An Approximation Algorithm for a Bottleneck Traveling Salesman Problem*

Ming-Yang Kao † Manan Sanghi ‡

Abstract

Consider a truck running along a road. It picks up a load L_i at point β_i and delivers it at α_i, carrying at most one load at a time. The speed on the various parts of the road in one direction is given by $f(x)$ and that in the other direction is given by $g(x)$. Minimizing the total time spent to deliver loads $L_1, \ldots, L_n$ is equivalent to solving the Traveling Salesman Problem (TSP) where the cities correspond to the loads L_i with coordinates (α_i, β_i) and the distance from L_i to L_j is given by $\int_{\alpha_i}^{\beta_j} f(x)dx$ if $\beta_j \geq \alpha_i$ and by $\int_{\beta_j}^{\alpha_i} g(x)dx$ if $\beta_j < \alpha_i$. This case of TSP is polynomially solvable with significant real-world applications.

Gilmore and Gomory obtained a polynomial time solution for this TSP [5]. However, the bottleneck version of the problem (BTSP) was left open. Recently, Vairaktarakis showed that BTSP with this distance metric is NP-complete [9].

We provide an approximation algorithm for this BTSP by exploiting the underlying geometry in a novel fashion. This also allows for an alternate analysis of Gilmore and Gomory's polynomial time algorithm for the TSP. We achieve an approximation ratio of $(2 + \gamma)$ where $\gamma \geq \frac{f(x)}{g(x)} \geq \frac{1}{\gamma}$ $\forall x$. Note that when $f(x) = g(x)$, the approximation ratio is 3.

1 Introduction

Consider n cities $C_1, C_2, \ldots, C_n$. Let c_{ij} be the distance from C_i to C_j. The problem of finding a tour that visits each city exactly once and minimizes the total travel distance is known as the traveling salesman problem (TSP). The bottleneck traveling salesman problem (BTSP) is to find a tour that visits each city exactly once and minimizes the maximum distance traveled between any two adjacent cities on the tour. Both the TSP and the BTSP are NP-hard in general [4]. We consider the distance metric first proposed by Gilmore and Gomory in [5] which has widespread practical applications [2, 6, 8, 10] and for which the TSP is polynomial time solvable. Unfortunately, the BTSP remains NP-hard for this distance metric [9].

Let each city C_i be specified by the coordinates (α_i, β_i) for $i = 1, 2, \ldots, n$. The distance metric considered in this paper is given by:

$$d(C_i, C_j) = c(\alpha_i, \beta_j) = \begin{cases} \int_{\alpha_i}^{\beta_j} f(x)dx & \text{if } \beta_j \geq \alpha_i \\ \int_{\beta_j}^{\alpha_i} g(x)dx & \text{if } \beta_j < \alpha_i \end{cases}$$

where $f(\cdot)$ and $g(\cdot)$ are integrable and $f(x), g(x) \geq 0$. Note that if $f(x) = g(x) = 1$, then $c(\alpha_i, \beta_j) = |\alpha_i - \beta_j|$. Also note that in [5], Gilmore and Gomory solve the TSP with a less restrictive condition viz. $f(x) + g(x) \geq 0$ for all x.

*Supported in part by NSF Grant EIA-0112934.

†Department of Computer Science, Northwestern University. Email: kao@cs.northwestern.edu.

‡Department of Computer Science, Northwestern University. Email: manan@cs.northwestern.edu.

Problem 1 (Gilmore-Gomory's Traveling Salesman Problem (GG-TSP)).
INPUT: *n pairs of numbers* $(\alpha_0, \beta_0), (\alpha_1, \beta_1), \ldots, (\alpha_{n-1}, \beta_{n-1})$.
OUTPUT: *A permutation* $\pi : \{0, \ldots, n-1\} \to \{0, \ldots, n-1\}$ *such that* $\sum_{i=0}^{n-1} c(\alpha_{\pi(i+1 \mod n)}, \beta_{\pi(i)})$ *is minimized.*

Problem 2 (Gilmore-Gomory's Bottleneck Traveling Salesman Problem (GG-BTSP)).
INPUT: *n pairs of numbers* $(\alpha_0, \beta_0), (\alpha_1, \beta_1), \ldots, (\alpha_{n-1}, \beta_{n-1})$.
OUTPUT: *A permutation* $\pi : \{0, \ldots, n-1\} \to \{0, \ldots, n-1\}$ *such that* $\max_{i=0}^{n-1} c(\alpha_{\pi(i+1 \mod n)}, \beta_{\pi(i)})$ *is minimized.*

Results GG-TSP can be solved in $O(n \log n)$ time [5, 10]. GG-BTSP can also be solved in $O(n \log n)$ time if either $f(x) = 0$ or $g(x) = 0$ [5, 10]. However, in general GG-BTSP is NP-hard [9]. In fact, the reduction used in [9] proves NP-hardness for the special case when $f(x) = g(x) = 1$.

In this paper, we give a $(2 + \gamma)$-approximation algorithm for GG-BTSP where $\gamma \geq \frac{f(x)}{g(x)} \geq \frac{1}{\gamma}$ $\forall x$. Note that this result immediately implies the following:

1. a 3-approximation algorithm when $c(\alpha_i, \beta_j) = |\alpha_i - \beta_j|$.
2. a $(2 + \max\{\frac{b}{a}, \frac{a}{b}\})$-approximation algorithm when $f(x) = a$ and $g(x) = b$.
3. a 3-approximation algorithm when $f(x) = g(x)$.
4. a $(2 + \frac{b}{a})$-approximation algorithm when $a \leq f(x), g(x) \leq b$.

Further, we uncover some interesting properties of the underlying geometry of the problem that shed new light on the structure of an optimal solution and hence allows for an alternate analysis of the polynomial time TSP algorithm presented in [5, 10].

Paper Layout Section 2 discusses some applications of GG-TSP and GG-BTSP. Section 3 formulates an equivalent problem of GG-BTSP, called BBCA, on bipartite graphs and defines some concepts and notations used in the paper. Section 4 derives a lower bound on the optimum bottleneck cost. Finally, Section 5 presents the approximation algorithm for GG-BTSP. Most of the proofs omitted from the main paper in the interest of space are presented in Appendix **??**.

2 Applications

The original motivation for the formulation of GG-TSP and GG-BTSP was job sequencing on a single state variable machine [5]. Consider a furnace and let temperature be its state variable. A number of jobs are to be given a heat treatment in the furnace. The i^{th} job will be started at temperature β_i and taken out of the furnace at temperature α_i. The temperature is then changed for the next job. Heating the furnace requires $f(x)$ amount of energy while cooling requires $g(x)$ when the temperature is x. The furnace is at temperature α_0 to start with and is required to be in state β_0 at the end. Sequencing the jobs to minimize the total energy is equivalent to GG-TSP. Sequencing to minimize the maximum energy required for changing between two jobs is equivalent to GG-BTSP.

Another application of the GG-TSP is in minimizing makespan in a two-machine flowshop with no-wait-in-process which is a building block for more general no-wait production systems [8].

Ball et al [2] consider an interesting special case of GG-TSP that arises in the context of optimal insertion of chips on a printed circuit board.

Another application of this problem formulation is in reconstructing sequential order from inaccurate adjacency information. Consider n women standing in a circle with each facing the clockwise direction. Each woman reports her own height α_i, and the height β_i of the one in front of her. Given this information, we want to reconstruct the order of the women in the circle. When the women make some errors in estimating heights, we may want to construct an ordering which minimizes the maximum of the differences between the height α_j reported by the j^{th} woman from the height β_i reported by the i^{th} woman for each pair of women i and j such that j is in front of i in the ordering. This problem is equivalent to GG-BTSP with $f(x) = g(x) = 1$.

One practical field where the problem of reconstructing such sequential order arises naturally is in interpreting nuclear magnetic resonance (NMR) spectroscopy data for solving a NMR protein structure. When exposed to an oscillating radio frequency field, the individual nuclei in a protein sample respond at specific resonance frequencies, called *chemical shifts*. These chemical shifts serve as identifiers of the corresponding atoms. The data from NMR experiments consists of spectral peaks where a peak can correspond to a pair of chemical shifts of atoms in adjacent amino acids on the protein backbone. The goal is to determine the correct order of these chemical shifts from such adjacency information provided in NMR spectral data. For NMR data interpretation, corresponding to the women in the aforementioned example we have spectral peaks, and corresponding to the pairs of reported heights we have pairs of chemical shifts associated with each peak. Some good references for extracting the adjacency information from NMR experiments can be found in [11, 12]. There is also extensive work in automatic resonance frequency assignment algorithms [1, 3, 7, 11–13].

Note that though we discuss the problem in terms of reconstructing a circular order, the transformation to reconstructing linear order, as is required for NMR data interpretation, is achieved in polynomial time. If the first and the last element in the linear order are known, the linear order problem can be reduced to GG-BTSP in linear time by assigning the first element and the front neighbor of the last element the identifier ∞, i.e., if f is the index for the first element and ℓ is the index for the last one, then $\alpha_f = \beta_\ell = \infty$. This forces a minimum cost circular order to place the last element before the first one. If the first and the last element are not known, then there are $2\binom{n}{2}$ options for them and hence if the time complexity of GG-BTSP is T, we can solve this linear order problem in $O(n^2T)$ time. Similarly, if either the first element or the last element is given, then we can solve the linear order problem in $O(nT)$ time. In this paper, we provide an approximation algorithm for GG-BTSP with a runtime of $O(n \log n)$; i.e., $T = O(n \log n)$.

3 Preliminaries

We first define an equivalent problem of GG-BTSP on bipartite graphs in Section 3.1. The rest of the paper focusses on solving this equivalent problem. Then we define some notations in Section 3.2, discuss some concepts in Section 3.3 and present basic lemmas in Section 3.4.

3.1 Problem Definition

Problem 3 (Bottleneck Bipartite Cyclic Augmentation (BBCA)).
INPUT*: A bipartite graph $G = (U, V, H)$ where H is a perfect matching, and a function $\phi : U \cup V \to \mathbb{R}$.*

OUTPUT: *A set of edges M such that the bipartite graph $G' = (U, V, H \cup M)$ is a hamiltonian cycle and* $\max_{(u,v)\in M}\ c(\phi(u), \phi(v))$ *is minimized.*

For $w \in U \cup V$, $\phi(w)$ is called the *potential* of w. The *cost* of an edge (u, v) where $u \in U$ and $v \in V$ is given by $c((u, v)) = c(\phi(u), \phi(v))$. The *cost* of a matching M is given by $c_M = \max_{e\in M} c(e)$. A set of edges M such that $G' = (U, V, H \cup M)$ is a cycle is called a *cyclic augmentation* of $G = (U, V, H)$.

Lemma 1. *GG-BTSP and BBCA can be reduced to each other in linear time.*

Proof. For reducing GG-BTSP to BBCA, given an instance I of GG-BTSP, (α_0, β_0), (α_1, β_1), ..., $(\alpha_{n-1}, \beta_{n-1})$, let $U = \{u'_0, u'_1, \ldots, u'_{n-1}\}$, $V = \{v'_0, v'_1, \ldots, v'_{n-1}\}$, $\phi(u'_i) = \alpha_i$, $\phi(v'_j) = \beta_j$, $H = \{(u'_0, v'_0),\ (u'_1, v'_1),\ \ldots,\ (u'_{n-1}, v'_{n-1})\}$, to form an instance I' of BBCA. Conversely, an instance I of GG-BTSP can be similarly derived from an instance I' of BBCA.

Next, as shown below, there is a 1-1 correspondence between a cyclic augmentation for I' and a permutation for I, and the costs of the two are equal. Therefore, a minimum cost cyclic augmentation for I' will be a minimum cost permutation for I.

Given a cyclic augmentation M for I', we can construct a permutation π of I as follows. Let $\pi(0) = 0$. If $\pi(k) = i$ and $(u'_j, v'_i) \in M$, then $\pi(k + 1 \mod n) = j$. Since M is a cyclic augmentation, π is a permutation.

Similarly, given a permutation π for I, the corresponding cyclic augmentation for I' is given by

$$M = \{(u'_j, v'_i) \mid \exists\ k \text{ such that } \pi(k) = i, \pi(k + 1 \bmod n) = j\}.$$

□

3.2 Notations

For the remainder of the paper, let $u_0, u_1, \ldots, u_{n-1}$ be the n vertices in U such that $\phi(u_0) \leq \phi(u_1) \leq \cdots \leq \phi(u_{n-1})$. Similarly, let $v_0, v_1, \ldots, v_{n-1}$ be the n vertices in V such that $\phi(v_0) \leq \cdots \leq \phi(v_{n-1})$.

If M is a matching between U and V, then let G_M denote the graph $G(U, V, H \cup M)$. Note that for any matching M, G_M is a set of simple cycles. If G_M contains exactly one cycle, then M is a cyclic augmentation. For $g \in U \cup V$, let e_g^M denote the edge adjacent to vertex g in M.

3.3 Concepts

It is useful to visualize the vertices in U as being arranged on the horizontal axis with their abscissa being the potential $\phi(u_i)$. Similarly the vertices in V can be visualized as being at a higher ordinate and with their abscissa being their corresponding potential. An edge (u_i, v_j) is a straight line connecting $\phi(u_i)$ and $\phi(v_j)$. See Figure 1 for an example visualization.

In our figures, we will represent the edges in H by dashed lines and the edges in M by solid lines.

Left, Right and In-between Edges For any three edges $e_1 = (u_a, v_b)$, $e_2 = (u_c, v_d)$ and $e_3 = (u_p, v_q)$. If $a < c$, then e_1 is said to be on the *left* and e_2 is said to be on the *right*. The edge e_2 is said to be *in-between* e_1 and e_3 if $a < c < p$ and $b < d < q$. Let η_{e_1,e_2} be the number of edges in-between e_1 and e_2.

$$\eta_{e_1,e_2} = |\ \{\ (u_r, v_s) \mid a < r < c, b < s < d\}\ |.$$

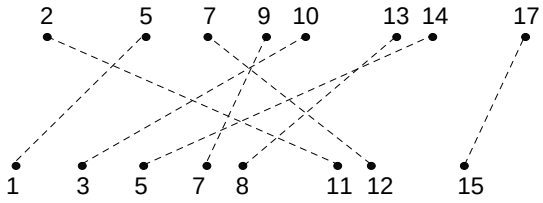

Figure 1: A visual representation of $G(U, V, H)$ where the potentials of the vertices in U are $1, 3, 5, 7, 8, 11, 12, 15$, the potentials of the vertices in V are $2, 5, 7, 9, 10, 13, 14, 17$, and H consists of 8 edges connecting the potential pairs $(1, 5), (3, 10), (5, 14), (7, 9), (8, 13), (11, 2), (12, 7)$ and $(15, 17)$.

Cross State and Straight State Given two edges (u_a, v_b) and (u_c, v_d) such that $a < c$, the edges are said to be in a *straight state* if $b < d$ and in a *cross state* if $b > d$. In our visualization, the edges in a cross state will intersect while those in a straight state will not (see Figure 2). An edge e_1 is said to *cross* e_2 if e_1 and e_2 are in a cross state. Note that if e_1 crosses e_2, then $\eta_{e_1,e_2} = 0$.

Cross Number The *cross number* of a matching M, denoted by Γ_M, is the number of pairs of edges which are in a cross state in M. Observe that

$$\Gamma_M = \frac{|\{(g, h) \mid g, h \in U \text{ and } e_g^M \text{ crosses } e_h^M\}|}{2}.$$

Note that the cross number of M is the number of intersections in its visualization (see Figure 2 for an example).

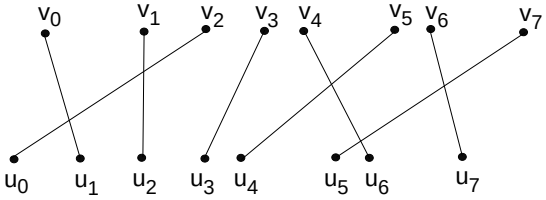

Figure 2: The cross number of this matching M is 5. The edges $e_{u_2}^M$ and $e_{u_3}^M$ are in a straight state. The edges $e_{u_5}^M$ and $e_{u_7}^M$ are in a cross state.

Exchange Given a matching M with two edges $e_1 = (u_a, v_b)$ and $e_2 = (u_c, v_d)$, an *exchange* on e_1 and e_2 returns a matching M' such that $M' = M \otimes (e_1, e_2) = (M \setminus \{e_1, e_2\}) \cup \{(u_a, v_d), (u_c, v_b)\}$. Note that if e_1 and e_2 are in a straight state in M, then their replacement edges are in a cross state in M'. Such exchanges are called *straight-to-cross* exchanges. Similarly, *cross-to-straight* exchanges are the ones on two edges in a cross state to result in two in a straight state. A *null exchange* on M is defined to be the operation which returns matching M.

Direct Pair The set of vertices $\{u_i, v_i\}$ is called the i^{th} *direct pair*. For any i $(0 \leq i \leq n-2)$, the i^{th} and the $(i+1)^{th}$ direct pairs are said to be *consecutive*.

Let $M_D = \{(u_0, v_0), (u_1, v_1), ..., (u_{n-1}, v_{n-1})\}$. Note that M_D is a matching with the minimum cost over all possible matchings. However, M_D may not be a cyclic augmentation.

Cluster A *cluster* is the union of consecutive direct pairs which belong to the same cycle in G_{M_D}. The i^{th} cluster from the left is denoted by ψ_i. Therefore, ψ_1 is the cluster containing the leftmost direct pair and ψ_j is the cluster containing the leftmost direct pair in $U \cup V \setminus (\psi_1 \cup \cdots \cup \psi_{j-1})$.

Note that the clusters define a partition of $U \cup V$. All the vertices in a cluster belong to the same cycle in G_{M_D} but all the vertices in the same cycle in G_{M_D} need not be in the same cluster. See Figure 3 for an illustration of clusters.

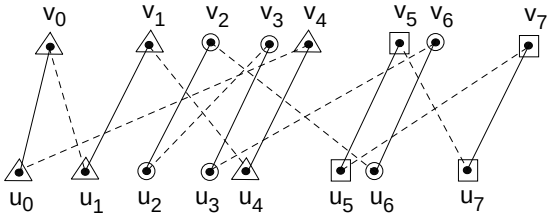

Figure 3: For this graph, G_{M_D} has three cycles $C_1 = u_0v_0u_1v_1u_4v_4$, $C_2 = u_2v_2u_6v_6u_3v_3$ and $C_3 = u_5v_5u_7v_7$. The vertices in C_1 are marked by a triangle, the ones in C_2 by a circle and those in C_3 by a square. Therefore, the clusters are $\psi_1 = \{u_0, v_0, u_1, v_1\}$, $\psi_2 = \{u_2, v_2, u_3, v_3\}$, $\psi_3 = \{u_4, v_4\}$, $\psi_4 = \{u_5, v_5\}$, $\psi_5 = \{u_6, v_6\}$, and $\psi_6 = \{u_7, v_7\}$.

Exchange Graph The *exchange graph* $\mathcal{X}$ for $G = (U, V, H)$ is a multigraph whose vertices correspond to the cycles in G_{M_D}. There is an edge between two cycles C and C' for every pair of consecutive clusters ψ_i and ψ_{i+1} such that ψ_i has vertices in C and ψ_{i+1} has vertices in C'. The *weight* of this edge is $\max\{c(\phi(u^{\psi_i}_{\text{right}}), \phi(v^{\psi_{i+1}}_{\text{left}})), c(\phi(u^{\psi_{i+1}}_{\text{left}}), \phi(v^{\psi_i}_{\text{right}}))\}$, where $(u^{\psi_i}_{\text{right}}, v^{\psi_i}_{\text{right}})$ are the rightmost direct pair in ψ_i and $(u^{\psi_{i+1}}_{\text{left}}, v^{\psi_{i+1}}_{\text{left}})$ are the leftmost direct pair in ψ_{i+1}. If $(u^{\psi_i}_{\text{right}}, v^{\psi_i}_{\text{right}}) = (u_k, v_k)$, then the *label* of the corresponding edge is $(k, k+1)$.

3.4 Lemmas

Lemma 2. *Given matchings M and M' between U and V, there exist a sequence of exchanges $x_0, x_1, \ldots, x_m$ for $m < n$ which transforms M to M'.*

Proof. We will construct a sequence of exchanges $x_0, \ldots, x_{n-1}$ which transforms M to M'. Let $M_k = M_D \otimes x_0 \otimes \cdots \otimes x_k$. A vertex y in M_k is said to be *satisfied* if its adjacent vertex in M_k is the same as its adjacent vertex in M'. If u_0 is satisfied, then x_0 is a null exchange. Else if the vertex adjacent to u_0 in M' is adjacent to u_i in M, then let x_0 be the exchange between $e^M_{u_0}$ and $e^M_{u_i}$. Now u_0 is satisfied in M_0. We can repeat the same process iteratively so that $u_0, u_1, \ldots, u_k$ are satisfied in M_k. Therefore, $M_{n-1} = M'$ and the exchanges $x_0, \ldots, x_{n-1}$ transforms M to M'. □

In Lemmas 3 through 6 below, for any two edges $e_1 = (u_a, v_b)$ and $e_2 = (u_c, v_d)$ such that $e_1, e_2 \in M$ and $a < c$, let $M' = M \otimes (e_1, e_2)$.

Lemma 3. *If e_1 and e_2 are in a straight state, then $c_{M'} \geq c_M$.*

Proof. We have the following 6 cases (see Figure 4):
Case 1: $\phi(v_d) \leq \phi(u_a)$. $c((u_c, v_b)) \geq \max\{c((u_a, v_b)), c((u_c, v_d))\}$.
Case 2: $\phi(v_b) \leq \phi(u_a), \phi(u_a) \leq \phi(v_d) < \phi(u_c)$. $c((u_c, v_b)) \geq \max\{c((u_a, v_b)), c((u_c, v_d))\}$.
Case 3: $\phi(v_b) \leq \phi(u_a), \phi(u_c) \leq \phi(v_d)$. $c((u_c, v_b)) \geq c((u_a, v_b))$ and $c((u_a, v_d)) \geq c((u_c, v_d))$.
Case 4: $\phi(u_a) \leq \phi(v_b) < \phi(u_c), \phi(u_a) \leq \phi(v_d) < \phi(u_c)$. $c((u_a, v_d)) \geq c((u_a, v_b))$ and $c((u_c, v_b)) \geq c((u_c, v_d))$.
Case 5: $\phi(u_a) \leq \phi(v_b) < \phi(u_c), \phi(u_c) \leq \phi(v_d)$. $c((u_a, v_d)) \geq \max\{c((u_a, v_b)), c((u_c, v_d))\}$.
Case 6: $\phi(u_c) \leq \phi(v_b)$. $c((u_a, v_d)) \geq \max\{c((u_a, v_b)), c((u_c, v_d))\}$. □

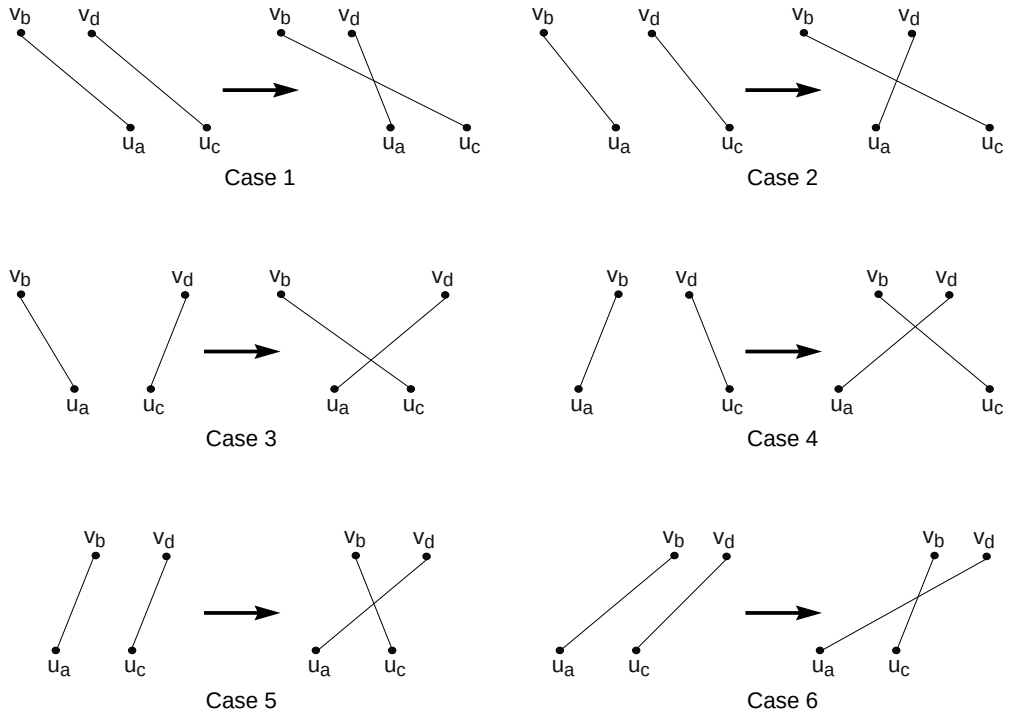

Figure 4: Case analysis for Lemma 3.

Lemma 4.

1. *If e_1 and e_2 are in the same cycle in G_M, then their replacement edges are in different cycles in $G_{M'}$.*

2. *If e_1 and e_2 are in different cycles in G_M, then their replacement edges are in a same cycle in $G_{M'}$.*

Proof. Statement 1: Let C be the cycle containing e_1 and e_2. Note that because G_M is bipartite, the path from u_a to u_c goes through either v_b or v_d but not both (see Figure 5(a)). $G_{M'}$ contains edges (u_a, v_d) and (u_c, v_b) in place of (u_a, v_b) and (u_c, v_d) and therefore $e_{u_a}^{M'}$ is in a different cycle from $e_{u_c}^{M'}$.
Statement 2: See Figure 5(b). □

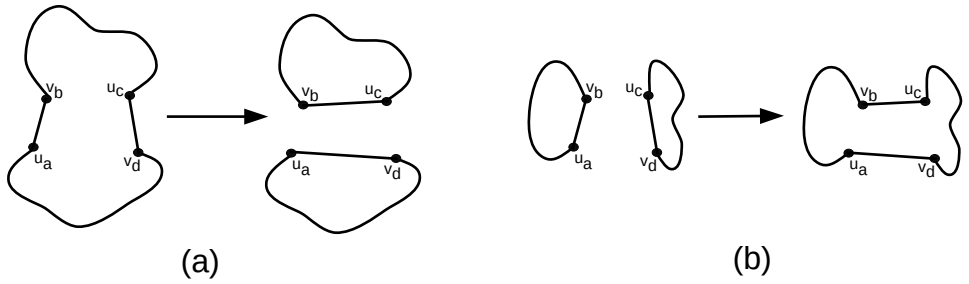

Figure 5: (a) An exchange between edges in a same cycle splits the cycle in two. (b) An exchange between edges in two different cycles joins the cycles.

Lemma 5. *If e_1 crosses e_2 and for some $u_p \in U$, $e^{M'}_{u_p}$ crosses $e^{M'}_{u_a}$, then $e^{M}_{u_p}$ crosses at least one of e_1 and e_2. By symmetry, if e_1 crosses e_2 and for some $u_p \in U$, $e^{M'}_{u_p}$ crosses $e^{M'}_{u_c}$, then $e^{M}_{u_p}$ crosses at least one of e_1 and e_2.*

Proof. Note that $e^{M'}_{u_a} = (u_a, v_d)$, $e^{M'}_{u_c} = (u_c, v_b)$ and $d < b$. Let $e^{M'}_{u_p} = e^{M}_{u_p} = (u_p, v_q)$. If $e^{M'}_{u_p}$ crosses $e^{M'}_{u_a}$, then we have two possible cases:
Case 1: $p < a$ and $q > d$. In this case $e^{M}_{u_p}$ crosses $e^{M}_{u_c}$.
Case 2: $p > a$ and $q < d$. In this case $e^{M}_{u_p}$ crosses $e^{M}_{u_a}$. □

Lemma 6.

1. *If e_1 and e_2 are in a straight state, then $\Gamma_{M'} = \Gamma_M + 1 + 2\eta_{e_1,e_2}$.*
2. *If e_1 and e_2 are in a cross state, then $\Gamma_{M'} = \Gamma_M - 1 - 2\eta_{e^{M'}_{u_a}, e^{M'}_{u_c}}$.*

Proof. Statement 1: To determine the difference in the cross numbers for M and M', we will consider the change in contribution to the cross number for each pair of vertices u_p and u_q in U.

If u_p and u_q are different from u_a and u_c, their incident edges cross in M' if and only if they cross in M. Hence, the contribution of (u_p, u_q) to the cross number is unchanged.

If u_p and u_q are u_a and u_c, then e_1 does not cross e_2 while $e^{M'}_{u_a}$ does cross $e^{M'}_{u_c}$. Hence, $\Gamma_{M'}$ increases by one due to (u_a, u_c).

Otherwise, we may assume without loss of generality that u_p is different from u_a and u_c but u_q is either u_a or u_c. We consider the total contribution of (u_p, u_a) and (u_p, u_c) to the cross number as follows.

If $e^{M}_{u_p}$ is in-between e_1 and e_2, then while $e^{M}_{u_p}$ crosses neither $e^{M}_{u_a}$ nor $e^{M}_{u_c}$, $e^{M'}_{u_p}$ crosses both $e^{M'}_{u_a}$ and $e^{M'}_{u_c}$. Since there are η_{e_1,e_2} edges in-between e_1 and e_2, $\Gamma_{M'}$ increases by $2\eta_{e_1,e_2}$ due to (u_p, u_a) and (u_p, u_c).

If $e^{M}_{u_p}$ crosses neither e_1 nor e_2 and $e^{M}_{u_p}$ is not in-between e_1 and e_2, then $e^{M'}_{u_p}$ also crosses neither $e^{M'}_{u_a}$ nor $e^{M'}_{u_c}$. Hence, the cross number does not change due to (u_p, u_a) and (u_p, u_c).

If $e^{M}_{u_p}$ crosses both $e^{M}_{u_a}$ and $e^{M}_{u_c}$, then $e^{M'}_{u_p}$ also crosses both $e^{M'}_{u_a}$ and $e^{M'}_{u_c}$. Hence, the cross number does not change due to (u_p, u_a) and (u_p, u_c).

If $e^{M}_{u_p}$ crosses exactly one of e_1 and e_2, then $e^{M'}_{u_p}$ also crosses exactly one of $e^{M'}_{u_a}$ and $e^{M'}_{u_c}$. Hence, the cross number does not change due to (u_p, u_a) and (u_p, u_c).
Statement 2: This follows immediately from Statement 1. □

4 Lower Bound on the Optimum Bottleneck Cyclic Augmentation

As observed in Section 3.2, for any perfect matching M between U and V, the graph G_M is a collection of simple cycles. Note that M_D is the minimum cost matching of $G(U, V, H)$. However, M_D may not be a cyclic augmentation i.e., $H \cup M_D$ may not be a hamiltonian cycle. Our strategy for solving BBCA is to begin with G_{M_D} and transform M_D into a cyclic augmentation by means of exchanges.

Recall from Lemma 4(2) that an exchange between two edges in different cycles, say C_1 and C_2, yields a graph in which all the vertices in C_1 and C_2 are in one cycle (see Figure 5). Alternately, from Lemma 4(1), an exchange between two edges in the same cycle yields a graph in which the vertices in that cycle are split into two distinct cycles. Furthermore, from Lemma 2, we know that for any two matchings M and M', M can be converted to M' by a sequence of exchanges. In this section we present Lemma 7 which identifies some useful properties of a minimum cost cyclic augmentation which allows us to restrict the search space for suitable exchanges to convert M_D to an approximately optimal cyclic augmentation. Then, using Lemma 8, we reduce our search space to exchanges corresponding to the edges in the exchange graph $\mathcal{X}$. As will be shown in Lemma 9, this allows us to derive a good lower bound on the cost of the optimal cyclic augmentation.

Lemma 7. *There exists a minimum bottleneck cost cyclic augmentation M^* for $G = (U, V, H)$ such that the following properties hold true:*
(P_1) Any edge $e \in M^*$ crosses either some edges on its left or some on its right but not both.
(P_2) For $e_1, e_2, e_3 \in M^*$, if e_1 crosses e_2 and e_3, then no other edge in M^* crosses both e_2 and e_3.
(P_3) If two vertices $u_p, u_q \in U$ are in the same cycle in G_{M_D}, then $e_{u_p}^{M^*}$ and $e_{u_q}^{M^*}$ do not cross.
(P_4) If two vertices $u_p, u_q \in U$ are in the same cycle in G_{M_D} and u_p is on the left of u_q, then

1. *$e_{u_p}^{M^*}$ cannot cross any edge to the right of $e_{u_q}^{M^*}$; and*
2. *$e_{u_q}^{M^*}$ cannot cross any edge to the left of $e_{u_p}^{M^*}$.*

Proof. The proof is by construction. Given a minimum cost cyclic augmentation M', we show that it can be transformed to a minimum cost cyclic augmentation M^* which satisfies the above 4 properties.

For each property P_i, given the smallest set of vertices $W \subseteq U \cup V$ for which P_i does not hold in M', we give a transformation T_i for constructing a new matching of cost no more than that of M' and a cross number smaller than that of M'. The algorithm for the construction begins with any minimum cost cyclic augmentation and repeatedly finds the smallest i such that P_i does not hold. Use T_i to correct this violation till a matching for which all the properties hold true is obtained. For the correctness and termination of this algorithm, we ensure that each of the transformations T_i satisfies the following two conditions. Assuming that P_j holds for all $j < i$, given any cyclic augmentation M' and the smallest set of vertices $W \in M'$ such that the edges incident to W do not satisfy P_i, $T_i(M', W)$ returns a matching M'' such that

1. M'' is a cyclic augmentation of cost no more than that of M'; and
2. $\Gamma_{M''} < \Gamma_{M'}$.

The first condition above ensures that after every transformation, we get a cyclic augmentation of the minimum cost. The second condition ensures that the total number of crosses decreases monotonically. Hence, we terminate either with a minimum cost cyclic augmentation which either satisfies all the properties or has no crosses. Since the only matching with no crosses is M_D and all the 4 properties do hold for M_D, in either case we are guaranteed to construct M^*.

Transformation T_1: Let $W = \{u_a, u_b, u_c\} \subseteq U$ and $e_1 = (u_a, v_q), e_2 = (u_b, v_p), e_3 = (u_c, v_d)$ such that e_2 crosses e_1 and e_3; e_1 is to the left of e_2 and e_3 is to the right of e_2. This implies that $a < b < c$ and $d < p < q$.

Let $M'' = M' \otimes \{e_1, e_3\}$. Since, M' was a cyclic augmentation, M'' will contain two cycles with $e'_1 = (u_a, v_d)$ and $e'_3 = (u_c, v_q)$ in different cycles. Therefore, e_2 will be in the same cycle as either e'_1 or e'_3. Suppose e_2 is in the same cycle as e'_1. Let $M''' = M'' \otimes \{e_{u_b}^{M''}, e'_3\}$. Now M''' is a cyclic augmentation and the edges $e_{u_a}^{M'''}$, $e_{u_b}^{M'''}$ and $e_{u_c}^{M'''}$ do not violate P_1. The other case when e_2 is in the same cycle as e'_3 is symmetric.

We have transformed M' to M''' using one cross-to-straight exchange and one straight-to-cross exchange. However, M' can be transformed to M''' using only cross-to-straight exchanges (see Figure 6). Therefore, the cost of M''' is no more than that of M' (using Lemma 3). Further, using Lemma 6(2) we have $\Gamma_{M'''} < \Gamma_{M'}$.

Transformation T_2: Let $W = \{u_a, u_c, u_p, u_r\}$ and $e_1 = (u_a, v_b), e_2 = (u_c, v_d), e_3 = (u_p, v_q), e_4 = (u_r, v_s)$ such that e_1 crosses e_2 and e_3 and so does e_4. Using P_1 we can conclude that e_2 and e_3 should be in a straight state and so do e_1 and e_4. Then without loss of generality $d < q < b < s$ and $a < r < c < p$ (see Figure 7). Let $M'' = M' \otimes ((u_a, v_b), (u_p, v_q))$. Again $G_{M''}$ contains two cycles C_1 and C_2 such that $(u_a, v_q) \in C_1$ and $(u_p, v_b) \in C_2$. We will now show that there exists a cross-to-straight exchange that combines C_1 and C_2. We have the following three cases:
Case 1: (u_r, v_s) and (u_c, v_d) belong to different cycles. We exchange (u_r, v_s) and (u_c, v_d).
Case 2: $(u_r, v_s), (u_c, v_d) \in C_1$. We exchange (u_r, v_s) and (u_p, v_b).
Case 3: $(u_r, v_s), (u_c, v_d) \in C_2$. We exchange (u_a, v_q) and (u_c, v_d).

Since all of these exchanges are cross-to-straight exchanges, the resultant matching between U and V is a minimum cost cyclic augmentation for $G = (U, V, H)$ with a smaller cross number than M'.

Transformation T_3: Let $W = \{u_p, u_q\} \subseteq U$ such that both are in the same cycle in G_{M_D} and $e_{u_p}^{M'}$ crosses $e_{u_q}^{M'}$. Consider $M'' = M' \otimes (e_{u_p}^{M'}, e_{u_q}^{M'})$. $G_{M''}$ contains two cycles, and $e_{u_p}^{M''}$ and $e_{u_q}^{M''}$ are in different cycles in $G_{M''}$, say, C_1 and C_2. Now we claim that there exist two edges $e_1 \in C_1$ and $e_2 \in C_2$ in a cross state. If the claim is true, then consider $M''' = M'' \otimes (e_1, e_2)$. M''' is a cyclic augmentation for $G = (U, V, H)$ with cost no more than that of M' and with $\Gamma_{M'''} < \Gamma_{M'}$ because we have only performed cross-to-straight exchanges.

To prove the claim by contradiction, suppose that the claim did not hold. Then we can perform a sequence of cross-to-straight exchanges till no two edges are in a cross state. Note that if there were no cross between C_1 and C_2 to begin with, using Lemma 5 we can conclude that none of these exchanges is between an edge incident to C_1 and one incident to C_2. Therefore, in the resultant graph the edges adjacent to u_p and u_q are in different cycles. But the resultant matching is M_D, and $e_{u_p}^{M_D}$ and $e_{u_q}^{M_D}$ are in the same cycle in G_{M_D}, hence reaching a contradiction. Therefore, the claim holds true.

Transformation T_4: Suppose the first condition of P_4 is not satisfied. The transformation for the other condition is similar. Let $W = \{u_p, u_q\}$ such that both u_p and u_q are in the same cycle

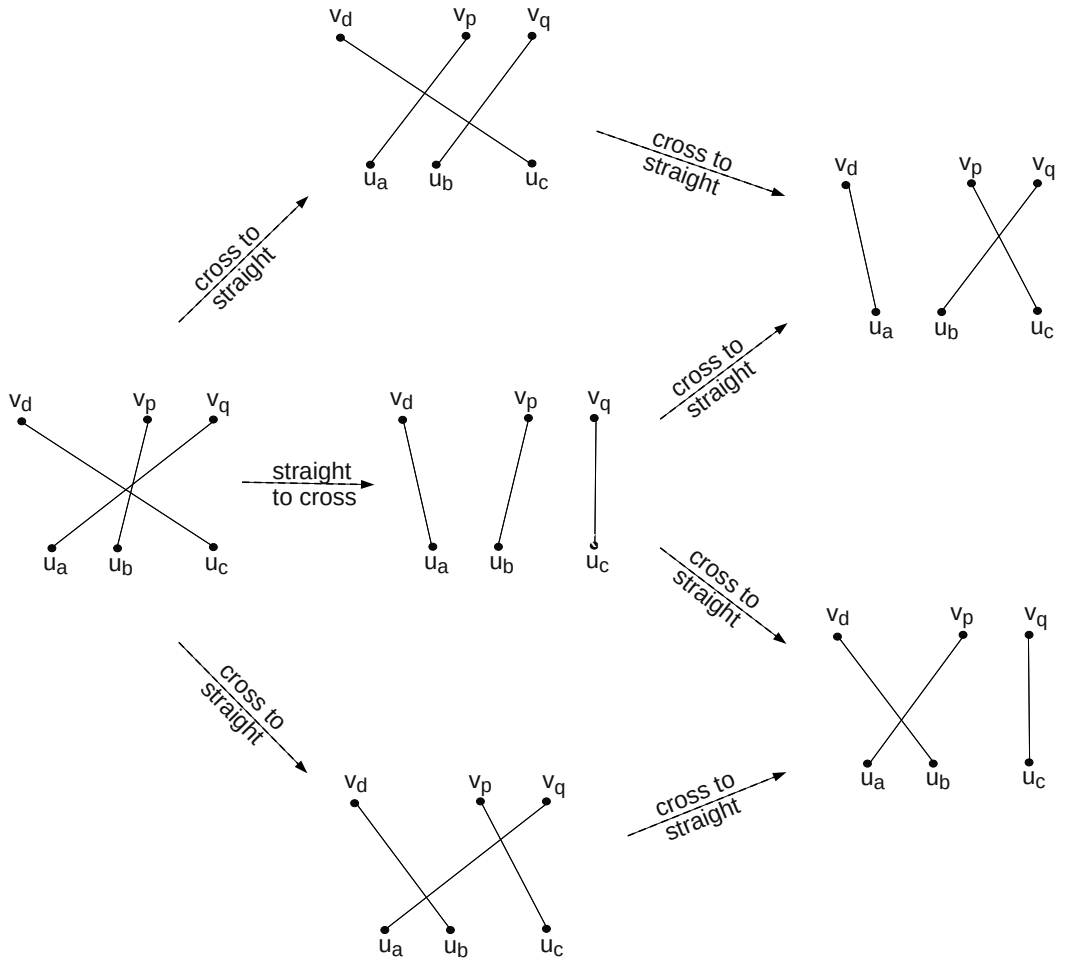

Figure 6: Illustration of transformation T_1.

in G_{M_D}. Let $e_{u_p}^{M'} = (u_p, v_a)$ and $e_{u_q}^{M'} = (u_q, v_b)$. Since the first condition of P_4 does not hold, there exists $(u_d, v_c) \in M'$ such that $e_{u_p}^{M'}$ crosses $e_{u_d}^{M'}$ and u_d is to the right of u_q, i.e., $p < q < d$. Using P_3 we have $c < a < b$. Let $M'' = M' \otimes (e_{u_p}^{M'}, e_{u_d}^{M'})$. $G_{M''}$ contains two cycles; and (u_p, v_c) and (u_d, v_a) in $G_{M''}$ are in different cycles, say, C_1 and C_2. Now (u_q, v_b) is in either C_1 or C_2. If $(u_q, v_b) \in C_1$, then we can exchange (u_q, v_b) and (u_d, v_a). If $(u_q, v_b) \in C_2$, then there must exist a cross between an edge in C_1 and an edge in C_2 by a similar argument to that used for T_3.

Hence, there exists another cross-to-straight exchange to combine C_1 and C_2 such that in the resultant matching M''', $e_{u_p}^{M'''}$ and $e_{u_d}^{M'''}$ do not cross, and M''' is constructed form M' using only cross-to-straight exchanges. Therefore, the cost of M''' is no more than that of M', and $\Gamma_{M'''} < \Gamma_{M'}$.

□

The next lemma uses the properties established in Lemma 7 to restrict the space of exchanges required for transforming M_D to an optimum bottleneck cyclic augmentation M^* to the exchanges corresponding to edges in $\mathcal{X}$.

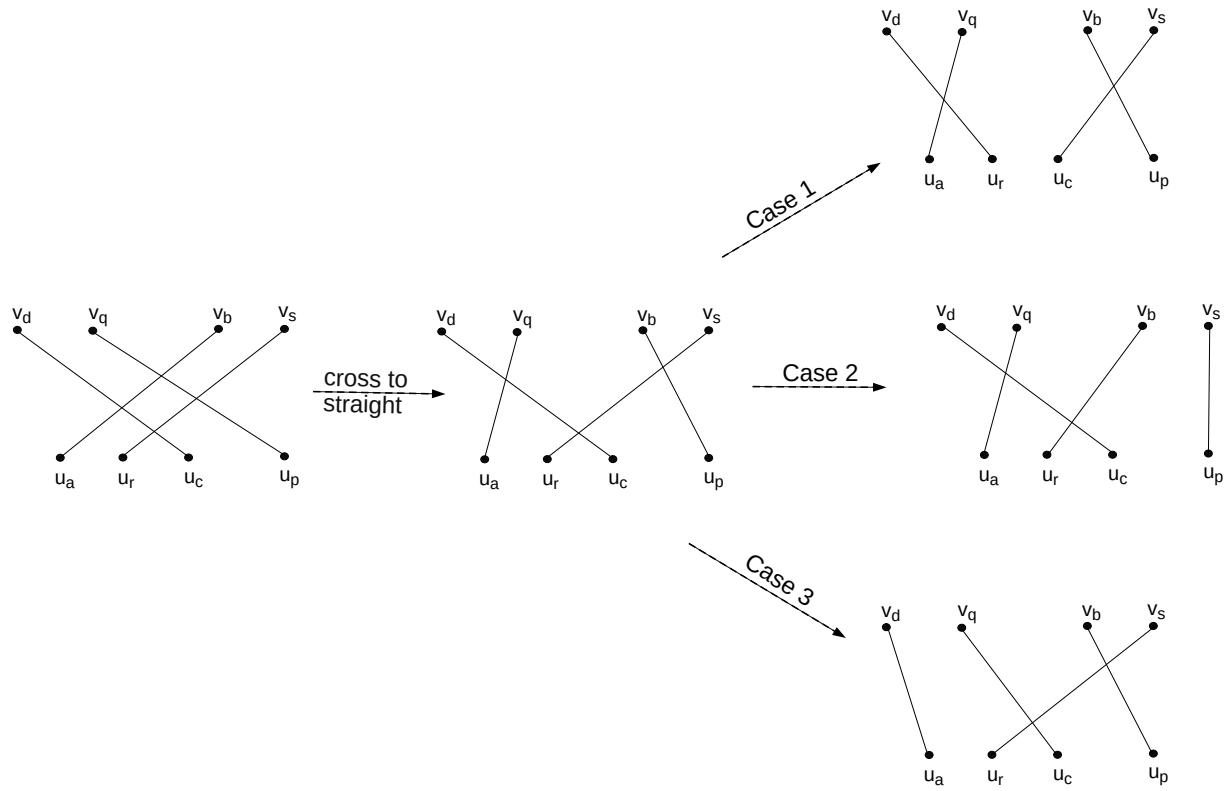

Figure 7: Illustration of transformation T_2.

Lemma 8. *M^* can be constructed by performing a series of exchanges on G_{M_D} where each exchange corresponds to a unique edge in the exchange graph $\mathcal{X}$.*

Proof. We will construct a sequence of exchanges $x_0, \ldots, x_{n-1}$ where each x_i is either a null exchange or corresponds to a unique edge in the exchange graph $\mathcal{X}$. Let $M_k = M_D \otimes x_0 \otimes \cdots \otimes x_k$. It suffices to show that $M_{n-1} = M^*$. A vertex in M_k is said to be *satisfied* if its adjacent vertex in M_k is the same as its adjacent vertex in M^*.

Let $H(k)$ denote the statement that in M_k either all the vertices in the first $k+1$ direct pairs are satisfied or exactly two are not, and that at least one of those two, h, is in the $(k+1)^{\text{th}}$ direct pair and the other is the one adjacent to it, g, in M_k such that $\phi(g) \leq \phi(h)$.

The proof of this lemma is by induction on $H(k)$ as follows. Note that if $H(n-1)$ is true, then all the vertices in M_{n-1} must be satisfied, i.e. M_{n-1} is the same as M^*.

Base Case: $k = 0$. Let x_0 be a null exchange. Either the vertices of first direct pair are satisfied or they are not. In either case, $H(0)$ is true.

Induction Step: $H(k-1)$ holds for some k, where $0 \leq k-1 \leq n-2$. Then, if all of the first k direct pairs are satisfied, let x_k be a null exchange and $H(k)$ will be true.

However, if two vertices of the first k direct pairs are not satisfied, let the two vertices be g and h such that $\phi(g) \leq \phi(h)$ and h belongs to the k^{th} direct pair. Note that in this case M_{k-1} must contain the edge (g, h). Without loss of generality, let $g \in U$ and $h \in V$. Since all the first k direct pairs except g and h are satisfied, the vertices adjacent to g and h in M^* must be to the right of the k^{th} direct pair.

Now let x_k be the exchange between (g, h) and (u_{k+1}, v_{k+1}). Using P_3 and P_4 we can conclude

that h and u_{k+1} (or v_{k+1}) cannot belong to same clusters. So the exchange x_k corresponds to an edge labeled $(k, k+1)$ in the exchange graph $\mathcal{X}$.

We need to show that either $(g, v_{k+1}) \in M^*$ or $(h, u_{k+1}) \in M^*$. To prove this by contradiction, suppose this is not so. That is $(g, v_{k+1}) \notin M^*$ and $(h, u_{k+1}) \notin M^*$. Then let the vertex adjacent to g in M^* be g' and that adjacent to v_{k+1} be v'_{k+1}. Similarly, let the vertex adjacent to h in M^* be h' and that adjacent to u_{k+1} be u'_{k+1}. We know now that g' should be to the right of v_{k+1} and h' should be right of u_{k+1}. By P_1, v'_{k+1} should be to the right of h', and u'_{k+1} should be to the right of g'. But by P_2 this is not possible. Hence, we have reached a contradiction.

Therefore, by induction we can conclude that $H(n-1)$ is true. □

Let c_{MST} be the weight of the heaviest edge in a minimum spanning tree over $\mathcal{X}$ and let $c_{\text{LB}} = \max\{c_{M_D}, c_{\text{MST}}\}$. Let the cost of the optimal bottleneck cyclic augmentation M^* be c_{OPT}.

Lemma 9. $c_{\text{OPT}} \geq c_{\text{LB}}$.

Proof. We first prove $c_{\text{OPT}} \geq c_{M_D}$ and then $c_{\text{OPT}} \geq c_{\text{MST}}$.

Since M_D contains no edges in cross state, it is the minimum cost matching of $G(U, V, H)$ and hence $c_{\text{OPT}} \geq c_{M_D}$.

From Lemma 8, there exists a series of exchanges $x_0, \ldots, x_{n-1}$ such that $M_k = M_D \otimes x_0 \otimes \cdots \otimes x_k$ and $M^* = M_{n-1}$ and each x_i is either a null exchange or corresponds to a unique edge e_i in $\mathcal{X}$. Let the set of edges in $\mathcal{X}$ corresponding to these exchanges be E_{OPT}.

Note that c_{M_k} is at least the weight of e_k, and since each of the x_i is a straight-to-cross exchange, $c_{M_{n-1}} \geq c_{M_{n-2}} \geq \cdots \geq c_{M_0}$. Therefore, $c_{M^*} = c_{M_{n-1}}$ is at least the maximum weighted edge in E_{OPT}. Using Lemma 4, since $M^* = M_{n-1}$ is a cyclic augmentation, E_{OPT} should form a spanning tree of $\mathcal{X}$. Therefore, $c_{\text{OPT}} \geq c_{\text{MST}}$. □

5 Approximation Algorithm for GG-BTSP

For finding the minimum bottleneck cost augmentation, we first construct a minimum spanning tree of the exchange graph and then perform exchanges corresponding to the edges of the spanning tree such that no exchange exceeds the weight of the heaviest edge in the spanning tree by a factor of $(2 + \gamma)$.

From Lemma 8, there exists a set of exchanges corresponding to the edges of exchange graph such that the resulting augmentation is of optimum cost. Furthermore, as observed in proof to Lemma 9, these edges form a spanning tree. However, note that the spanning tree corresponding to the optimal augmentation need not be the minimum spanning tree over the exchange graph. Furthermore, the cost of the augmentation is only lower bounded by the heaviest weighted edge in the corresponding spanning tree. Hence, there is scope for improving the analysis by tightening the lower bound.

Lemma 10. *At Step 5b of Approx-BTSP, either* $c(e^M_{u_{a_i}}) \leq c_{\text{LB}}$ *or* $c(e^M_{v_{a_i}}) \leq c_{\text{LB}}$.

Proof. This is proven by induction on $H(k)$ where $H(k)$ denotes the statement that after k iterations of the algorithm

1. Either $c(e^M_{u_{a_k}}) \leq c_{\text{LB}}$ or $c(e^M_{v_{a_k}}) \leq c_{\text{LB}}$; and
2. For all $x > a_k$, $e^M_{u_x} = e^M_{v_x} = (u_x, v_x)$.

Algorithm 1 Approx-BTSP

1. Let $M \leftarrow M_D$.
2. Construct the exchange graph $\mathcal{X}$.
3. Find a minimum spanning tree T of $\mathcal{X}$.
4. Sort the edges in T in the increasing order of their label. Let the ordered edges be $e_1, \ldots, e_m$.
5. For $i = 1$ to m,
   (a) Let the label of e_i be $(a_i, a_i + 1)$.
   (b) **if** $c(e^M_{u_{a_i}}) \leq c_{\text{LB}}$,
   then $M \leftarrow M \otimes (e^M_{u_{a_i}}, e^M_{u_{(a_i+1)}})$;
   else $M \leftarrow M \otimes (e^M_{v_{a_i}}, e^M_{u_{(a_i+1)}})$.
6. Output $M_{\mathbf{OUT}} = M$.

Base Case: $k = 1$, $M = M_D$ and hence all the edges in M have cost at most c_{LB}. Therefore, $H(1)$ is true.

Induction Step: $H(k-1)$ holds for some k, where $0 \leq k - 1 \leq m - 1$. Note that, $a_k \geq a_{k-1} + 1$ since the edges were sorted according to their labels. Without loss of generality, assume that at $(k-1)^{\text{th}}$ iteration $c(e^M_{u_{a_{k-1}}}) \leq c_{\text{LB}}$. Therefore after the $(k-1)^{\text{th}}$ iteration, $e^M_{u_{a_{k-1}}} = (u_{a_{k-1}}, v_{a_{k-1}+1})$ and for all $x > a_{k-1} + 1$, $e^M_{u_x} = e^M_{v_x} = (u_x, v_x)$. Note that weight of the edge labeled $(a_{k-1}, a_{k-1} + 1)$ is at least $c(u_{a_{k-1}}, v_{a_{k-1}+1})$ and since $e_{k-1} \in T$, $c(e^M_{v_{a_{k-1}+1}}) \leq c_{\text{LB}}$.

Therefore at the k^{th} iteration, $c(e^M_{v_{a_{k-1}+1}}) \leq c_{\text{LB}}$ and for all $x > a_{k-1}+1$, $e^M_{u_x} = e^M_{v_x} = (u_x, v_x)$. Since $a_k \geq a_{k-1} + 1$, $H(k)$ holds. □

Lemma 11. *If* $\gamma \geq \frac{f(x)}{g(x)} \geq \frac{1}{\gamma}\ \forall x$, *then* $c(b,a) \leq \gamma \cdot c(a,b)$.

Proof. We have the following three cases.
Case 1: $a < b$. Then $c(a,b) = \int_a^b f(x)dx$ and $c(b,a) = \int_a^b g(x)dx \leq \int_a^b \gamma \cdot f(x)dx = \gamma \cdot c(a,b)$.
Case 2: $a > b$. Then $c(a,b) = \int_b^a g(x)dx$ and $c(b,a) = \int_b^a f(x)dx \leq \int_b^a \gamma \cdot g(x)dx = \gamma \cdot c(a,b)$.
Case 3: $a = b$. Then $c(a,b) = c(b,a)$. □

Theorem 1. *Running time of Algorithm Approx-BTSP is* $O(n \log n)$.

Proof. Steps 1, 2 and 5 take $O(n)$ time while Steps 3 and 4 take $O(n \log n)$ time. Therefore, the time complexity of the algorithm is $O(n \log n)$. □

Theorem 2. *If* $\gamma \geq \frac{f(x)}{g(x)} \geq \frac{1}{\gamma}\ \forall x$, *then* $M_{\boldsymbol{OUT}}$ *is a cyclic augmentation of cost no more than* $(2+\gamma) \cdot c_{\text{OPT}}$

Proof. We need to prove the following two parts:

1. $M_{\mathbf{OUT}}$ is a cyclic augmentation.
2. $c_{M_{\mathbf{OUT}}} \leq (2+\gamma) \cdot c_{\text{OPT}}$.

Note that every exchange performed in the algorithm is between edges belonging to two different cycles. Therefore, using Lemma 4, the number of cycles in M decreases with every iteration. If M_D has $m+1$ cycles, the minimum spanning tree T contains m edges and hence after m iterations M consists of just one cycle and is hence a cyclic augmentation. This completes the proof of Part 1.

For Part 2, we will show that the following invariant holds true for the algorithm: $c_M \leq (2+\gamma)\cdot c_{\text{LB}}$. Consider the i^{th} iteration of the algorithm. Let the matching before the i^{th} iteration be M' and the one after be M''. Assuming $c_{M'} \leq (2+\gamma)\cdot c_{\text{LB}}$, we need to show that $c_{M''} \leq (2+\gamma)\cdot c_{\text{LB}}$. From Lemma 10, either $c(e^{M'}_{u_{a_i}}) \leq c_{\text{LB}}$ or $c(e^{M'}_{v_{a_i}}) \leq c_{\text{LB}}$. Without loss of generality, assume $c(e^{M'}_{u_{a_i}}) \leq c_{\text{LB}}$. Let $e^{M'}_{u_{a_i}} = (u_{a_i}, h)$. $M'' = (M' \setminus \{(u_{a_i}, h), (u_{a_i+1}, v_{a_i+1})\}) \cup \{(u_{a_i}, v_{a_i+1}), (u_{a_i+1}, h)\}$. Clearly $c((u_{a_i}, v_{a_i+1})) \leq c_{\text{LB}}$ because weight of edge labeled (a_i, a_i+1) is at least $c((u_{a_i}, v_{a_i+1}))$.

So all we need to show is that $c((u_{a_i+1}, h)) \leq (2+\gamma)\cdot c_{\text{LB}}$.

$$c(\phi(u_{a_i+1}), \phi(v_{a_i})),\ c(\phi(u_{a_i}), \phi(v_{a_i})),\ c(\phi(u_{a_i}), \phi(h)) \leq c_{\text{LB}}$$

$$\begin{aligned} c(\phi(u_{a_i+1}), \phi(u_{a_i})) &\leq c(\phi(u_{a_i+1}), \phi(v_{a_i})) + c(\phi(v_{a_i}), \phi(u_{a_i})) \\ &\leq c(\phi(u_{a_i+1}), \phi(v_{a_i})) + \gamma\cdot c(\phi(u_{a_i}), \phi(v_{a_i})) \qquad \text{(using Lemma 11)} \\ &\leq (1+\gamma)\cdot c_{\text{LB}} \end{aligned}$$

$$\begin{aligned} c((u_{a_i+1}, h)) &= c(\phi(u_{a_i+1}), \phi(h)) \\ &\leq c(\phi(u_{a_i+1}), \phi(u_{a_i})) + c(\phi(u_{a_i}), \phi(h)) \\ &\leq (1+\gamma)\cdot c_{\text{LB}} + c_{\text{LB}} \\ &= (2+\gamma)\cdot c_{\text{LB}} \end{aligned}$$

□

References

[1] C. Bailey-Kellogg, S. Chainraj, and G. Pandurangan, *A Random Graph Approach to NMR Sequential Assignment*, in Proceedings of the 8^{th} Annual International Conference on Computational Molecular Biology, 2004, pp. 58–67.

[2] M. O. Ball and M. J. Magazine, *Sequencing of Insertions in Printed Circuit Board Assembly*, Operations Research, 36 (1988), pp. 192–201.

[3] Z.-Z. Chen, T. Jiang, G. Lin, J. Wen, D. Xu, J. Xu, and Y. Xu, *Approximation Algorithms for NMR Spectral Peak Assignment*, Theoretical Computer Science, 299 (2003), pp. 211–229.

[4] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman & Co., New York, NY, USA, 1979.

[5] P. C. Gilmore and R. E. Gomory, *Sequencing a One State-Variable Machine: A Solvable Case of the Traveling Salesman Problem*, Operations Research, 12 (1964), pp. 655–679.

[6] G. Gutin and A. P. Punnen, *The Traveling Salesman Problem and Its Variations*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2002.

[7] T. K. Hitchens, J. A. Lukin, Y. Zhan, S. A. McCallum, and G. S. Rule, *MONTE: An Automated Monte Carlo Based Approach to Nuclear Magnetic Resonance Assignment of Proteins*, Journal of Biomolecular NMR, 25 (2003), pp. 1–9.

[8] S. S. Reddi and C. V. Ramamoorthy, *On the Flow-Shop Sequencing Problem with No Wait in Process*, Operational Research Quarterly, 23 (1972), pp. 323–331.

[9] G. L. Vairaktarakis, *On Gilmore-Gomory's open question for the bottleneck TSP*, Operations Research Letters, 31 (2003), pp. 483–491.

[10] ——, *Simple Algorithms for Gilmore-Gomory's Traveling Salesman and Related Problems*, Journal of Scheduling, 6 (2003), pp. 499–520.

[11] O. Vitek, J. Vitek, B. Craig, and C. Bailey-Kellogg, *Model-Based Assignment and Inference of Protein Backbone Nuclear Magnetic Resonances*, Statistical Applications in Genetics and Molecular Biology, 3 (2004), pp. 1–22.

[12] X. Wan, D. Xu, C. M. Slupsky, and G. Lin, *Automated Protein NMR Resonance Assignments*, in Proceedings of the 2^{nd} IEEE Computer Society Conference on Bioinformatics, 2003, pp. 197–208.

[13] Y. Xu, D. Xu, D. Kim, V. Olman, J. Razumovskaya, and T. Jiang, *Automated Assignment of Backbone NMR Peaks Using Constrained Bipartite Matching*, Computing in Science and Engineering, 4 (2002), pp. 50–62.