An approximation algorithm for the Bandpass-2 problem

Weitian Tong∗,† Zhi-Zhong Chen† Lusheng Wang‡ Yinfeng Xu∗∥
Jiuping Xu∗,** Randy Goebel∗,†† Guohui Lin∗,‡‡

May 11, 2014

Abstract

The general Bandpass-B problem is NP-hard and can be approximated by a reduction into the weighted B-set packing problem, with a worst case performance ratio of O(B^2). When B = 2, a maximum weight matching gives a 2-approximation to the problem. In this paper, we call the Bandpass-2 problem simply the Bandpass problem. The Bandpass problem can be viewed as a variation of the maximum traveling salesman problem, in which the edge weights are dynamic rather than given at the front. We present an O(B^2) approximation algorithm for the problem. Such an improved approximation is built on an intrinsic structural property proven for the optimal solution and several novel schemes to partition a b-matching into desired matchings.

Keywords: The Bandpass problem, maximum weight b-matching, acyclic 2-matching, approximation algorithm, worst case performance ratio

1 Introduction

In optical communication networks, a sending point uses a binary matrix A_{n \times m} to send n information packages to m different destination points, in which the entry a_{ij} = 1 if information package i is not destined for point j, or a_{ij} = 0 otherwise. To achieve the highest cost reduction via wavelength division multiplexing technology, an optimal packing of information flows on different wavelengths into groups is necessary [3]. Under this binary matrix representation, every B consecutive 1’s in a column indicates an opportunity for merging information to reduce the communication cost, where B is a pre-specified positive integer called the bandpass number. Such a set of B consecutive 1’s

∗Department of Computing Science, University of Alberta. Edmonton, Alberta T6G 2E8, Canada.
†Email: weitian@ualberta.ca
‡Division of Information System Design, Tokyo Denki University. Hatoyama, Saitama 350-0394, Japan. Email: zzchen@mail.dendai.ac.jp
§Department of Computer Science, City University of Hong Kong. Kowloon, Hong Kong, China. Email: cswang@cityu.edu.hk
∥Email: yfxu@scu.edu.cn
∗∗Email: xujiuping@scu.edu.cn
∗†Email: rgoebel@ualberta.ca
‡‡Correspondence author. Email: guohui@ualberta.ca
in a column of the matrix is said to form a bandpass. When counting the number of bandpasses in the matrix, no two of them in the same column are allowed to share any common rows. The computational problem, the Bandpass-B problem, is to find an optimal permutation of rows of the input matrix $A_{n \times m}$ such that the total number of extracted bandpasses in the resultant matrix is maximized [4, 3, 13]. Note that though multiple bandpass numbers can be used in practice, for the sake of complexities and costs, usually only one fixed bandpass number is considered [3].

The general Bandpass-B problem, for any fixed $B \geq 2$, has been proven to be NP-hard [13]. In fact, the NP-hardness of the Bandpass-2 problem can be proven by a reduction from the well-known Hamiltonian path problem [10], where in the constructed binary matrix $A_{n \times m}$, a row maps to a vertex, a column maps to an edge, and $a_{ij} = 1$ if and only if edge $e_j$ is incident to vertex $v_i$. It follows that there is a row permutation achieving $n - 1$ bandpasses if and only if there is a Hamiltonian path in the graph.

On the approximability, the Bandpass-B problem has a close connection to the weighted $B$-set packing problem [10]. Given an instance $I$ of a maximization problem $\Pi$, let $C^*(I)$ ($C(I)$, respectively) denote the value of the optimal solution (the value of the solution produced by an algorithm, respectively). The performance ratio of the algorithm on $I$ is $C^*(I)/C(I)$. The algorithm is a $\rho$-approximation if $\sup_I C^*(I)/C(I) \leq \rho$. By taking advantages of the approximation algorithms designed for the weighted $B$-set packing problem [2, 5], the Bandpass-B problem can be approximated within $O(B^2)$ [13]. Moreover, since the maximum weight matching problem is solvable in cubic time, the Bandpass-2 problem admits a simple maximum weight matching based 2-approximation algorithm [13]. In the sequel, we call the Bandpass-2 problem simply the Bandpass problem.

In a preliminary version of this paper [18], Tong et al. presented a first improved approximation algorithm for the Bandpass problem using two maximum weight matchings. Their algorithm has a worst case performance ratio of $\frac{36}{19} \approx 1.8948$. In more details, their algorithm computes a maximum weight matching in the edge-weighted graph constructed from the input matrix, and then computes another maximum weight matching in the residual graph; Through breaking cycles in the union of these two matchings, a collection of paths are achieved and concatenated into a Hamiltonian path. When estimating the weight of this Hamiltonian path, the authors present an intrinsic structural property for the optimal row permutation, and show that the weight of the second maximum weight matching is greater than or equal to a portion of certain bandpasses in the optima. These special bandpasses in the optima are characterized using the first maximum weight matching.

Instead of breaking cycles in the union of two matchings, Chen and Wang [7] presented an alternative to compute a second matching such that the union of the two matchings is guaranteed acyclic. The keys to this success are two lemmas that partition a 2-matching (an acyclic 2-matching, respectively) into $4$ ($3$, respectively) candidate matchings. Still based on the structural property characterized in [18], this alternative algorithm achieves a better performance ratio of $\frac{220}{117} \approx 1.8805$.

In this paper, we advance the novel 2-matching partitioning scheme to show that two edge disjoint 2-matchings can be partitioned into 7.5 desired matchings such that the union of each of them and the very first maximum weight matching is guaranteed acyclic. Including here all the detailed proofs missed from the two preliminary versions [18, 7], we show that our new approximation algorithm has a performance ratio of $\frac{426}{227} \approx 1.8767$. 
2 The approximation algorithm

A reduction from the Hamiltonian path problem has been used to prove the NP-hardness of the Bandpass problem. But the Bandpass problem does not readily reduce to the maximum traveling salesman problem (Max-TSP) for approximation algorithm design. The main reason is that, an instance graph of Max-TSP is static, in that all (non-negative) edge weights are given at the front, while in the Bandpass problem the number of bandpasses extracted between two consecutive rows in a row permutation is permutation dependent. Nevertheless, as shown in the sequel, our design idea is based on maximum weight b-matchings for b = 1, 2, and 4, similarly as in approximating Max-TSP [16, 12, 6, 15]. Formally, in Max-TSP, a complete edge-weighted graph is given, where the edge weights are non-negative integers, and the goal is to compute a Hamiltonian cycle with the maximum weight. Note that there are several variants of Max-TSP been studied in the literature. In our case, the input graph is undirected (or symmetric) and the edge weights do not necessarily satisfy the triangle inequality. The following Lemma 1 states the currently best approximation result for Max-TSP.

Lemma 1 [15] The Max-TSP admits an O(n^3)-time \( \frac{2}{3} \)-approximation algorithm, where n is the number of vertices in the graph.

In our Bandpass problem, since we can always add a row of all 0’s if needed, we assume without loss of generality that the number of rows, n, is even. A b-matching of a graph is a subgraph in which the degree of each vertex is at most b. A maximum weight b-matching of an edge weighted graph can be computed in O(n^2m) time [9, 11, 14], where n is the number of vertices and m is the number of edges in the graph. Note that a 2-matching is a collection of vertex-disjoint cycles and paths. A 2-matching is acyclic if it does not contain any cycle (i.e., it is a collection of vertex-disjoint paths).

Given the input binary matrix \( A_{n \times m} \), let \( r_i \) denote the i-th row. We first construct a graph \( G \) of which the vertex set is exactly the row set \( \{ r_1, r_2, \ldots, r_n \} \). Between rows \( r_i \) and \( r_j \), the static edge weight is defined as the maximum number of bandpasses that can be formed between the two rows, and is denoted as \( w(i,j) \). In the sequel we use row (of the matrix) and vertex (of the graph) interchangeably.

For a row permutation \( \pi = (\pi_1, \pi_2, \ldots, \pi_n) \), its i-th row is the \( \pi_i \)-th row in the input matrix. We call a maximal segment of consecutive 1’s in a column of \( \pi \) a strip of \( \pi \). The length of a strip is defined to be the number of 1’s therein. A length-\( \ell \) strip contributes exactly \( \lfloor \frac{\ell}{2} \rfloor \) bandpasses to the permutation \( \pi \). We use \( S_\ell(\pi) \) to denote the set of all length-\( \ell \) strips of \( \pi \), and \( s_\ell(\pi) = |S_\ell(\pi)| \). Let \( b(\pi) \) denote the number of bandpasses extracted from the permutation \( \pi \). We have

\[
b(\pi) = \sum_{\ell=2}^{n} s_\ell(\pi) \left\lfloor \frac{\ell}{2} \right\rfloor = s_2(\pi) + \sum_{\ell=3}^{n} s_\ell(\pi) \left\lfloor \frac{\ell}{2} \right\rfloor. \quad (2.1)
\]

Let \( p(\pi) \) denote the number of pairs of consecutive 1’s in the permutation \( \pi \). A length-\( \ell \) strip contributes exactly \( \ell - 1 \) pairs to the permutation \( \pi \). We have

\[
p(\pi) = \sum_{\ell=2}^{n} s_\ell(\pi) (\ell - 1) = s_2(\pi) + \sum_{\ell=3}^{n} s_\ell(\pi) (\ell - 1). \quad (2.2)
\]
2.1 Algorithm description

In our algorithm denoted as APPROX, the first step is to compute a maximum weight matching $M_1$ in graph $G$. Recall that there are an even number of rows. Therefore, $M_1$ is a perfect matching (even though some edge weights could be 0). Let $w(M_1)$ denote the sum of its edge weights, indicating that exactly $w(M_1)$ bandpasses can be extracted from the row pairings suggested by $M_1$. These bandpasses are called the bandpasses of $M_1$.

Next, every 1 involved in a bandpass of $M_1$ is changed to 0. Let the resultant matrix be denoted as $A'_{n \times m}$, the resultant edge weight between rows $r_i$ and $r_j$ be $w'(i,j)$ — which is the maximum number of bandpasses that can be formed between the two revised rows, and the corresponding residual graph be denoted as $G'$. One can see that if an edge $(r_i, r_j)$ belongs to $M_1$, then the new edge weight $w'(i,j) = 0$. In the second step of APPROX, we compute a maximum weight 4-matching $C$ in graph $G'$, which is further decomposed in $O(n^{2.5})$ time into two 2-matchings denoted as $C_1$ and $C_2$ \[11, 8\]. Let $w'(C)$ denote the weight (the number of bandpasses) of $C$ in the residual graph $G'$. It is noted that no bandpass of $C$ shares a 1 with any bandpass of $M_1$. Using $M_1$ and $C_1$ and $C_2$, by Lemma[5] we can compute a matching $M_2$ from $C$ of weight at least $\frac{1}{7} \cdot w'(C)$ such that $G[M_1 \cup M_2]$ is guaranteed acyclic.

In the third step, we use the $\frac{2}{3}$-approximation algorithm described in \[15\] to compute a Hamiltonian path $\mathcal{P}$ in $G'$ whose weight is at least $\frac{2}{3}$ of the maximum weight of a Hamiltonian path. Then, using $M_1$ and $\mathcal{P}$, by Lemma[3] we can compute another matching $M_2$ from $\mathcal{P}$ of weight at least $\frac{1}{3} \cdot w'(\mathcal{P})$ such that $G[M_1 \cup M_2]$ is guaranteed acyclic.

In the last step, we choose the larger one between the two $M_2$’s found in the last two steps, and arbitrarily stack the paths in $G[M_1 \cup M_2]$ to give a row permutation $\pi$. Note that the number of bandpasses extracted from $\pi$, $b(\pi)$, is greater than or equal to $w(M_1) + w'(M_2)$.

2.2 Performance analysis

Let $\pi^*$ denote the optimal row permutation such that its $b(\pi^*)$ is maximized over all row permutations. Correspondingly, $S_2(\pi^*)$ denotes the set of length-2 strips in $\pi^*$, which contributes exactly $s_2(\pi^*)$ bandpasses towards $b(\pi^*)$. The key part in the performance analysis for algorithm APPROX is to estimate $w'(M_2)$, as done in the following.

First, we partition the bandpasses of $S_2(\pi^*)$ into four groups: $B_1$, $B_2$, $B_3$, $B_4$. Note that bandpasses of $S_2(\pi^*)$ do not share any 1 each other. $B_1$ consists of the bandpasses of $S_2(\pi^*)$ that also belong to matching $M_1$ (such as the one between rows $r_a$ and $r_b$ in Figure[1]); $B_2$ consists of the bandpasses of $S_2(\pi^*)$ such that they are uniquely paired up to contribute a 1 each to form a bandpass of $M_1$ (the other 1 in each bandpass of $B_2$ is thus not shared by any other bandpass of $M_1$); $B_3$ consists of the bandpasses of $S_2(\pi^*)$, each of which shares a 1 with at least one bandpass of $M_1$, and if it shares a 1 with only one bandpass of $M_1$ then the other 1 of this bandpass of $M_1$ is not shared by any other bandpass of $S_2(\pi^*)$; $B_4$ consists of the remaining bandpasses of $S_2(\pi^*)$. Figure[1] illustrates some examples of these bandpasses, where bandpasses of $S_2(\pi^*)$ are in ovals and bandpasses of $M_1$ are in boxes.
Figure 1: An illustration of the bandpasses of $S_2(\pi^*)$ (in ovals) and the bandpasses of $M_1$ (in boxes) for grouping purpose. A horizontal line in the figure represents a row, led by its index. Rows that are adjacent in $\pi^*$ and/or row pairs of $M_1$ are intentionally ordered adjacently. In this figure, rows $r_a$ and $r_b$ are adjacent in $\pi^*$, denoted as $(r_a, r_b) \in \pi^*$, and edge $(r_a, r_b) \in M_1$ as well; the bandpasses between these two rows in $S_2(\pi^*)$ thus belong to $B_1$. Edges $(r_i, r_j), (r_j, r_k), (r_\ell, r_u) \in M_1$, while $(r_i, r_j), (r_k, r_\ell) \in \pi^*$; the bandpasses between rows $r_i$ and $r_j$ and between rows $r_k$ and $r_\ell$ in $S_2(\pi^*)$ shown in the figure have their group memberships indicated beside them respectively.

By the definition of partition, we have

$$s_2(\pi^*) = |B_1| + |B_2| + |B_3| + |B_4|. \quad (2.3)$$

From these “group” definitions, we know all bandpasses of $B_1$ are in $M_1$. Also, one pair of bandpasses of $B_2$ correspond to a distinct bandpass of $M_1$. Bandpasses of $B_3$ can be further partitioned into subgroups such that a subgroup of bandpasses together with a distinct maximal subset of bandpasses of $M_1$ form into an alternating cycle or path of length at least 2. Moreover, 1) when the path length is even, the number of bandpasses of this subgroup of $B_3$ is equal to the number of bandpasses of this subset of bandpasses of $M_1$; 2) when the path length is odd, 2a) either the number of bandpasses of this subgroup of $B_3$ is 1 greater than the number of bandpasses of this subset of bandpasses of $M_1$, 2b) or the path length has to be at least 5 and so the number of bandpasses of this subgroup of $B_3$ is at least $\frac{2}{3}$ of the number of bandpasses of this subset of bandpasses of $M_1$. It follows from 1), 2a) and 2b) that with respect to $B_3$, $M_1$ contains at least $\frac{2}{3}|B_3|$ corresponding bandpasses. That is,

$$w(M_1) \geq |B_1| + \frac{1}{2}|B_2| + \frac{2}{3}|B_3|. \quad (2.4)$$

Clearly, all bandpasses of $B_1$ are in the residual graph $G'$, while none of $B_1 \cup B_2 \cup B_3$ is in graph $G'$ since each one of them shares a 1 with at least a bandpass of $M_1$.

Note that the bandpasses of $B_2$ are paired up such that each pair of the two bandpasses share a 1 with a bandpass of $M_1$. Assume without loss of generality that these two bandpasses of $B_2$ are formed between rows $r_i$ and $r_j$ and between rows $r_k$ and $r_\ell$, respectively, and that the involved bandpass of $M_1$ is formed between rows $r_j$ and $r_k$ (see Figure 1). That is, in the optimal row permutation $\pi^*$, rows $r_i$ and $r_j$ are adjacent, and rows $r_k$ and $r_\ell$ are adjacent; while edge
Lemma 2 Assume edge \( (r_j, r_k) \in M_1 \), and that one bandpass of \( (r_j, r_k) \) shares 1 with two bandpasses of \( B_2 \). Then in graph \( G \) edge \( (r_j, r_k) \) is adjacent to at most four edges in the optimal row permutation \( \pi^* \), at most two of which are incident at row \( r_j \), and at most two of which are incident at row \( r_k \).

Proof. The lemma is straightforward from the above discussion, and the fact that edge \( (r_j, r_k) \) does not belong to the Hamiltonian path \( \pi^* \).

Continuing with the above discussion, assuming that edge \( (r_j, r_k) \in M_1 \), and that one bandpass of \( (r_j, r_k) \) shares 1 with two bandpasses of \( B_2 \), which are formed between rows \( r_i \) and \( r_j \) and between rows \( r_k \) and \( r_\ell \), respectively (see Figure 1). We know that in residual graph \( G' \), between rows \( r_i \) and \( r_k \), and rows \( r_j \) and \( r_\ell \), in the same column there is a bandpass (which contributes 1 towards the edge weight \( w'(i, \ell) \)). We call bandpasses identified in this way the induced bandpasses. From Lemma 2, edge \( (r_j, r_k) \) is adjacent to at most two edges of \( \pi^* \) incident at row \( r_j \). It follows that in residual graph \( G' \), row \( r_\ell \) can form induced bandpasses with at most four other rows. In other words, the subgraph of \( G' \) induced by the edges containing induced bandpasses, denoted as \( G'_s \), is a 4-matching in \( G' \).

Lemma 3 \( G'_s \) is a 4-matching in graph \( G \), and its weight \( w'(G'_s) \geq \frac{1}{2}|B_2| \).

Proof. From the above discussion, \( G'_s \) is a 4-matching in residual graph \( G' \). Since the edge set of \( G' \) and the edge set of \( G \) are the same, disregarding edge weights, \( G'_s \) is a 4-matching of graph \( G \). The second half of the lemma can be simply argued as follows. Since every pair of bandpasses of \( B_2 \) leads to an induced bandpass, all the edge weights in \( G'_s \) sum up to at least \( \frac{1}{2}|B_2| \), which is the number of bandpass pairs in \( B_2 \). This finishes the proof.

In \( O(n^{2.5}) \) time, a 4-matching such as \( G'_s \) can be decomposed into two 2-matchings \([11, 8]\), each of which is a collection of vertex-disjoint cycles or paths.

Lemma 4 Let \( C \) be a 2-matching of graph \( G \) such that no edge of \( M_1 \) is also an edge of \( C \). Then, we can partition the edge set of \( C \) into four matchings \( X_0, X_1, X_2, X_3 \) such that \( G[M_1 \cup X_j] \) is an acyclic 2-matching for all \( j \in \{0, 1, 2, 3\} \). Moreover, the partitioning takes \( O(n\alpha(n)) \) time, where \( \alpha(\cdot) \) is the inverse Ackerman function.

Proof. Hassin and Rubinstein \([12]\) have shown that we can compute two disjoint matchings \( X_0 \) and \( X_1 \) in \( C \) such that the following two conditions hold:

- Both \( G[M_1 \cup X_0] \) and \( G[M_1 \cup X_1] \) are acyclic 2-matchings of \( G \).
• Each vertex of $C$ is incident to at least one edge of $X_0 \cup X_1$.

For convenience, let $Y$ be the set of edges in $C$ but not in $X_0 \cup X_1$. By the second condition, $Y$ is a matching. Consider the graph $H = (V, M_1 \cup Y)$. Obviously, $H$ is a collection of vertex-disjoint paths and cycles, and each cycle of $H$ contains at least two edges of $Y$. For each cycle $C$ of $H$, we mark an arbitrary edge of $C$ that also belongs to $Y$. Let $X_3$ be the set of marked edges, and $X_2 = Y \setminus X_3$. Then, both $G[M_1 \cup X_2]$ and $G[M_1 \cup X_3]$ are acyclic 2-matchings of $G$.

It is not hard to see that with the famous union-find data structure [17], the computation of $X_0$ and $X_1$ described in [12] can be done in $O(n \alpha(n))$ time. Once knowing $X_0$ and $X_1$, we can obtain $X_2$ and $X_3$ in $O(n)$ time.

In general, Lemma 4 cannot be improved by partitioning the edge set of $C$ into three matchings instead of four matchings. To see this, it suffices to consider a concrete example, where $C$ is just a cycle of length 4 and $M_1$ consists of the two edges connecting nonadjacent vertices in $C$.

Let $C_1$ and $C_2$ denote the two 2-matchings constituting to the maximum weight 4-matching $C$ of residual graph $G'$. Using Lemma 4 alone, $C_1$ can be partitioned into four matchings $X_0, X_1, X_2, X_3$ and $C_2$ can be partitioned into four matchings $Y_0, Y_1, Y_2, Y_3$, such that $G[M_1 \cup Z_j]$ is an acyclic 2-matching for all $Z \in \{X, Y\}$ and $j \in \{0, 1, 2, 3\}$. The following lemma states a slightly better partition when we consider $C_1$ and $C_2$ simultaneously.

**Lemma 5** The weight of matching $M_2$ is $w'(M_2) \geq \frac{1}{15}|B_2|$.

**Proof.** Let $C_1$ and $C_2$ denote the two 2-matchings constituting to the maximum weight 4-matching $C$ of residual graph $G'$. Based on the discussion in the last paragraph, we firstly use Lemma 4 to partition the edge set of $C_1$ into four matchings $X_0, X_1, X_2, X_3$ and the edge set of $C_2$ into four matchings $Y_0, Y_1, Y_2, Y_3$, such that $G[M_1 \cup Z_j]$ is an acyclic 2-matching for all $Z \in \{X, Y\}$ and $j \in \{0, 1, 2, 3\}$.

Note that by Lemma 4, $X_2 \cup X_3$ is a matching and that $X_3$ contains the marked edges, each of which, say $e = (u, v)$, is the lightest edge of the corresponding cycle, say $C$, formed in $G[M_1 \cup X_2 \cup X_3]$. $C$ is an even cycle. If $C$ contains at least 6 edges, then $w'(X_3 \cap C) = w'(e) \leq \frac{1}{2}w'(X_2 \cap C)$. The following process is to swap certain edges among $X_0, X_1, X_2, X_3$ and $Y_0, Y_1, Y_2, Y_3$ to guarantee property (P) that each of $G[M_1 \cup X_i]$ for $i = 0, 1$ and $G[M_1 \cup Y_j]$ for $j \in \{0, 1, 2, 3\}$ is an acyclic 2-matching, and that $X_2 \cup X_3$ is a matching and $G[M_1 \cup X_2 \cup X_3]$ contains no length-4 cycles.

Let $C = (u, v, x, y)$ be a length-4 cycle in $G[M_1 \cup X_2 \cup X_3]$, and assume that $X_2 \cup X_3 = \{(u, v), (x, y)\}$. Then, we call edges $(u, v)$ and $(x, y)$ a problematic pair. Our swapping process is to resolve such problematic pairs. We distinguish three cases.

In the first case, edges $(u, x) \notin C_1$ and $(v, y) \notin C_1$.

Assume the other edges of $C_1$ incident at $u, v, x, y$ are $(u, 1), (v, 2), (x, 3), (y, 4)$, respectively. These four edges thus all belong to $G[M_1 \cup X_0]$ and $G[M_1 \cup X_1]$. If at least three of them belong to
\[G[M_1 \cup X_0], \text{ then in } G[M_1 \cup X_1] \text{ three vertices among } u, v, x, y \text{ have degree 1 and thus they cannot be in the same connected component of } G[M_1 \cup X_1]. \text{ We can move (exactly) one of edges } (u,v) \text{ and } (x,y) \text{ to } X_1, \text{ while maintaining property (P)}.\]

We examine next where exactly two of the four edges belong to \(G[M_1 \cup X_0].\) Assume without loss of generality that \((u,1) \in G[M_1 \cup X_0].\) If \((y,4) \in G[M_1 \cup X_0],\) then the connected component in \(G[M_1 \cup X_1]\) containing \(u\) has only one edge \((u,y),\) which belongs to \(M_1.\) Thus, if the other edge of \(C_1\) incident at vertex 1 belongs to \(X_1,\) we can move edge \((u,1)\) from \(X_0\) to \(X_2 \cup X_3,\) and move edge \((u,v)\) from \(X_2 \cup X_3\) to \(X_0;\) if the other edge of \(C_1\) incident at vertex 1 does not belong to \(X_1\) (and thus it must be in \(X_2 \cup X_3),\) we can move edge \((u,1)\) from \(X_0\) to \(X_1,\) and move edge \((u,v)\) from \(X_2 \cup X_3\) to \(X_0.\) Either way, we maintain property (P) while resolving a problematic pair of \(X_2 \cup X_3.\)

If \((v,2) \in G[M_1 \cup X_0],\) then vertices \(u\) and \(v\) have degree 1 in \(G[M_1 \cup X_1].\) Thus, if the other edge of \(C_1\) incident at vertex 1 does not belong to \(X_1,\) then vertex 1 has degree 1 in \(G[M_1 \cup X_1]\) as well. We conclude that vertices \(u,v,1\) cannot reside in the same connected component of \(G[M_1 \cup X_1].\)

When \(u\) and \(v\) are not connected, we can move edge \((u,v)\) from \(X_2 \cup X_3\) to \(X_1;\) when \(u\) and 1 are not connected, we can move edge \((u,1)\) from \(X_0\) to \(X_1,\) and move edge \((x,y)\) from \(X_2 \cup X_3\) to \(X_0.\) Again, either way, we maintain property (P) while resolving a problematic pair of \(X_2 \cup X_3.\) Symmetric scenarios can be argued in the same way for vertices 2, 3, 4. In the remaining scenario, the other edges of \(C_1\) incident at vertices 1, 2, 3, 4 all belong to \(X_0 \cup X_1.\) We then move edges \((u,1), (v,2), (x,3), (y,4)\) from \(X_0 \cup X_1\) to \(X_2 \cup X_3,\) and move edges \((u,v) ((x,y), \text{ respectively})\) from \(X_2 \cup X_2\) to \(X_0\) (\(X_3,\) respectively). Note that none of these four edges would form with any other edge into a problematic pair.

Lastly, if \((x,3) \in G[M_1 \cup X_0],\) then vertices \(u\) and \(x\) have degree 1 in \(G[M_1 \cup X_1].\) Thus, if the other edge of \(C_1\) incident at vertex 1 belongs to \(X_1,\) then vertex 1 has degree 1 in \(G[M_1 \cup X_1].\) We can move edge \((u,1)\) from \(X_0\) to \(X_2 \cup X_3,\) and move edge \((u,v)\) from \(X_2 \cup X_3\) to \(X_0.\) If the other edge of \(C_1\) incident at vertex 1 does not belong to \(X_1,\) then vertex 1 has degree 1 in \(G[M_1 \cup X_1]\) as well. We conclude that vertices \(u,x,1\) cannot reside in the same connected component of \(G[M_1 \cup X_1].\)

When \(u\) and 1 are not connected, we can move edge \((u,1)\) from \(X_0\) to \(X_1,\) and move edge \((u,v)\) from \(X_2 \cup X_3\) to \(X_0.\) Symmetric scenarios can be argued in the same way for vertices 2, 3, 4. In the remaining scenario, none of the other edges of \(C_1\) incident at vertices 1, 2, 3, 4 belongs to \(X_0 \cup X_1,\) and that vertices \(u \text{ and } 1\) \((v \text{ and } 2, x \text{ and } 3, y \text{ and } 4, \text{ respectively})\) are connected in \(G[M_1 \cup X_1]\) \((G[M_1 \cup X_0], G[M_1 \cup X_1], G[M_1 \cup X_0], \text{ respectively). It follows that we may move edge \((u,1)\) from \(X_0\) to \(X_1,\) move edge \((y,4)\) from \(X_1\) to \(X_0,\) and move edge \((u,v)\) from \(X_2 \cup X_3\) to \(X_0,\) to resolve the problematic pair.

In the second case, edges \((u,x) \notin C_1\) but \((v,y) \in C_1.\)

Assume the other edges of \(C_1\) incident at \(u, x\) are \((u,1), (x,3),\) respectively. These two edges and edge \((v,y)\) all belong to \(G[M_1 \cup X_0]\) and \(G[M_1 \cup X_1].\) Without loss of generality, assume \((v,y) \in X_1;\) it follows that vertices \(v\) and \(y\) have degree 1 in \(G[M_1 \cup X_0].\) If one of edges \((u,1)\) and \((x,3)\) does not belong to \(G[M_1 \cup X_0],\) say \((u,1),\) then we can move \((u,v)\) from \(X_2 \cup X_3\) to \(X_0,\) while maintaining property (P).

If both edges \((u,1)\) and \((x,3)\) belong to \(G[M_1 \cup X_0],\) then vertices \(u\) and \(x\) have degree 1 in
$G[M_1 \cup X_1]$. When the other edge of $C_1$ incident at vertex 1 does not belong to $X_1$ (but $X_2 \cup X_3$), we can move edge $(u, 1)$ from $X_0$ to $X_1$, and move edge $(u, v)$ from $X_2 \cup X_3$ to $X_0$; the symmetric scenario can be argued in the same way for vertex 3; When the other edge of $C_1$ incident at vertex 1 and the other edge of $C_1$ incident at vertex 3 both belong to $X_1$, we can move edges $(u, 1)$ and $(v, 3)$ from $X_0$ to $X_2 \cup X_3$, move edge $(v, y)$ from $X_1$ to $X_2 \cup X_3$, move edge $(u, v)$ from $X_2 \cup X_3$ to $X_0$, and move edge $(x, y)$ from $X_2 \cup X_3$ to $X_1$. Note that none of these three edges $(u, 1)$, $(v, 3)$ and $(v, y)$ would form with any other edge into a problematic pair.

In the last case, edges $(u, x) \in C_1$ and $(v, y) \in C_1$.

Assume without loss of generality that $(u, x) \in X_0$ and $(v, y) \in X_1$. Since $C_2$ do not share any edge with $C_1$, we consider the degrees of vertices $u, v, x, y$ in $G[M_1 \cup Y_1]$ for $i = 0, 1, 2, 3$. If in one of these four acyclic 2-matchings, say $G[M_1 \cup Y_0]$, at least three of the four vertices have degree 1, say $u, v, x$, then we can move edge $(u, v)$ from $C_1$ to $Y_0$, and thus the problematic pair of $X_2 \cup X_3$ is resolved. In the other cases, in each $G[M_1 \cup Y_i]$ for $i = 0, 1, 2, 3$, exactly two of the four vertices have degree 1.

Let the two edges of $C_2$ incident at $u$ ($v, x, y$, respectively) be $(u, 1)$ and $(u, 1')$ ($(v, 2)$ and $(v, 2')$, $(x, 3)$ and $(x, 3')$, $(y, 4)$ and $(y, 4')$, respectively).

If $(u, 1), (y, 4) \in Y_0$, then $u$ and $y$ both have degree 1 in one of $G[M_1 \cup Y_i]$ for $i = 1, 2, 3$, say in $G[M_1 \cup Y_3]$. It follows that if the other edge of $C_2$ incident at vertex 1 does not belong to $Y_3$, then we can move edge $(u, 1)$ from $Y_0$ to $Y_3$, and move edge $(u, v)$ from $C_1$ to $Y_0$ to resolve the problematic pair of $X_2 \cup X_3$; or if the other edge of $C_2$ incident at vertex 4 does not belong to $Y_3$, then we can move edge $(y, 4)$ from $Y_0$ to $Y_3$, and move edge $(x, y)$ from $C_1$ to $Y_0$ to resolve the problematic pair of $X_2 \cup X_3$. In the remaining scenario, the other edge of $C_2$ incident at vertex 1 (vertex 4, respectively) belongs to $Y_3$. Note that in either $G[M_1 \cup Y_1]$ or $G[M_1 \cup Y_2]$, vertex $u$ has degree 1, and we assume without loss of generality that vertex $u$ has degree 1 in $G[M_1 \cup Y_1]$. Note also that vertex 1 has degree 1 in $G[M_1 \cup Y_1]$. If edge $(y, 4') \not\in Y_1$, then vertex $y$ has degree 1 as well, and thus we can move edge $(u, 1)$ from $Y_0$ to $Y_1$, and move edge $(u, v)$ from $C_1$ to $Y_0$ to resolve the problematic pair of $X_2 \cup X_3$; if edge $(y, 4') \in Y_1$ but the other edge of $C_2$ incident at vertex 4 does not belong to $Y_3$, then we can move edge $(y, 4')$ from $Y_1$ to $Y_3$, move edge $(u, 1)$ from $Y_0$ to $Y_1$, and move edge $(u, v)$ from $C_1$ to $Y_0$ to resolve the problematic pair of $X_2 \cup X_3$. Therefore, we only need to argue the scenario where the other edge of $C_2$ incident at vertex 4' belongs to $Y_3$.

Symmetrically considering $Y_2$, we may assume without loss of generality that the other edge of $C_2$ incident at vertex 1' belongs to $Y_3$. Consequently, vertices $u, 1, 1'$ all have degree 1 in $G[M_1 \cup Y_1]$, and thus $u$ and at least one of 1 and 1' are not connected. If $u$ and 1 are not connected, we can move edge $(u, 1)$ from $Y_0$ to $Y_1$, and move edge $(u, v)$ from $C_1$ to $Y_0$ to resolve the problematic pair of $X_2 \cup X_3$; if $u$ and 1' are not connected, we can move edge $(u, 1')$ from $Y_2$ to $Y_1$, move edge $(u, 1)$ from $Y_0$ to $Y_2$, and move edge $(u, v)$ from $C_1$ to $Y_0$ to resolve the problematic pair of $X_2 \cup X_3$.

If $(u, 1), (v, 2) \in Y_0$, then $u$ and $v$ both have degree 1 in one of $G[M_1 \cup Y_i]$ for $i = 1, 2, 3$, say in $G[M_1 \cup Y_3]$. The following discussion is very similar to the above paragraph, though slightly simpler. Firstly, if $x$ and $y$ are not connected in $G[M_1 \cup Y_0]$ ($u$ and $v$ are not connected in $G[M_1 \cup Y_3]$, respectively), then we can move edge $(x, y)$ ($(u, v)$, respectively) from $C_1$ to $Y_0$ ($Y_3$, respectively) to directly resolve the problematic pair of $X_2 \cup X_3$. Secondly, if the other edge of $C_2$ incident at vertex 1 does not belong to $Y_3$, then we can move edge $(u, 1)$ from $Y_0$ to $Y_3$, and move edge $(x, y)$ from
contains no length-4 cycles. Subsequently, we let $X$ such that the last inequality follows from Lemma 3 and the fact that

$$10 \quad \text{Tong et al.}$$

Let Lemma 6 for $(p, q)$ in $X$ above two paragraphs. Repeatedly applying the above process to resolve the problematic pairs of

$$M \quad \text{by adding edges not in}$$

$$G \quad \text{degree 1 in}$$

$$Y \quad \text{x, that} \quad \text{(}$$

Note that $Y \quad \text{pair of} \quad X$$\text{G} \quad \text{has its weight greater than or equal to}$$$$\frac{1}{7.5} \left( w'(C_1) + w'(C_2) \right) \geq \frac{1}{7.5} \times \frac{1}{2} |B_2| = \frac{1}{15} |B_2|,$$

where the last inequality follows from Lemma 3 and the fact that $w'(C) \geq w'(G'_s)$.

$$\square$$

The next lemma says that Lemma 4 can be improved if the input 2-matching is acyclic.

**Lemma 6** Let $P$ be an acyclic 2-matching of $G$ such that no edge of $M_1$ is also an edge of $P$. Then, we can partition the edge set of $P$ into three matchings $Y_0, Y_1, Y_2$ such that $G[M_1 \cup Y_j]$ is an acyclic 2-matching for all $j \in \{0, 1, 2\}$. Moreover, the partitioning takes $O(n \alpha(n))$ time.

**Proof.** Note that $P$ is a collection of vertex-disjoint paths. We claim that if $P$ has two or more connected components, then we can connect the connected components of $P$ into a single path by adding edges not in $M_1$ to $P$. To see this claim, suppose that $P$ has two or more connected components. Obviously, we can connect the connected components of $P$ into a single path by adding edges to $P$. Unfortunately, some edges of $M_1$ may have been added to $P$. To remove edges of $M_1$ from $P$, we start at one endpoint of $P$ and process the edges of $P$ in order as follows:

- Let $s$ and $t$ be the current endpoints of $P$, and $(u, v)$ be the current edge we want to process. Without loss of generality, we may assume that the removal of $(u, v)$ from $P$ yields a path $P_u$ from $s$ to $u$ and another path $P_v$ from $v$ to $t$, and further assume that the edges of $P_u$ have been processed. Note that at most one of $s = u$ and $v = t$ is possible because $n \geq 3$. If $(u, v) \notin M_1$, then we proceed to process the other edge incident to $v$ than $(u, v)$. Otherwise, $(v, s) \notin M_1$ or $(u, t) \notin M_1$ because $M_1$ is a matching and at most one of $s = u$ and $v = t$ is possible. If $(v, s) \notin M_1$, then we modify $P$ by deleting edge $(u, v)$ and adding edge $(v, s)$ and
proceed to process the other edge incident to \( v \) than \((v, s)\). On the other hand, if \((u, t) \notin M_1\), then we modify \( \mathcal{P} \) by deleting edge \((u, v)\) and adding edge \((u, t)\) and proceed to process the other edge incident to \( t \) than \((u, t)\).

By the above claim, we may assume that \( \mathcal{P} \) is a single path \( \mathcal{P} = (v_1, v_2, \ldots, v_\ell+1) \), and denote \( e_j = (v_j, v_{j+1}) \) for \( j = 1, 2, \ldots, \ell \).

We next detail how to partition the edge set of \( \mathcal{P} \) into three required matchings \( Y_0, Y_1, \) and \( Y_2 \). Initially, we set \( Y_0 = \{e_1\}, Y_1 = \{e_2\}, \) and \( Y_2 = \{e_3\}. \) Then, for \( j = 4, 5, \ldots, \ell \) (in this order), we try to find a \( k \in \{0, 1, 2\} \) such that \( Y_k \cup \{e_j\} \) is a matching and \( G[M_1 \cup Y_k \cup \{e_j\}] \) is an acyclic 2-matching of \( G \). To explain how to find \( k \), fix an integer \( j \in \{4, 5, \ldots, \ell\} \). Let \( b \) be the integer in \{0, 1, 2\} with \( e_{j-1} \in Y_b \), and \( b' \) and \( b'' \) be the two integers in \{0, 1, 2\} \backslash \{b\}. If \( G[M_1 \cup Y_b] \) (respectively, \( G[M_1 \cup Y_{b'}] \)) contains no path between \( v_j \) and \( v_{j+1} \), then we can set \( k = b' \) (respectively, \( k = b'' \)) and we are done. So, we may also assume that \( G[M_1 \cup Y_{b'}] \) contains a path \( P' \) between \( v_j \) and \( v_{j+1} \) and \( G[M_1 \cup Y_{b''}] \) contains a path \( P'' \) between \( v_j \) and \( v_{j+1} \). See Figure 2.

Let \( v_{i'} \) (respectively, \( v_{i''} \)) be the neighbor of \( v_j \) in \( P' \) (respectively, \( P'' \)), and \( v_{h'} \) (respectively, \( v_{h''} \)) be the neighbor of \( v_{j+1} \) in \( P' \) (respectively, \( P'' \)). Then, none of edges \((v_{j-1}, v_j), (v_j, v_{j+1})\), and \((v_{j+1}, v_{j+2})\) can appear in \( P' \) (respectively, \( P'' \)), because \((v_{j-1}, v_j) \in Y_b \) and neither \((v_j, v_{j+1})\) nor \((v_{j+1}, v_{j+2})\) has been processed. So, all of \((v_{j-1}, v_{i'}), (v_{j+1}, v_{i''}), (v_j, v_{h'}), \) and \((v_{j+1}, v_{h''})\) belong to \( M_1 \). Thus, \( i' = i'' \) and \( h' = h'' \) because \( M_1 \) is a matching. Consequently, one edge incident to \( v_{i'} \) (respectively, \( v_{i''} \)) in \( \mathcal{P} \) belongs to \( Y_{i'} \) and the other belongs to \( Y_{i''} \). Hence, \( i' < j-1 \) and \( h' < j-1 \).

Since \( e_{j-1} \in Y_b \), either \( e_{j-2} \in Y_{i'} \) or \( e_{j-2} \in Y_{i''} \). We assume that \( e_{j-2} \in Y_{i'} \); the case where \( e_{j-2} \in Y_{i''} \) is similar. Since \( P'' \) is a path between \( v_j \) and \( v_{j+1} \) in \( G[M_1 \cup Y_{i''}] \), \( G[M_1 \cup Y_{i''}] \) contains no path between \( v_j \) and \( v_{j+1} \). Thus, \( G[M_1 \cup Y_{i'} \cup \{e_{j-1}\}] \) is an acyclic 2-matching of \( G \). Hence, we move \( e_{j-1} \) from \( Y_b \) to \( Y_{i'} \). A crucial point is that the degree of \( v_{i'} \) in \( G[M_1 \cup Y_b] \) is 1. This is true, because \( v_{i'} \) appears in both \( P' \) and \( P'' \) and in turn cannot be incident to an edge in \( Y_b \). By this crucial point and the fact that \( v_{i'} \) and \( v_j \) belong to the same connected component in \( G[M_1 \cup Y_b \cup \{e_j\}] \), we know that \( G[M_1 \cup Y_b \cup \{e_j\}] \) is an acyclic 2-matching of \( G \). Therefore, we can set \( k = b \).

![Figure 2: An illustration of moving \((v_{j-1}, v_j)\) from \( Y_b \) to \( Y_{i'} \) and adding \((v_j, v_{j+1})\) to \( Y_b \), where (1) the dashed lines indicate edges in \( M_1 \), (2) the thin solid lines indicate edges of \( \mathcal{P} \) that have not been processed, (3) the lines labeled with \( b \) (respectively, \( b' \) or \( b'' \)) indicate edges in \( Y_b \) (respectively, \( Y_{i'} \), or \( Y_{i''} \)), and (4) the two curves may contain edges of \( M_1 \).](image)

Obviously, with the famous union-find data structure \([17]\), the above partitioning of the edge set \( \mathcal{P} \) into \( Y_0, Y_1, Y_2 \) can be done in \( O(n\alpha(n)) \) time. \( \square \)

In general, Lemma 6 cannot be improved by partitioning the edge set of \( \mathcal{P} \) into two matchings.
Lemma 7 The weight of matching $M_2$ is $w'(M_2) \geq \frac{7}{27} |B_4|$.

**Proof.** Note that graph $G'$ contains all bandpasses of $B_4$, which is an acyclic 2-matching. From Lemma 1 we can compute a Hamiltonian path $P$ in $G'$ of weight at least $\frac{7}{9}$ of the optimum, and thus of weight at least $\frac{7}{27} |B_4|$. The above Lemma 6 guarantees that

$$w'(M_2) \geq \frac{1}{3} w'(P) \geq \frac{7}{27} |B_4|.$$  

\[\square\]

**Theorem 1** Algorithm APPROX is an $O(n^4)$-time $\frac{426}{227}$-approximation for the Bandpass problem.

**Proof.** The running time of algorithm APPROX is dominated by the computing for those maximum weight $b$-matchings, for $b = 1, 2, 4$, which can be done in $O(n^4)$ time. Since $M_1$ is the maximum weight matching in graph $G$, from Eq. (2.2) we have

$$w(M_1) \geq \frac{1}{2} p(\pi^*) \geq \frac{1}{2} \left( s_2(\pi^*) + \sum_{\ell=3}^{n} s_\ell(\pi^*)(\ell - 1) \right). \tag{2.5}$$

Combining Eqs. (2.4) and (2.5), we have for any real number $y \in [0, 1]$,

$$w(M_1) \geq \frac{y}{2} \left( s_2(\pi^*) + \sum_{\ell=3}^{n} s_\ell(\pi^*)(\ell - 1) \right) + (1 - y) \left( |B_1| + \frac{1}{2} |B_2| + \frac{2}{3} |B_3| \right). \tag{2.6}$$

The permutation $\pi$ produced by algorithm APPROX contains $b(\pi) \geq w(M_1) + w'(M_2)$ bandpasses, as indicated at the end of Section 2.1. From Lemmas 5 and 7 we have for any real number $x \in [0, 1]$,

$$b(\pi) \geq w(M_1) + x \frac{1}{15} |B_2| + (1 - x) \frac{7}{27} |B_4|. \tag{2.7}$$

Together with Eqs. (2.3) and (2.6), the above Eq. (2.7) becomes,

$$b(\pi) \geq w(M_1) + x \frac{1}{15} |B_2| + (1 - x) \frac{7}{27} |B_4| \geq \frac{y}{2} \left( s_2(\pi^*) + \sum_{\ell=3}^{n} s_\ell(\pi^*)(\ell - 1) \right) + (1 - y) \left( |B_1| + \frac{1}{2} |B_2| + \frac{2}{3} |B_3| \right) + x \frac{1}{15} |B_2| + (1 - x) \frac{7}{27} |B_4| \geq \frac{57}{142} \left( s_2(\pi^*) + \sum_{\ell=3}^{n} s_\ell(\pi^*)(\ell - 1) \right) + \frac{14}{213} |B_1| + \frac{28}{213} s_2(\pi^*). \tag{2.8}$$
where the last inequality is achieved by setting \( x = \frac{35}{71} \) and \( y = \frac{57}{71} \). Note that for all \( \ell \geq 3 \),
\[
(\ell - 1) \geq \frac{3}{2} \left\lfloor \frac{\ell}{2} \right\rfloor.
\]
It then follows from Eqs. (2.8) and (2.11) that
\[
b(\pi) \geq \frac{227}{426} \left( s_2(\pi^*) + \frac{171}{227} \times \frac{3}{2} \sum_{\ell=3}^{n} s_\ell(\pi^*) \left\lfloor \frac{\ell}{2} \right\rfloor \right) \geq \frac{227}{426} b(\pi^*).
\]
(2.9)

That is, the worst-case performance ratio of algorithm APPROX is at most \( \frac{426}{227} \).
\[\square\]

3 Conclusions and future work

In this paper, we presented a \( \frac{426}{227} \)-approximation algorithm for the Bandpass problem (\( \frac{426}{227} \approx 1.8767 \)), which improves the first non-trivial approximation (\( \approx 1.8948 \)) and a subsequent approximation (\( \approx 1.8805 \)). Our algorithm is based on maximum weight \( b \)-matchings, for \( b = 1, 2, 4 \), similar to tackling the closely related Max-TSP. The intrinsic structural property we proved for the optimal row permutation and the maximum weight matching is fundamental, without which no better lower bound on the optimum can be built. The schemes we developed to partition a \( b \)-matching, for \( b = 2 \) and \( b = 4 \), into desired matchings are also interesting, and could potentially be further improved.

For the Max-TSP, Serdyukov presented a \( \frac{4}{3} \)-approximation algorithm based on the maximum weight assignment (or called cycle cover) and the maximum weight matching [16], which has been improved to the currently best \( \frac{9}{7} \)-approximation algorithm in Lemma 11. We believe that the Bandpass problem can be better approximated by either improving the two key \( b \)-matching partitioning schemes presented in this paper, or introducing new structural properties and/or new techniques; yet we also believe that there will be a gap from \( \frac{9}{7} \), due to the “dynamic” edge weights.

On the other hand, Hassin and Rubinstein gave a randomized approximation algorithm for the Max-TSP with expected performance ratio \( \frac{33}{23} (= 1.32) \) [12] (which was subsequently de-randomized in [6]). It would be interesting to design a randomized approximation for the Bandpass problem too, with a better than 1.8767 expected performance ratio.

Acknowledgement

Weitian Tong, Randy Goebel, and Guohui Lin are supported in part by NSERC.

References

[1] R. P. Anstee. A polynomial algorithm for \( b \)-matching: An alternative approach. Information Processing Letters, 24:153–157, 1987.

[2] E. M. Arkin and R. Hassin. On local search for weighted packing problems. Mathematics of Operations Research, 23:640–648, 1998.
3. D. A. Babayev, G. I. Bell, and U. G. Nuriyev. The bandpass problem: combinatorial optimization and library of problems. *Journal of Combinatorial Optimization*, 18:151–172, 2009.

4. G. I. Bell and D. A. Babayev. Bandpass problem. In *Annual INFORMS meeting*, 2004. October 2004, Denver, CO, USA.

5. B. Chandra and M. M. Halldórsson. Greedy local improvement and weighted set packing approximation. In *ACM-SIAM Proceedings of the Tenth Annual Symposium on Discrete Algorithms (SODA’99)*, pages 169–176, 1999.

6. Z.-Z. Chen, Y. Okamoto, and L. Wang. Improved deterministic approximation algorithms for Max TSP. *Information Processing Letters*, 95:333–342, 2005.

7. Z.-Z. Chen and L. Wang. An improved approximation algorithm for the bandpass-2 problem. In *Proceedings of the 6th Annual International Conference on Combinatorial Optimization and Applications (COCOA 2012)*, volume 7402 of LNCS, pages 185–196, 2012.

8. R. Diestel. *Graph Theory*. Springer, 3rd edition, 2005.

9. H. Gabow. An efficient reduction technique for degree-constrained subgraph and bidirected network flow problems. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing (STOC’83)*, pages 448–456, 1983.

10. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W. H. Freeman and Company, San Francisco, 1979.

11. F. Harary. *Graph Theory*. Addison-Wesley, 1969.

12. R. Hassin and S. Rubinstein. Better approximations for Max TSP. *Information Processing Letters*, 75:181–186, 2000.

13. G. Lin. On the Bandpass problem. *Journal of Combinatorial Optimization*, 22:71–77, 2011.

14. D. L. Miller and J. F. Pekny. A staged primal-dual algorithm for perfect $b$-matching with edge capacities. *ORSA Journal on Computing*, 7:298–320, 1995.

15. K. E. Paluch, M. Mucha, and A. Madry. A 7/9 - approximation algorithm for the maximum traveling salesman problem. In *Proceedings of the 12th International Workshop on APPROX and the 13th International Workshop on RANDOM*, volume 5687 of LNCS, pages 298–311, 2009.

16. A. I. Serdyukov. An algorithms for with an estimate for the traveling salesman problem of the maximum. *Upravlyayemye Sistemy*, 25:80–86, 1984.

17. R. E. Tarjan. Efficiency of a good but not linear set union algorithm. *Journal of the ACM*, 22:215–225, 1975.

18. W. Tong, R. Goebel, W. Ding, and G. Lin. An improved approximation algorithm for the bandpass problem. In *Proceedings of the Joint Conference of the Sixth International Frontiers of Algorithmics Workshop and the Eighth International Conference on Algorithmic Aspects of Information and Management (FAW-AAIM 2012)*, LNCS 7285, pages 351–358, 2012.