *and* $1 \leq j_1, j_2, \ldots, j_{(k-2)} < p$;
   (ii) $i_u \neq j_v$ *for all* $u \neq v$; *and*
   (iii) $|\{i_1, i_2, \ldots, i_{k-2}\} \cup \{j_1, j_2, \ldots, j_{k-2}\}| \geq k-1$.

   Then for the $k \times k$ submatrix C' of C corresponding to the rows and columns $\{i_1, i_2, \ldots, i_{(k-2)}, p, p+1\}$ and $\{j_1, j_2, \ldots, j_{(k-2)}, p, q\}$ or $\{i_1, i_2, \ldots, i_{(k-2)}, p, q\}$ and $\{j_1, j_2, \ldots, j_{(k-2)}, p, p+1\}$, respectively, there exists an optimal solution γ to BTSP(C') in which node $(k-1)$ is adjacent to node k, (that is, either $\gamma(k) = k-1$ or $\gamma(k-1) = k$).

2 *For any $r \leq k$, BTSP($C^{(U,r)}$) is pyramidally solvable.*

Theorem 24 *If C is a max-BK-II(k) matrix for some integer $4 \leq k \leq n$, then BTSP(C) is pyramidally solvable.*

Proof. It is easy to see that the property max-BK-II(k) is upper hereditary. If the result is not true then there exist k, n, $4 \leq k < n < \infty$, such that $MBVI(max-BK-II(k)) = n$. Let C be an $n \times n$ cost matrix satisfying the property max-BK-II(k) such that there does not exist any optimal solution to BTSP(C) that is pyramidal. Let γ be an optimal solution to BTSP(C). By Lemma 21, $(n-1)$ is a peak of γ. Choose a set $\{i_1, i_2, \ldots, i_{k-2}\}$ of distinct nodes in $\{1, 2, \ldots, n-2\}$ including the nodes $\gamma^{-1}(n-1)$ and $\gamma^{-1}(n)$. Remove the arcs $\{(i_j, \gamma(i_j)) : 1 \leq j \leq k-2\} \cup \{(n, \gamma(n)), (n-1, \gamma(n-1))\}$ from G_γ to get $k-2$ node-disjoint, paths $\{P_{u_i,v_i} : 1 \leq i \leq k-2\}$, in addition to isolated nodes n and $(n-1)$. If we contract the paths $\{P_{u_i,v_i} : 1 \leq i \leq k-2\}$ in G_γ to nodes $\{1', 2', \ldots, (k-2)'\}$, respectively, the resultant digraph defines a tour γ' on the node set $\{1', 2', \ldots, (k-2)', n-1, n\}$. Let C' be the submatrix of C corresponding to the rows $\{v_1, v_2, \ldots, v_{(k-1)}, n-1, n\}$ and columns $\{u_1, u_2, \ldots, u_{(k-1)}, n-1, n\}$. Let us index the rows and columns of C' by

$\{1', 2', (k-2)', n-1, n\}$, in that order. Then, by property max-BK-II(k), there exists an optimal solution ψ' for BTSP(C') such that node n is adjacent to node $(n-1)$ in ψ'. In $G_{\psi'}$, replace the nodes $\{1', 2', \ldots, (k-2)'\}$ by the respective paths $\{P_{u_i,v_i} : 1 \leq i \leq k-2\}$ to get the digraph corresponding to a tour ψ on N. Let $a = \max\{c_{rs} : (r,s)$ is an arc in P_{u_i,v_i} for some i, $1 \leq i \leq k-2\}$. Then $c_{\max}(\gamma) = \max\{a, c'_{\max}(\gamma')\}$ and $c_{\max}(\psi) = \max\{a, c'_{\max}(\psi')\}$. Hence, $c_{\max}(\gamma) \geq c_{\max}(\psi)$ and ψ is an optimal solution to BTSP(C). But node $(n-1)$ is not a peak node of ψ. We thus have a contradiction. ■

The diagonal elements of the cost matrix C do not form part of any tour. However, solutions produced by certain algorithms for TSP and BTSP such as the Gilmore-Gomory type algorithms, and some sufficiency conditions for validity of these algorithms depend on the values of diagonal elements. The sufficiency condition below falls in this category.

A cost matrix C satisfies property $\mathcal{P}_I$ if and only if

(i) for any i, j, k such that $1 \leq i, j < k < n$ and $i \neq j$, $\max\{c_{ik}, c_{kj}\} \geq \max\{c_{ij}, c_{kk}\}$;

(ii) for any i, j such that $1 < i < j \leq n$, $\max\{c_{j,i-1}, c_{ii}\} \geq \max\{c_{ji}, c_{i,i-1}\}$;

(iii) C^T satisfies conditions (i) and (ii) where C^T is the transpose of C.

Theorem 25 *If matrix C satisfies property $\mathcal{P}_I$ then BTSP(C) is pyramidally solvable.*

Proof. If the result is not true, then there exists some $n < \infty$ such that $MBVI(\mathcal{P}_I) = n$. Let C be an $n \times n$ matrix satisfying the property $\mathcal{P}_I$ such that none of the optimal tours for BTSP(C) is pyramidal. Let γ be an optimal solution to BTSP(C). Let $\gamma^{-1}(n) = a$ and $\gamma(n) = b$. We assume that $a < b$. Since the property $\mathcal{P}_I$ is an upper hereditary matrix property, it follows by Lemma 21 that node $(n-1)$ is a peak node of γ and hence, $b < n-1$. Define a tour γ' on N as follows:

$\gamma'(i) = \gamma^{(b)}(i)$ $\forall i \in \{1, 2, \ldots, b\} - \{a\}$; $\gamma'(a) = n$; $\gamma'(i) = i-1$ $\forall i \in \{b+1, \ldots, n\}$. The digraph $G_{\gamma'}$ can be obtained from G_γ by adding and removing arcs in the following sequence. For $i = n-1, \ldots, b+1$, in that order, remove arcs (u_i, i), (i, v_i) and add arcs (u_i, v_i) and (i, i); then for $i = b+1, \ldots, n-1$, in that order, remove arcs $(n, i-1)$ and (i, i) and add arcs $(i, i-1)$ and (n, i). By property $\mathcal{P}_I$, each time the maximum of the costs of the arcs in the modified digraph either decreases or remains the same. Hence, $c_{\max}(\gamma') \leq c_{\max}(\gamma)$ and thus, γ' is an optimal solution to BTSP(C). But $(n-1)$ is not a peak of γ' and this contradicts

Lemma 21. ■

A cost matrix C is called a *max-distribution matrix* if and only if

$$\max\{c_{iv}, c_{ju}\} \geq \max\{c_{iu}, c_{jv}\} \text{ for all } i < j \text{ and } u < v.$$

Every max-distribution matrix satisfies both the properties $\mathcal{P}_I$ and max-BK-II(4). Hence, Corollary 26 below follows from theorems 24 and 25.

Corollary 26 *[147] If C is a max-distribution matrix, then BTSP(C) is pyramidally solvable.*

A cost matrix C is said to be *upward (downward) graded on its rows* if and only if $c_{ij} \leq c_{i,j+1}$ ($c_{ij} \geq c_{i,j+1}$) for all i, j. It is said to be *upward (downward) graded on its columns* if and only if C^T is downward (upward) graded on its rows. A matrix is said to be *doubly graded* if and only if it is graded (upward or downward) on both its rows and its columns. We call a matrix graded upward on its rows and downward on its columns a *(UD)-graded* matrix and we similarly define *(UU)-graded, (DU)-graded* and *(DD)-graded* matrices.

A matrix C is *diagonally outward graded* if and only if

(i) $c_{ij} \leq c_{i,j+1}$ for all $i \in N$ and $j = i, \ldots, n-1$;
(ii) $c_{ij} \geq c_{i,j+1}$ for all $i \in N$ and $j = 1, \ldots, i-1$; and
(iii) C^T satisfies conditions (i) and (ii).

C is *diagonally inward graded* if and only if $-C$ is diagonally outward graded.

Every doubly graded matrix C can be transformed into a (UU)-graded matrix or a (UD)-graded matrix by renumbering the elements of N and reordering the rows and columns of C accordingly. Every (UU)-graded matrix is a max-distribution matrix. Also, if C is diagonally outward graded, then it is a max-distribution matrix [801]. Hence by Corollary 26, in each of these cases BTSP(C) is pyramidally solvable. In the former case, the following stronger result is proved in [361].

Theorem 27 *[361] If C is a (UU)-graded matrix, then $\gamma = (1, 2, \ldots, n, 1)$ is an optimal solution to BTSP(C).*

Proof. Since C is a (UU)-graded matrix, $c_{\max}(\gamma) = \max\{c_{i,i+1} : i \in \{1, 2, \ldots, n-1\} = c_{u,u+1}$ for some u. Now for any tour ψ on N, there exist i, v such that $i \leq u < v$ and $\psi(i) = v$. But then $c_{\max}(\psi) \geq c_{iv} \geq c_{u,u+1}$. ■

Using the same arguments as in the proof of Theorem 24, we can prove the following result.

Theorem 28 *Suppose matrix C satisfies the following conditions:*

(i) *for any distinct i, j, k such that $1 \leq i, j < k < n$, $\max\{c_{ik}, c_{kj}\} \geq c_{ij}$;*

(ii) *for any i, j such that $1 < i < j \leq n$, $c_{j,i-1} \geq \max\{c_{ji}, c_{i,i-1}\}$;*

(iii) *C^T satisfies (i) and (ii).*

Then, BTSP(C) is pyramidally solvable.

A matrix C is a max-Klyaus matrix if and only if for any i, j, k such that $i < j < k$,

$$c_{ki} \geq \max\{c_{kj}, c_{ji}\} \quad \text{and} \quad c_{ik} \geq \max\{c_{ij}, c_{jk}\}.$$

A max-Klyaus matrix satisfies both property max-BK-II(4) and the conditions of Theorem 28. Hence, as a corollary to theorem 28, we get the following.

Corollary 29 *[147] If C is a max-Klyaus matrix, then BTSP(C) is pyramidally solvable.*

For other pyramidally solvable cases of BTSP, we refer the reader to [801].

4.2. Subclass of BTSP Solvable Using GG-Patchings

We now modify the GG scheme for TSP discussed in Chapter 11 in the context of BTSP. We call it BTGG scheme.

For any $S \subseteq \{1, 2, \ldots, n-1\}$, let $P(S)$ be the set of all permutations of elements of S. We denote

$$d[C, S] = \min\{c_{\max}(\psi) : \psi = \beta_{i_1} \circ \cdots \circ \beta_{i_u},\ (i_1, \ldots, i_u) \in P(S)\}.$$

(See Appendix A for the definitions of permutation β_i and operation $\circ$.) For any cost matrix C and any two permutations μ and σ on N, the permuted cost matrix $C^{\mu,\sigma}$ is defined as $c_{ij}^{\mu,\sigma} = c_{\mu(i),\sigma(j)}$. We denote $C^{\xi,\sigma}$ by C^{σ}, where ξ is the identity permutation.

The BTGG scheme works as follows. We start with a given permutation π. If π is a tour, the algorithm outputs this tour. Otherwise, we patch the subtours of π by constructing a patching pseudograph $G_p^{\pi} = (N_p^{\pi}, E_p^{\pi})$, finding an optimal spanning tree $E_p^{\pi}[T^*]$ of G_p^{π}, finding an optimal ordering of elements of $E_p^{\pi}[T^*]$ and performing patching operations corresponding to the edges in $E_p^{\pi}[T^*]$ in the optimal order. For details of various terms used above and insight into the basics of the

algorithm we refer to Chapter 11. A formal description of the algorithm is given below.

Algorithm : BTGG Scheme

Input: An $n \times n$ cost matrix C and a permutation π on N.

Step 1: If π is a tour then output π and stop. Suppose π has m subtours, for some $m > 1$, on node sets $N_1, \ldots, N_m$.

Step 2: Construct the patching pseudograph $G_p^\pi = (N_p^\pi, E_p^\pi)$ where, $N_p^\pi = \{1, \ldots, m\}$ and $E_p^\pi = \{e_i = (u, v) : 1 \leq i < n; i \in N_u; (i+1) \in N_v\}$.

Step 3: Find a spanning tree in G_p^π with edge set $E_p^\pi[T^*]$ such that $d[C^\pi, T^*]$ is minimum. Let $(i_1, i_2, \ldots, i_{m-1})$ be an ordering of the elements of T^* such that $d[C^\pi, T^*] = c_{\max}(\pi \circ \beta_{i_1} \circ \cdots \circ \beta_{i_{m-1}})$. The tour $\gamma^* = \pi \circ \beta_{i_1} \circ \cdots \circ \beta_{i_{m-1}}$ is the desired output. Stop.

4.2.1 Sufficiency Conditions for Validity of BTGG Scheme.

Now we give sets of sufficiency conditions on matrix C' under which for any C and π such that $C^\pi = C'$, the BTGG Scheme, with π as the starting permutation, produces an optimal solution to BTSP(C). For any set $Y \subseteq N$ with at least two elements, we define the range of Y as the set $\{i, i+1, \ldots, j\}$ such that $i \leq j$, $Y \subseteq \{i, i+1, \ldots, j+1\}$ and i and $j+1$ are in Y and we denote it by $[i, j]$. For definitions of various other terms used below, we refer to Chapter 11.

Definition 30 *An $n \times n$ matrix C satisfies max patching property I (MPP-I) if and only if the following condition holds.*

Let ψ be an arbitrary permutation on N with ℓ non-trivial subtours on the vertex sets $N^1, N^2, \ldots, N^\ell$. Let $\{S_1, S_2, \ldots, S_\ell\}$ be pairwise disjoint subsets of N and $S = \cup_{i=1}^{\ell} S_i$ such that

(i) for each i, each element of S_i lies in the range of N^i and each region of N^i contains at the most one element of S_i; and

(ii) $|S| \equiv (\sum_{i=1}^{\ell}(|N^i| - 1) \pmod 2)$.

Then, $c_{\max}(\psi) \geq d[C, S]$.

Theorem 31 *If an $n \times n$ matrix C' satisfies the property MPP-I, then for any cost matrix C and any permutation π on N such that $C' = C^\pi$, the BTGG scheme with π as the starting permutation produces an optimal solution to BTSP(C).*

Proof. Suppose C' satisfies the property MPP-I and let a cost matrix C and a permutation π on N be such that $C' = C^\pi$. Let η be an optimal

solution to BTSP(C) and let $\psi = \pi^{-1} \circ \eta$. Suppose ψ has ℓ non-trivial subtours on node sets $N^1, N^2, \ldots, N^\ell$. By Corollary 42 of Chapter 11, there exists a subset S of N with partition $\{S_1, S_2, \ldots, S_\ell\}$ such that (i) for each i, every element of S_i lies in the range N^i and every region of N^i contains at the most one element of S_i; (ii) $E_p^\pi[S]$ is the edge set of a spanning tree of the patching pseudograph G_p^π, and therefore, (iii) $|S| \equiv ((\sum_{i=1}^{\ell}(|N^i| - 1) \pmod 2$. Let $(i_1, i_2, \ldots, i_m)$ be an ordering of the elements of S for which $c_{\max}^\pi(\beta_{i_1} \circ \beta_{i_2} \circ \cdots \circ \beta_{i_m}) = d[C^\pi, S]$. Then, $\gamma^* = \pi \circ \beta_{i_1} \circ \beta_{i_2} \circ \cdots \circ \beta_{i_m}$ is a tour on N, $c_{\max}(\eta) = c_{\max}^\pi(\psi)$ and $c_{\max}(\gamma^*) = d[C^\pi, S]$. Hence, $c_{\max}(\eta) \geq c_{\max}(\gamma^*)$ and γ^* is an optimal solution to BTSP(C). ■

Definition 32 *A matrix C satisfies max patching property II (MPP-II) if and only if the following condition holds.*

Let ψ be an arbitrary permutation on N. Suppose ψ has ℓ non-trivial subtours and G_ψ^I (for definition, see Chapter 11, Section 4.2) has r connected components of sizes $v_1, v_2, \ldots, v_r$. Let $X_\psi^1, X_\psi^2, \ldots, X_\psi^r$ be the unions of the node sets of the non-trivial subtours of ψ corresponding, respectively, to nodes of the r connected components of G_ψ^I. Let S be any subset of N with a partition $\{S_1, S_2, \ldots, S_r\}$, such that

(i) for each i, every element of S_i lies in the range X_ψ^i, every region of X_ψ^i contains at the most one element of S_i and $|S_i| \leq (|X_\psi^i| - v_i)$; and

(ii) $|S| \equiv ((\sum_{i=1}^{r} |X_\psi^i|) - \ell) \pmod 2$. Then $c_{\max}(\psi) \geq d[C, S]$.

Theorem 33 *If a matrix C' satisfies property MPP-II then for any cost matrix C and any permutation π on N such that $C' = C^\pi$, the BTGG scheme with π as the starting permutation produces an optimal solution to BTSP(C).*

Theorem 33 can be proved along the same lines as Theorem 31 using Corollary 43 of Chapter 11.

As a corollary, we get the following result.

Corollary 34 *Suppose C^π is a max-distribution matrix. Then the BTGG scheme with π as the starting permutation produces an optimal solution to BTSP(C).*

Outline of Proof. It will be sufficient to show that C^π satisfies the property MMP-II. Thus, consider an arbitrary permutation ψ on N. Suppose ψ has ℓ non-trivial subtours $\mathfrak{C}_1, \mathfrak{C}_2, \ldots, \mathfrak{C}_\ell$ and G_ψ^I has r

connected components of sizes $v_1, v_2, \ldots, v_r$. Let $X^1_\psi, X^2_\psi, \ldots, X^r_\psi$ be the unions of the node sets of the non-trivial subtours of ψ corresponding, respectively, to nodes of the r connected components of G^I_ψ and let $[p_1, q_1], \ldots, [p_r, q_r]$ be the ranges of the sets $X^1_\psi, \ldots, X^r_\psi$, *respectively*. Let S be any subset of N with a partition $\{S_1, S_2, \ldots, S_r\}$, such that for any $1 \leq i \leq r$, every element of S_i lies in the range X^i_ψ, every region of X^i_ψ contains at the most one element of S_i and $|S_i| \leq (|X^i_\psi| - v_i)$.

For any distinct i, j belonging to the same connected component of G^I_ψ, there exist nodes k, u of $\mathfrak{C}_i$ and $\mathfrak{C}_j$, respectively, such that the ranges of $\{k, \psi(k)\}$ and $\{u, \psi(u)\}$ intersect and $c^\pi_{\max}(\psi \circ \alpha_{k,u}) \leq c^\pi_{\max}(\psi)$. Let $\psi' = \psi \circ \alpha_{k,u}$. By repeating this process, we get a permutation η such that $c^\pi_{\max}(\eta) \leq c^\pi_{\max}(\psi)$ and the node sets of the non-trivial subtours of η are $X^1_\psi, X^2_\psi, \ldots, X^r_\psi$.

For any i, $1 \leq i \leq r$, and any $j \in [p_i, q_i] - X^i_\psi$, there exists $u \in X^i_\psi$ such that $u < j < \eta(u)$. Let $\eta' = \eta \circ \alpha_{u,j}$. Then $c^\pi_{\max}(\eta') \leq c^\pi_{\max}(\eta)$. By repeating this process, we get a permutation ϕ such that $c^\pi_{\max}(\phi) \leq c^\pi_{\max}(\eta)$ and ϕ is dense on the node set $X = \cup^r_{i=1}\{p_i, \ldots, q_i\}$. Since C^π is a max-distribution matrix, it follows by Corollary 26 that $d[C^\pi, X] \leq c^\pi_{\max}(\phi)$. The result now follows from the fact that since $S \subseteq X$, $d[C^\pi, S] \leq d[C^\pi, X]$.

We do not know the complexity of implementation of the BTGG scheme when C^π is a max-distribution matrix. However when the patching pseudograph G^π_p is a multi-path or a multi-tree, BTGG-scheme can be implemented in polynomial time using the same idea as in the case of GG-scheme discussed in Chapter 11.

For an arbitrary permutation π on N, let $N^1, \ldots, N^k$ be the node sets of the non-trivial subtours, (that is, subtours with at least two nodes) of π. Let

$$d^{in}(C, \pi) = \max\{c_{i,\pi(i)} : i \in \cup^k_{j=1} N^j\}.$$

For any $S \subseteq \{1, 2, \ldots, n-1\}$, let $P(S)$ be the set of all permutations of elements of S. Let

$$d^{in}[C, S] = \min\{d^{in}(C, \psi) : \psi = \beta_{i_1} \circ \cdots \circ \beta_{i_u} \text{ and } (i_1, \ldots, i_u) \in P(S)\}.$$

Suppose π is an optimal solution to the bottleneck assignment problem on C (and therefore, the identity permutation ξ is an optimal solution to the bottleneck assignment problem on C^π). Then, if we modify the properties MPP-I and MPP-II as well as the BTGG scheme by replacing the functions $c_{\max}(\pi)$ and $d[C, S]$ by functions $d^{in}(C, \pi)$ and $d^{in}[C, S]$, respectively, the modified properties can be shown to be sufficient for the modified BTGG scheme to produce an optimal solution to BTSP(C). It

may be noted that if C^π is a max-distribution matrix, then π is an optimal solution to the bottleneck assignment problem on C. If C^π is a (UU)-graded matrix, then by Theorem 27, for $S = \{u, u+1, \ldots, v\}$ for any $1 \leq u < v < n$, $d^{in}[C^\pi, S] = \max\{c^\pi_{i,i+1} : i \in S\}$. Thus, Step 3 of the modified BTGG scheme reduces to a bottleneck spanning tree problem, where each edge $e_i \in E_p^\pi$ is assigned a weight $c^\pi_{i,i+1}$, and this problem can be solved in $O(n)$ [361]. We thus have the following result.

Corollary 35 *[361] If C^π is a (UU)-graded matrix, then the BTGG-scheme with π as the starting permutation and with the function $d[C^\pi, T^*]$ replaced by the function $d^{in}[C^\pi, T^*]$ produces an optimal solution to BTSP(C) in $O(n)$ time.*

If C^π is a (UD)-graded matrix then for permutation ψ defined as $\psi(i) = n - i + 1$, $C^{\pi\circ\psi}$ is a (UU)-graded matrix. Thus, BTSP when C^π is a (UD)-graded matrix can be solved in $O(n)$ time. Suppose C is a Gilmore-Gomory type matrix (see Chapter 11) with $f(x) \geq 0$ and $g(x) = 0$ for all x and let us assume, without loss of generality, that $b_1 \leq \cdots \leq b_n$. If we define a permutation π on N such that $a_{\pi(1)} \leq \cdots \leq a_{\pi(n)}$, then C^π is a (UU)-graded matrix. Finding such a permutation π takes $O(n \log n)$ time. Thus, in this case BTSP(C) can be solved in $O(n \log n)$ time [360].

If C is a sum matrix, (that is, $c_{ij} = a_i + b_j$ for all i, j), or a small matrix, (that is, $c_{ij} = \min\{a_i, b_j\}$), then let us assume without loss of generality that $a_1 \geq \cdots \geq a_n$. If we define π such that $b_{\pi(1)} \leq \cdots \leq b_{\pi(n)}$, then in each of these cases, C^π is a (UU)-graded matrix. Again, finding such a permutation π takes $O(n \log n)$ time. Hence, in each of these cases BTSP(C) can be solved in $O(n \log n)$ time. For the small matrix case, BTGG scheme can be implemented in $O(n)$ time using the same idea as in the case of TSP on such matrices, as discussed in Chapter 11, Section 4.3. An alternate $O(n)$ scheme for the small matrix case is given in [798] that is similar to the scheme by Gabovich [340] for TSP on such matrices. If C is a large matrix, then every tour on N has the same bottleneck cost and thus the problem is trivial.

In [361] it is shown that if the cost matrix C is a graded matrix, then BTSP(C) can be solved in $O(n^2)$ time using a modification of the BTGG scheme that allows patchings that are not GG-patchings.

4.3. BTSP on Ordered Product Matrices

A cost matrix C is called an *ordered product matrix* if and only if there exist $\{a_i, b_i : i \in N\}$ such that $a_1 \leq a_2 \leq \cdots \leq a_n$, $b_1 \geq b_2 \geq \cdots \geq b_n$ and $c_{ij} = a_i b_j$.

The following result by Van Der Veen [801] improved upon the previous $O(n^2)$ algorithm of [147] for BTSP on ordered product matrices.

Theorem 36 *[801] BTSP on an ordered product cost matrix can be solved in $O(n)$ time.*

Proof. If all the a_i's have the same sign or all the b_j's have the same sign then the cost matrix C is a doubly graded matrix and an $O(n)$ algorithm follows from Corollary 35. Suppose there exist $1 \leq u, v < n$ such that $a_u < 0 \leq a_{u+1}$ and $b_v \geq 0 > b_{v+1}$.

Case (i): $u = v$. Let $m = n - v$. In this case, we can partition the matrix C as

$$C = \begin{bmatrix} D^{1,1} & D^{1,2} \\ D^{2,1} & D^{2,2} \end{bmatrix},$$

where matrices $D^{1,1}, D^{1,2}, D^{2,1}$ and $D^{2,2}$ are of sizes $u \times u$, $u \times m$, $m \times u$ and $m \times m$, respectively, and $D^{1,1} \leq 0$, $D^{2,2} \leq 0, D^{1,2} > 0$ and $D^{2,1} \geq 0$. Obviously, for every tour γ on N, $c_{\max}(\gamma) \geq 0$ and hence, using Lemma 19, we consider matrix C' where, $c'_{ij} = \max\{C_{i,j}, 0\}$. The matrix C' has the form:

$$C' = \begin{bmatrix} 0 & A^{1,2} \\ A^{2,1} & 0 \end{bmatrix},$$

where $A^{1,2}$ is a positive (UU)-graded matrix and $A^{2,1}$ is a non-negative (DD)-graded matrix and therefore, C' is a max-distribution matrix. Hence, BTSP(C') is pyramidally solvable. For any pyramidal tour γ on N there is exactly one pair $\{i, j\} \subseteq N$ such that $i \leq u < \gamma(i)$ and $j > u \geq \gamma(j)$ and $c'_{\max}(\gamma) = \max\{c'_{i,\gamma(i)}, c'_{j,\gamma(j)}\}$. Since $A^{1,2}$ is a (UU)-graded matrix and $A^{2,1}$ is a (DD)-graded matrix, an optimal choice of such a pair of arcs $\{(i, \gamma(i)), (j, \gamma(j))\}$ belongs to the set

$$S = \{\{(u, u+2), (u+1, u-1)\}, \{(u-1, u+1), (u+2, u)\}, \\ \{(u-1, u+2), (u+1, u)\}, \{(u, u+1), (u+2, u-1)\}\}.$$

An optimal tour can thus be obtained by finding $\{(x, y), (s, t)\} \in S$ with minimum value of $\max\{c_{xy}, c_{st}\}$ and constructing a pyramidal tour containing these arcs.

Case (ii): $u < v$. Let $\ell = v - u$ and $m = n - v$. We can partition the matrix C as

$$C = \begin{bmatrix} D^{1,1} & D^{1,2} & D^{1,3} \\ D^{2,1} & D^{2,2} & D^{2,3} \\ D^{3,1} & D^{3,2} & D^{3,3} \end{bmatrix},$$

where $D^{1,1} \leq 0$, $D^{1,2} \leq 0, D^{1,3} \geq 0, D^{2,1} \geq 0, D^{2,2} \geq 0, D^{2,3} \leq 0, D^{3,1} \geq 0, D^{3,2} \geq 0$ and $D^{3,3} \leq 0$. In this case too, it is easy to see that $c_{\max}(\gamma) \geq$

0 for every tour γ on N and hence, using Lemma 19, we consider matrix C', where for all $1 \leq i, j \leq n$, $c'_{ij} = \max\{c_{ij}, 0\}$. We can partition C' as

$$C' = \begin{bmatrix} 0 & 0 & A^{1,3} \\ A^{2,1} & A^{2,2} & 0 \\ A^{3,1} & A^{3,2} & 0 \end{bmatrix},$$

where $A^{1,3}$ is a non-negative (UU)-graded matrix and each of $A^{2,1}, A^{2,2}$, $A^{3,1}$ and $A^{3,2}$ is a non-negative (DD)-graded matrix. Therefore, C' is a max-distribution matrix and hence, BTSP(C') is pyramidally solvable. The bottleneck cost of any tour on N will be decided by the arcs in the tour corresponding to the entries of the matrices $A^{1,3}$, $A^{2,1}$, $A^{2,2}$, $A^{3,1}$ and $A^{3,2}$. Using the structure of these matrices, it has been shown in [801] that there exists an optimal pyramidal tour for which the set of the arcs of the tour corresponding to the entries of these five matrices belongs to the set

$$S = \{\{(x, x-2)\}, \{(u-1, v+2)\}, \{(v+2, v), (u-1, v+1)\},$$
$$\{(u+1, u-1), (u, v+2)\}, \{(u+1, u-1), (v+2, v), (u, v+1)\}\},$$

where $x \in \{u+2, \ldots, v+1\}$ is such that $c'_{x,x-2} = \min\{c'_{i,i-2} : i \in \{u+2, \ldots, v+1\}\}$. Thus, an optimal pyramidal tour can be obtained in $O(n)$ time.

Case (iii): $u > v$. In this case, the result can be proved along the same lines as in case (ii). ■

4.4. BTSP on Symmetric, Diagonally Circular Inward Graded Matrix

In [33], $O(n)$ algorithms are presented for two subclasses of MSTSP. In the first case, each node in N is associated with a point on a line, while in the second case, each node is associated with a point on a circle. In each case, for any $i, j \in N$, weight w_{ij} is the Euclidean distance between the corresponding points. We consider this as instances of BTSP with edge costs $c_{ij} = -w_{ij}$. The case corresponding to points on a line is obviously a special case of the case of points on a circle. In the case of points on a line, the cost matrix is a symmetric, diagonally inward graded matrix and it is a Gilmore-Gomory type matrix (see Chapter 11) with $a_i = b_i$ for all $i \in N$ and $f(x) = g(x) \leq 0$ for all x. In the case of points on a circle, the cost matrix is a special case of what we call a symmetric, diagonally circular inward graded matrix (SDCI-matrix), and which we define below. We assume here that all indices and node labels are taken *modulo* n.

A symmetric cost matrix C is an *SDCI-matrix* if only if there exist $1 \leq \ell(1) \leq \ell(2) \leq \cdots \leq \ell(n) < 2n$ such that for all $i \in N$,

(i) $i \leq \ell(i) \leq n + i$;

(ii) $c_{i,x-1} \geq c_{ix}$ for all $x \in \{i+1, \ldots, \ell(i)\}$ and $c_{i,x+1} \geq c_{ix}$ for all $x \in \{\ell(i)+1, \ldots, n+i-1\}$;

(iii) $c_{x+1,y} \leq c_{xy}$ for all $y \geq \ell(i)+1$ and $x = i-1, i-2, \ldots, y-n$ and $c_{x+1,y} \geq c_{xy}$ for all $y \leq \ell(i)$ and $x = i, i+1, \ldots, y-1$.

We say that C is an SDCI-matrix with respect to $\{\ell(1), \ell(2), \ldots, \ell(n)\}$.

It should be noted that a symmetric, diagonally inward graded matrix is an SDCI-matrix with $\ell(i) = n$ for all i; and the negative of a symmetric diagonally inward graded matrix is a max-Klyaus matrix. In the case of points on a circle, if for each point i, we define $\ell(i)$ such that each of the sets of points $\{i, i+1, \ldots, \ell(i)\}$ and $\{i, i-1, \ldots, \ell(i)+1\}$ lies on an arc subtending an angle less than 180^0 at the centre of the circle, then the cost matrix is an SDCI-matrix with respect to these values of $\{\ell(1), \ell(2), \ldots, \ell(n)\}$.

For any set $S \subset N$, let us define $\lambda_S = \min\{c_{ij} : i, j \in S\}$.

The following facts given in [33] will be useful. Facts 37 and 38 are easy to verify and Fact 39 follows from Fact 38.

Fact 37 *If $n = 2k+1$, for some positive integer k, then for any subset S of N of cardinality $k+1$ and any tour γ on N, there exist u, v in S such that $\gamma(u) = v$. Therefore, λ_S is a lower bound on the optimal objective function value of BTSP(C).*

Fact 38 *If $n = 2k$, for some positive integer k, then for any pair of subsets S_1 and S_2 of N of cardinality k each, such that $S_1 \cap S_2 \neq \emptyset$, and any tour γ on N, either $\{u, \gamma(u)\} \subseteq S_1$ for some u or $\{v, \gamma(v)\} \subseteq S_2$ for some v. Hence, $\min\{\lambda_{S_1}, \lambda_{S_2}\}$ is a lower bound on the optimal objective function value of BTSP(C).*

Fact 39 *Suppose $n = 2k$, for some positive integer k. Let S_1, S_2 and S_3 be subsets of N of cardinality k each such that $\lambda_{S_1} \geq \lambda_{S_2} \geq \lambda_{S_3}$ and $\lambda_{S_3} \geq \lambda_S$ for any other subset S of N of cardinality k. Then, the following are true.*

(i) *λ_{S_3} is a lower bound on the optimal objective function value of BTSP(C).*

(ii) *If $S_1 \cap S_2 \neq \emptyset$, then λ_{S_2} is a lower bound on the optimal objective function value of BTSP(C).*

Theorem 40 *If C is an SDCI-matrix then BTSP(C) can be solved in $O(n)$ time.*

Proof. We follow the proof technique in [33] for the case of points on a circle.

Case (i) $n = 2k + 1$ **for some k:** For any $i \in N$, let $F_i = \{i, i+1, \ldots, i+k\}$ and $B_i = \{i, i-1, \ldots, i-k\}$. It follows from Fact 37 that $\lambda = \max\{\lambda_{F_i}, \lambda_{B_i} : 1 \leq i \leq n\}$ is a lower bound to the optimal objective function value of BTSP(C). Since, C is an SDCI-matrix, $\max\{\lambda_{F_i}, \lambda_{B_i}\} = \max\{c_{i,i+k}, c_{i,i-k}\}$ for all $i \in N$ and hence, $\lambda = \max\{\{c_{i,i+k}, c_{i,i-k}\} : i \in N\}$. Let $\gamma^* = (1, k+2, 2k+3, \ldots, 2k(k+1)+1, 1)$. Then, $c_{\max}(\gamma^*) = \lambda$ and, it follows from the fact that $\gcd\{n, k+1\} = 1$ that γ^* is a tour on N. Hence, γ^* is an optimal solution to BTSP(C).

Case (ii) $n = 2k$**:** In this case, let us define for any $i \in N$, $F_i = \{i, i+1, \ldots, i+k-1\}$. Since C is an SDCI-matrix, we have the following. For any $i \in N$,

(i) if $\ell(i) \geq i + k - 1$, then $c_{i,i+k-1} = \lambda_{F_i}$;

(ii) if $\ell(i) < i+k-1$, then $c_{i,i+k-1} \leq \min\{c_{i,i+k+1}, c_{i-1,i+k}, c_{i-2,i+k-1}\}$, and from (i) we have that $c_{i,i+k+1} = \lambda_{F_{i+k+1}}, c_{i-1,i+k} = \lambda_{F_{i+k}}$ and $c_{i-2,i+k-1} = \lambda_{F_{i+k-1}}$. Hence, if i_1, i_2, i_3 in N are such that $\lambda_{F_{i_1}} \geq \lambda_{F_{i_2}} \geq \lambda_{F_{i_3}}$ and $\lambda_{F_{i_3}} \geq \lambda_{F_i}$ for every other i, then $\lambda_{F_{i_j}} = c_{i_j, i_j+k-1}$ for $j = 1, 2, 3$. It now follows from Fact 39 that if $\{i_1, \ldots, i_1 + k - 1\} \cap \{i_2, \ldots, i_2+k-1\} \neq \emptyset$, then c_{i_2,i_2+k-1} is a lower bound on the optimal objective function value of BTSP(C); else, c_{i_3,i_3+k-1} is a lower bound. Consider the tour $\gamma = (i_1, (k+1+i_1), (2(k+1)+i_1), \ldots, (2k(k+1)+i_1), i_1) = (i_1, u_1, \ldots, u_{2k}, i_1)$. Let j be minimum index such that $u_j \in \{i_1 + k - 1, i_1 + 2k - 1\}$. Let γ^* be the tour $(i_1, u_1, \ldots, u_j, u_{2k}, u_{2k-1}, u_{j+1}, i_1)$. The tour γ^* attains the lower bound stated above. ■

4.5. BTSP on Symmetric, Circulant Digraph

The complexities of both TSP and BTSP on a circulant digraph (See definition in Chapter 11 ,Section 10.5) are open. An $O(n \log n)$ scheme for BTSP on symmetric circulant digraphs is given in [147] and we discuss it next.

Theorem 41 *[147] Let $\hat{G} = (N, \hat{E})$ be the undirected graph obtained from a symmetric circulant digraph $G(n, a_1, -a_1, a_2, -a_2, \ldots, a_m, -a_m)$, with $m < n$, by replacing every pair of arcs $\{(i,j), (j,i)\}$ by edge (i,j). Let $\gcd\{n, a_1, \ldots, a_m\} = g$. Then $\hat{G}$ has precisely g connected components $\hat{G}_1, \ldots, \hat{G}_g$ on node sets $N_i = \{i + kg : k \in \{0, \ldots, (n/g - 1)\}\}$, respectively, and each connected component is Hamiltonian.*

Proof. Let us prove the result by induction on m. For $m = 1$, let $\gcd\{n, a_1\} = g_1$. In this case, the result follows from elementary results in number theory [434] and for each $i \in \{1, \ldots, g\}$, the connected component $\hat{G}_i$ has a Hamiltonian tour $\gamma_i^1 = (i, (i+a_1) \bmod n, \ldots, (i+(n/g-1)a_1) \bmod n, i)$.

Suppose now that the result is true for all $m \leq u$ for some $u < n$. Let us prove the result for $m = u$. Let $\gcd\{n, a_1, \ldots, a_u\} = g_u$ and let $\gcd\{n, a_1, \ldots, a_{u-1}\} = g_{u-1}$. Let $n = rg_{u-1}$ and $g_{u-1} = kg_u$. By induction, $\hat{G}'$, the undirected graph corresponding to $G(n, a_1, -a_1, \ldots, a_{u-1}, -a_{u-1})$, has g_{u-1} connected components, each having a Hamiltonian tour. It follows from elementary number theory [434] that the nodes $x_1 = 1, x_2 = 1 + a_u, \ldots, x_k = 1 + (k-1)a_u$ belong to different connected components of $\hat{G}'$ which we denote by $\hat{G}'_1, \hat{G}'_2, \ldots, \hat{G}'_k$, respectively. Also, edges $\{(x_i, x_{i+1}) : i \in \{1, \ldots, k-1\}\}$ are in $\hat{G}$ and the k components $\hat{G}'_1, \ldots, \hat{G}'_k$ combine to form a connected component H_1 of H. Let $((1 =)i_0, i_1, \ldots, i_s, i_0)$ be a tour in $\hat{G}'_1$. Then, by symmetry, it follows that $(x_j, i_1 + x_j - 1, \ldots, i_s + x_j - 1, x_j)$ is a tour in $\hat{G}'_j$ for $j \in \{1, \ldots, k\}$ and edges $\{(x_j + i_v - 1, x_{j+1} + i_v - 1) : j \in \{1, \ldots, k-1\}, v \in \{0, \ldots, s\}\}$ are in $\hat{G}$. It is now easy to verify that the following process produces a tour in the $\hat{G}_1$. Remove edges $(i_s, 1)$ and $(i_s + x_2 - 1, x_2)$ and add edges $(1, x_2)$ and $(i_s, i_s + x_2 - 1)$; recursively, for $j = 2, \ldots, k-1$, remove edges $(x_j + i_{j-2} - 1, x_j + i_{(j-1)} - 1)$ and $(x_{j+1} + i_{j-2} - 1, x_{j+1} + i_{j-1} - 1)$ and add edges $(x_j + i_{j-2} - 1, x_{j+1} + i_{j-2} - 1)$ and $(x_j + i_{j-1} - 1, x_{j+1} + i_{j-1} - 1)$, where for any $j > s$, $i_j = i_{j \bmod s}$. Because of symmetry, similar method can be used to obtain a tour in every other connected component of H. ■

Now consider an instance BTSP(C) where for some symmetric circulant digraph $G(n, a_1, -a_1, a_2, -a_2, \ldots, a_m, -a_m)$ and numbers $d_1, \ldots, d_m$, the cost matrix C is defined as follows: for any $1 \leq i, j \leq n$, if $|i - j| = a_u \bmod n$ for some $u \in \{1, \ldots, m\}$, then $c_{ij} = d_u$; else, $c_{ij} = \infty$. The following $O(n \log n)$ scheme for this problem, based on Theorem 41, is given in [147]. Without loss of generality, let us assume that $d_1 \leq d_2 \leq \cdots \leq d_m$. For each $i = 1, \ldots, m$, find $\gcd\{n, a_1, \ldots, a_i\}$. All these *gcd* values can be calculated in $O(nlogn)$ time. Find minimum $u \in \{1, \ldots, m\}$ such that $\gcd\{n, a_1, \ldots, a_u\} = 1$. Find a tour γ^* in $G(n, a_1, -a_1, a_2, -a_2, \ldots, a_u, -a_u)$ as in the proof of Theorem 41. Then γ^* is the desired optimal solution.

4.6. BTSP on a Halin Graph

In this section we consider the BTSP restricted to a Halin graph. In Chapter 11, we have seen that TSP on a Halin graph with n nodes can be solved in $O(n)$ time. Using this algorithm within the binary search version of the threshold algorithm [269] for bottleneck problems, BTSP on a Halin graph can be solved in $O(n \log n)$ time. We now discuss an $O(n)$ algorithm to solve BTSP on a Halin graph [668].

Recall that a Halin graph G is obtained by embedding a tree T with no node of degree two in the plane and joining the leaf nodes by a cycle so that the resulting graph is planar. (see appendix A.) The nodes of G corresponding to the pendant nodes of T are called *outer nodes*. All other nodes are *internal nodes*. A Halin graph with exactly one internal node is called a *wheel*. It is easy to see that BTSP on a wheel can be solved in $O(n)$ time. As in the case of TSP on a Halin graph discussed in Chapter 11, we solve BTSP on a Halin graph by recursively collapsing fans. However, unlike the TSP, for BTSP we need an auxiliary objective function to record information generated during the iterations, and to maintain the invariant property of the objective function value in each iteration.

Let $\mathbb{F}$ be the family of all the Hamiltonian cycles (tours) in G. For each outer node i of G, let $e_{i,1}$, $e_{i,2}$ and $e_{i,3}$ be the edges incident to i. Any tour $\mathcal{H}$ in $\mathbb{F}$ uses precisely two of these three edges. Let $p(e_{i,1}, e_{i,2})$ be the 'penalty' incurred if $\mathcal{H}$ uses the edges $e_{i,1}$ and $e_{i,2}$; $p(e_{i,2}, e_{i,3})$ be the 'penalty' incurred if $\mathcal{H}$ uses the edges $e_{i,2}$ and $e_{i,3}$; and $p(e_{i,1}, e_{i,3})$ be the 'penalty' incurred if $\mathcal{H}$ uses the edges $e_{i,1}$ and $e_{i,3}$. Thus, for the tour $\mathcal{H}$ and outer node i, define

$$P_i(\mathcal{H}) = \begin{cases} p(e_{i,1}, e_{i,2}) & \text{if } e_{i,1} \text{ and } e_{i,2} \in \mathcal{H} \\ p(e_{i,2}, e_{i,3}) & \text{if } e_{i,2} \text{ and } e_{i,3} \in \mathcal{H} \\ p(e_{i,1}, e_{i,3}) & \text{if } e_{i,1} \text{ and } e_{i,3} \in \mathcal{H} \end{cases} \tag{8}$$

Consider the modified bottleneck TSP on a Halin graph defined as follows:

$$\begin{array}{ll} \text{MBTSP:} & \text{Minimize } f(\mathcal{H}) \\ & \text{Subject to} \\ & \qquad \mathcal{H} \in \mathbb{F}, \end{array}$$

where
$f(\mathcal{H}) = \max\{\max\{c_e | e \in \mathcal{H}\}, \max\{P_i(\mathcal{H}) | i \text{ is an outer node of } G\}\}$

Note that for each outer node i and tour $\mathcal{H}$ in $\mathbb{F}$, $P_i(\mathcal{H})$ can be identified in $O(1)$ time using the triplet $[p(e_{i,1}, e_{i,2}), p(e_{i,1}, e_{i,3}), p(e_{i,2}, e_{i,3})]$, called *penalty triplet*. At the beginning of the algorithm, we may choose $p(e_{i,1}, e_{i,2}) = p(e_{i,1}, e_{i,3}) = p(e_{i,2}, e_{i,3}) = 0$. Thus, with this choice of the penalty triplets, MBTSP is equivalent to BTSP. In our algorithm, we successively collapse fans into a pseudo-nodes and then update the values of the penalty triplets. Note that this fan collapsing scheme will eventually result in a wheel and hence we have to solve MBTSP on a wheel.

4.6.1 Algorithm for MBTSP on a wheel. Consider a wheel $W = (V, E)$ with u as its internal node. Let $1, 2, \ldots, n$ be the outer nodes such that $e_i = (i, i+1) \in E$, $i = 1, 2, \ldots, n$, where $n + 1 \equiv 1$. Consider the cycle $\Delta = (1, 2, \ldots, n, 1)$. Let e^* and e^{**} be edges of Δ such that

$$c_{e^*} = \max\{c_e \mid e \in \Delta\}$$

and

$$c_{e^{**}} = \max\{c_e \mid e \in \Delta - \{e^*\}\}.$$

Every tour on W is obtained by deleting from Δ some edge e_r and introducing edges (u, r) and $(u, r+1)$. We denote this tour by $\mathcal{H}_r$. Thus, $P_r(\mathcal{H}_r) = p(e_{r-1}, (u, r))$, $P_{r+1}(\mathcal{H}_r) = p(e_{r+1}, (u, r+1))$, and for every other $i : i \neq r, r+1$, $P_i(\mathcal{H}_r) = p(e_i, e_{i+1})$. Define

$$\delta_i = \max\{c_{(u,i)}, c_{(u,i+1)}, p(e_{i+1}, (u, i+1)), p(e_{i-1}, (u, i))\}.$$

Choose r, s, t such that

$$\begin{aligned} p(e_r, e_{r+1}) &= \max\{p(e_i, e_{i+1}) \mid e_i \in \Delta\} \\ p(e_s, e_{s+1}) &= \max\{p(e_i, e_{i+1}) \mid e_i \in \Delta, e_i \neq e_r\} \\ p(e_t, e_{t+1}) &= \max\{p(e_i, e_{i+1}) \mid e_i \in \Delta, e_i \neq e_r, e_s\}. \end{aligned}$$

(Here, $e_{n+1} = e_1$.) Now, given the values of $\delta_i, p(e_r, e_{r+1}), p(e_s, e_{s+1})$ and $p(e_t, e_{t+1})$, we can compute $f(\mathcal{H}_i)$, the objective function value of the tour $\mathcal{H}_i = \Delta - e_i + \{(u, i), (u, i+1)\}$ for MBTSP on W, in $O(1)$ time by considering 10 different cases [668].

Choose q such that $f(\mathcal{H}_q) = \min\{f(\mathcal{H}_i) | 1 \leq i \leq n\}$. Then the tour $\mathcal{H}_q$ is an optimal solution to MBTSP on W. Now

$$e^*, e^{**}, p(e_r, e_{r+1}), p(e_s, e_{s+1}), \text{ and } p(e_t, e_{t+1})$$

can be identified in $O(n)$ time. The quantity δ_i can be identified in $O(1)$ time for each $i = 1, 2, \ldots, n$. Thus, MBTSP on W can be solved in $O(n)$ time.

4.6.2 MBTSP on a Halin Graph. We now consider MBTSP on a Halin graph $G = (N, E)$ which is not a wheel. Note that such a graph has at least two fans. Let S be a fan of G with center u and outer nodes $u_1, u_2, \ldots, u_m$, where $(u_1, u_2, \ldots, u_m)$ is a path in S. The cutset $\{S, N - S\}$ contains precisely three edges $e_{S,1}, e_{S,2}$, and $e_{S,3}$ incident with nodes u_1, u, and u_m, respectively.

Any tour $\mathcal{H}$ in G uses exactly two of the three edges $e_{S,1}, e_{S,2}$, and $e_{S,3}$. If $\mathcal{H}$ uses $e_{S,1}$ and $e_{S,3}$, then $\mathcal{H} \cap S$ has the structure

$$(u_1, u_2, \ldots, u_j, u, u_{j+1}, \ldots, u_m) \tag{9}$$

If $\mathcal{H}$ uses $e_{S,1}$ and $e_{S,2}$, then $\mathcal{H} \cap S$ is of the form

$$(u, u_m, u_{m-1}, \ldots, u_2, u_1) \tag{10}$$

and if $\mathcal{H}$ uses $e_{S,2}$ and $e_{S,3}$, then $\mathcal{H} \cap S$ will be of the form

$$(u_m, u_{m-1}, \ldots, u_2, u_1, u) \tag{11}$$

Let $g(e_{S,2}, e_{S,3})$ be the contribution of S to the MBTSP objective function value of an optimal tour in G using edges $e_{S,2}$ and $e_{S,3}$. Then

$$g(e_{S,2}, e_{S,3}) = \max\{x_1, y_1\}$$

where

$$x_1 = \max\left\{\{c_{u_i,u_{i+1}} \mid 1 \leq i \leq m-1\}, c_{u,u_1}\right\}$$

and

$$\begin{aligned} y_1 = \max\{&\{p((u_i, u_{i+1}), (u_{i+1}, u_{i+2})) | 1 \leq i \leq m-2\}, \\ &p((u_{m-1}, u_m), e_{S,3}), p((u_1, u_2), (u, u_1)), p((u, u_1), e_{S,2})\}. \end{aligned}$$

Similarly, let $g(e_{S,1}, e_{S,2})$ be the contribution of S to the objective function value of an optimal tour in G using edges $e_{S,1}$ and $e_{S,2}$. Then

$$g(e_{S,1}, e_{S,2}) = \max\{x_2, y_2\}$$

where,

$$x_2 = \max\left\{\{c_{u_i,u_{i+1}} \mid 1 \leq i \leq m-1\}, c_{u,u_m}\right\}$$

and

$$\begin{aligned} y_2 = \max\{&\{p((u_i, u_{i+1}), (u_{i+1}, u_{i+2})), 1 \leq i \leq m-2\}, \\ &p((u_{m-1}, u_m), (u_m, u)), p((u_1, u_2), e_{S,1}), p((u, u_m), e_{S,2})\}. \end{aligned}$$

We similarly define $g(e_{S,1}, e_{S,3})$, the contribution of S to the objective function value of an optimal tour in G using edges $e_{S,1}$ and $e_{S,3}$. The quantities $g(e_{S,1}, e_{S,2}), g(e_{S,1}, e_{S,3})$, and $g(e_{S,2}, e_{S,3})$ can be computed in $O(p)$ time by appropriate modifications of the procedure for solving MBTSP on a wheel.

Consider the Halin graph $G \times S$ obtained from G by collapsing S into a pseudo-node $\hat{v}$. Let $e_{\hat{v},1}, e_{\hat{v},2}, e_{\hat{v},3}$ be the edges in $G \times S$ incident on node $\hat{v}$ which correspond to edges $e_{S,1}, e_{S,2}, e_{S,3}$, respectively.

Assign to the penalty triplet $[p(e_{\hat{v},1}, e_{\hat{v},2}), p(e_{\hat{v},1}, e_{\hat{v},3}), p(e_{\hat{v},2}, e_{\hat{v},3})]$ the values $[g(e_{S,1}, e_{S,2}), g(e_{S,1}, e_{S,3}), g(e_{S,2}, e_{S,3})]$, respectively. Further, let $c_{e_{\hat{v},1}} = c_{e_{S,1}}, c_{e_{\hat{v},2}} = c_{e_{S,2}}$, and $c_{e_{\hat{v},3}} = c_{e_{S,3}}$.

The optimal objective function value of MBTSP on G and the optimal objective function value of the corresponding instance of MBTSP on $G \times S$ are the same. Note that $G \times S$ is also a Halin graph. If it is a wheel then MBTSP on the wheel can be solved using the algorithm discussed earlier. If it is not a wheel, then it contains at least two fans and the fan collapsing scheme can be continued and eventually we reach a wheel. Backtracking from the optimal solution on this wheel, an optimal solution to MBTSP (and hence to BTSP) on G can be constructed by expanding the pseudo-nodes. We leave it to the reader to verify that the procedure can be implemented in $O(n)$ time.

5. Variations of the Bottleneck TSP

Let us now discuss some variations of the BTSP that subsume both TSP and BTSP. We first consider traveling salesman problem under categorization [678]. Let $S_1, S_2, \ldots, S_p$ be a partition of the edge set E of the graph (digraph) $G = (N, E)$ and let $\mathbb{F}$ be the family of all Hamiltonian cycles in G. Let c_e be the cost of edge $e \in E$. For any $\mathcal{H} \in \mathbb{F}$, define

$$f_1(\mathcal{H}, C) = \max_{1 \le i \le p} \sum_{e \in \mathcal{H} \cap S_i} c_e$$

and

$$f_2(\mathcal{H}, C) = \sum_{1}^{p} \max_{e \in \mathcal{H} \cap S_i} c_e$$

where summation over the empty set yields a value $-\infty$ and maximum over the emplty set yields zero.

There are two versions of the traveling salesman problem under categorization which are denoted by TSPC-1 and TSPC-2. TSPC-1 seeks a Hamiltonian cycle $\mathcal{H}'$ in G such that $f_1(\mathcal{H}', C)$ is as small as possible, whereas TSPC-2 attempts find a Hamiltonian cycle of G that minimizes $f_2(\mathcal{H}, C)$. For $p = 1$, TSPC-1 reduces to the TSP and TSPC-2 reduces to the BTSP. For $p = |E|$, TSPC-1 reduces to the BTSP and TSPC-2 reduces to the TSP. Thus TSPC-1 and TSPC-2 are proper generalizations of both TSP and BTSP and hence are NP-hard. We have seen that BTSP and TSP can be solved in linear time on Halin graph. However, it is possible to show that both TSPC-1 and TSPC-2 are strongly NP-hard on Halin graphs [678]. However, for fixed p, TSPC-2 is solvable in polynomial time on a Halin graph whereas TSPC-1 remains NP-hard

on Halin graphs even for $p = 2$ [678]. Similar generalizations of MAX TSP and MSTSP can be constructed.

Suppose that a Hamiltonian cycle represents the order in which n jobs are to be processed sequentially in an assembly line. Then the objective function of the TSP measures the total processing time. Under this model, each of TSPC-1 and TSPC-2 can be interpreted as the problem of minimizing the completion time of jobs in an assembly line when there is some series-parallel order relation for processing the jobs. More precisely, if jobs corresponding to the same subset are to be processed sequentially, where as different subsets can be processed in parallel, then the TSPC-1 objective function measures the total processing time. If jobs corresponding to the same subsets can be done in parallel, but a new subset of jobs can be taken only after processing all jobs from the current set, TSPC-2 objective function measures the total processing time.

Another problem that simultaneously generalizes BTSP and TSP is the k-sum traveling salesman problem (k-sum TSP) [411, 681, 802]. In a k-sum TSP, one seeks a Hamiltonian cycle $\mathcal{H}$ of a graph (digraph) G on n nodes where the sum of the k largest edge-weights of $\mathcal{H}$ is as small as possible. For $k = n$, k-sum TSP reduces to the TSP and for $k = 1$ it reduces to the BTSP. The k-sum TSP can be solved by solving $O(m)$ TSP's where m is the number of edges in G.

Acknowledgements: We are thankful to P. Sharma and P. Toth for their comments on an earlier version of this chapter. This work was partially supported by the NSERC grant OPG0008085 awarded to S.N. Kabadi and the NSERC grant OPG0170381 awarded to A.P. Punnen.