Parameterized and Approximation Algorithms for Finding Two Disjoint Matchings

Zhi-Zhong Chen[1], Ying Fan[2], and Lusheng Wang[2]

[1] Division of Information System Design, Tokyo Denki University, Hatoyama, Saitama 350-0394, Japan
zzchen@mail.dendai.ac.jp
[2] Department of Computer Science, City University of Hong Kong, Tat Chee Avenue, Kowloon, Hong Kong SAR
yingying1988@gmail.com, lwang@cs.cityu.edu.hk

Abstract. We first present a fixed-parameter algorithm for the NP-hard problem of deciding if there are two matchings M_1 and M_2 in a given graph G such that $|M_1| + |M_2|$ is no less than a given number k. The algorithm runs in $O\left(m + k \cdot k! \cdot \left(2\sqrt{2}\right)^k \cdot n^2 \log n\right)$ time, where n (respectively, m) is the number of vertices (respectively, edges) in G. We then present a combinatorial approximation algorithm for the NP-hard problem of finding two disjoint matchings in a given edge-weighted graph G so that their total weight is maximized. The algorithm achieves an approximation ratio of roughly 0.76 and runs in $O\left(m + n^3\alpha(n)\right)$ time, where α is the inverse Ackermann function.

Keywords: Fixed-parameter algorithms, approximation algorithms, graph algorithms, matchings, NP-hardness.

1 Introduction

Throughout this paper, a graph means an undirected graph that may have parallel edges but no self-loops. A graph is *simple* if it has no parallel edges. A *matching* in a graph G is a set F of edges in G such that no two edges in F share an endpoint. A *maximum matching* in G is a matching in G whose cardinality is maximized over all matchings in G. Given a graph G, the *maximum matching problem* (MM for short) requires the computation of a maximum matching in G. MM is very fundamental in many areas and has been extensively studied in the literature.

In this paper, we consider a generalization of MM, called the *maximum two-matching problem* (MTM for short). Given a graph G, MTM requires the computation of two disjoint matchings in G whose total cardinality is maximized. Motivated by call admittance issues in satellite based telecommunication networks, Feige *et al.* [5] introduced MTM (among others) and showed its APX-hardness. They also observed that MTM is obviously a special case of the well-known *maximum coverage problem* (see [10]): We wish to cover the maximum number of edges of a given graph G with two sets each of which is a matching

P. Widmayer, Y. Xu, and B. Zhu (Eds.): COCOA 2013, LNCS 8287, pp. 1–12, 2013.

of G. Since this special case of the maximum coverage problem can be approximated by a greedy algorithm within a ratio of 0.75 [10], so can be MTM. They then gave a randomized approximation algorithm for MTM that achieves an expected ratio of $\frac{10}{13} \approx 0.769$. Their algorithm is based on an LP approach and random rounding. In particular, their LP has an exponential number of constraints and hence can only be solved by using the ellipsoid method together with a separation oracle. Hence, their algorithm is extremely slow although its running time is polynomial.

The *simple* case of MTM (SMTM for short) where the input graph is simple has been studied recently [5,12,3,2,11,1]. Feige et al. [5] gave a simple approximation algorithm for SMTM that achieves a ratio of 0.8. This ratio was then improved in a series of papers [12,3,2,11,1]. The best known ratio achieved by a polynomial-time approximation algorithm for SMTM is roughly 0.842 [1]. All known approximation algorithms for SMTM start by using Hartvigsen's polynomial-time algorithm [8] to compute a maximum-sized subgraph H of the input graph such that the degree of each vertex in H is at most 2 and there is no cycle of length 3 in H. Unfortunately, Hartvigsen's algorithm only works for simple graphs.

In this paper, we first consider the parameterized complexity of MTM. We show that MTM is fixed-parameter tractable by designing an algorithm that checks, in $O\left(m + k \cdot k! \cdot \left(2\sqrt{2}\right)^k \cdot n^2 \log n\right)$ time, if a given n-vertex m-edge graph G contains two disjoint matchings M_1 and M_2 such that $|M_1| + |M_2|$ is no less than a given number k. Our algorithm first reduces the problem for a given input (G, k) to the problem for (G', k) such that G' has a vertex set U with $|U| < k$ and $G' - U$ is edgeless. It then solves the problem for (G', k) with the help of Gabow's algorithm [7] for the maximum-weight degree-constrained subgraph problem.

We then consider the weighted version of MTM (MWTM for short), where each edge of the input graph G is given a nonnegative weight and the goal is to find two disjoint matchings whose total weight is maximized. MWTM is also a special case of the maximum coverage problem: We wish to cover the maximum-weight set of edges of a given graph G with two sets each of which is a matching of G. Since this special case of the maximum coverage problem can be approximated by a greedy algorithm within a ratio of 0.75 [10], so can be MWTM. However, all the ideas used in the known approximation algorithms [5,12,3,2,11,1] for MTM cannot be applied to MWTM because the algorithms call Hartvigsen's algorithm [8] which only works for nonweighted simple graphs.

We can observe that the algorithm of Feige *et al.* [5] for MTM can be slightly modified into a *randomized* approximation algorithm for MWTM that achieves an *expected* ratio of $\frac{10}{13} \approx 0.769$. However, as mentioned before, Feige *et al.*'s algorithm is extremely slow. So, in this paper, we present a completely new (*deterministic*) approximation algorithm for MWTM that achieves a ratio of roughly 0.76. Our new algorithm is combinatorial and runs in $O(m + n^3\alpha(n))$ time, where α is the inverse Ackermann function. The algorithm is motivated by the approaches developed in [14,9,4] for the *maximum traveling salesman*

problem which is the problem of finding a maximum-weight Hamiltonian cycle in a given edge-weighted complete graph.

Due to lack of space, some proofs are omitted.

2 Basic Definitions

Let G be a graph. We denote the vertex set of G by $V(G)$, and denote the edge set of G by $E(G)$. For a subset U of $V(G)$, $G - U$ denotes the graph obtained from G by removing the vertices in U (together with the edges incident to them). For a subset F of $E(G)$, $G - F$ denotes the graph obtained from G by removing the edges in F. The *degree* of a vertex v in G is the number of edges incident to v in G. Two edges of G are *adjacent* if they have at least one common endpoint.

A *cycle* in G is a connected subgraph of G in which each vertex is of degree 2. A *path* in G is either a single vertex of G or a connected subgraph of G in which exactly two vertices are of degree 1 and the others are of degree 2. The *length* of a cycle or path C is the number of edges in C and is denoted by $|C|$. A *k-cycle* is a cycle of length k. If the length of a cycle or path P is odd, then we say that P is *odd*; otherwise, we say that P is *even*. A *2-matching* of G is a subgraph H of G with $V(H) = V(G)$ in which the degree of each vertex is at most 2. Note that each connected component of a 2-matching is a path or cycle. A 2-matching $\mathcal{C}$ of G is *even* if each cycle in $\mathcal{C}$ is even. A *semi-path set* of G is a set F of edges in G such that each connected component of the graph $(V(G), F)$ is a path or a 2-cycle. A *matching* of G is a (possibly empty) set of pairwise nonadjacent edges of G. A *perfect matching* of G is a matching M of G such that each vertex of G is incident to an edge in M. An *independent set* of G is a set of vertices no two of which are adjacent in G.

3 The Parameterized Algorithm for MTM

Throughout this section, fix a graph G and a nonnegative integer k. We want to decide if G has two disjoint matchings M_1 and M_2 with $|M_1| + |M_2| \geq k$. In other words, we want to decide if G has an even 2-matching with at least k edges. To this end, we first perform the following five steps:

1. For each pair $\{u, v\}$ of vertices in G such that G has more than two edges between u and v, remove all but two edges between u and v from G. (*Comment:* This step removes redundant edges from G because a 2-matching of G uses at most two edges between each pair of vertices in G. After this step, G has $O\left(n^2\right)$ edges, where n is the number of vertices in G.)
2. Initialize $b = k$, $H = G$, and $\mathcal{C} = (V(G), \emptyset)$.
3. While $b > 0$ and H has at least one edge, perform the following two steps:
   (a) Add an arbitrary edge $\{u, v\}$ of H to $\mathcal{C}$, delete $\{u, v\}$ from H, and decrease b by 1.
   (b) Delete from H all edges e such that the graph obtained from $\mathcal{C}$ by adding e has a connected component that is not a path.

4. If $b = 0$, then output "yes" and halt.
5. Obtain a set I of vertices in $\mathcal{C}$ by initializing $I = \emptyset$ and then for each connected component P of $\mathcal{C}$, adding to I an arbitrary vertex of P whose degree in $\mathcal{C}$ is at most 1. (*Comment:* I contains all vertices of degree 0 in $\mathcal{C}$.)

Obviously, if our algorithm halts in Step 4, then $\mathcal{C}$ is an even 2-matching (indeed, a collection of vertex-disjoint paths) of G with k edges. So, for further discussion, we assume that our algorithm does not halt in Step 4.

Lemma 1. *I is an independent set of G.*

Lemma 2. *Let $U = V(G) - I$. Then, U contains at most $k - 1$ vertices.*

Our algorithm then constructs an edge-weighted graph by performing the following step:

6. Let $\mathcal{U}$ be the edge-weighted graph whose vertex set is U and whose edge set is constructed as follows.
   (a) For each edge e of G between two vertices of U, add e to $\mathcal{U}$ and assign a weight of 1 to e.
   (b) For each vertex $v \in I$ and for each (unordered) pair $\{u_1, u_2\}$ of distinct vertices in U such that both u_1 and u_2 are adjacent to v in G, add an edge between u_1 and u_2 to $\mathcal{U}$ and assign a weight of 2 to it.

Note that since $\mathcal{U}$ may have parallel edges, we need to assign distinct labels to the edges of $\mathcal{U}$ in order to distinguish them. So, each edge e of $\mathcal{U}$ has two endpoints, a weight, and a label. For convenience, we say that two even 2-matchings $\mathcal{C}_1$ and $\mathcal{C}_2$ of $\mathcal{U}$ are *the same* if ignoring the labels of edges of $\mathcal{C}_1$ and $\mathcal{C}_2$ yields the same graph. If P is a path or cycle in $\mathcal{U}$, then the *weight* of P is the total weight of edges in P. A 2-matching $\mathcal{C}$ in $\mathcal{U}$ is *even* if the weight of each cycle of $\mathcal{C}$ is even, and is *properly marked* if no vertex of degree at least 1 in $\mathcal{C}$ is marked but zero or more vertices of degree 0 in $\mathcal{C}$ are marked.

Lemma 3. *$\mathcal{U}$ has at most $(k-1)! \cdot \left(2\sqrt{2}\right)^{k-1}$ distinct properly marked even 2-matchings.*

Proof. A simple way of enumerating all properly marked even 2-matchings in $\mathcal{U}$ is as follows.

First, we enumerate all partitions of U into cyclically ordered subsets. It is widely known that there are exactly $|U|!$ such partitions. So, there are at most $(k-1)!$ such partitions for $|U| < k$.

Next, for each partition $\mathcal{P}$ of U into cyclically ordered subsets, we try all possible ways to transform $\mathcal{P}$ into a properly marked even 2-matching of $\mathcal{U}$. To see the details, let s be the number of singleton subsets in $\mathcal{P}$, and $S_1, \ldots, S_h$ be the nonsingleton subsets in $\mathcal{P}$. Consider an arbitrary $i \in \{1, \ldots, h\}$ and let $n_i = |S_i|$. Since S_i is cyclically ordered, we can view S_i as a cycle. Note that we can transform S_i into a path or a cycle of $\mathcal{U}$. To transform S_i into a cycle of $\mathcal{U}$, we have at most 2^{n_i} ways because S_i has n_i edges and we have at most two choices to handle each edge e with endpoints u_1 and u_2 in S_i as follows:

- If either $\mathcal{U}$ has no edge with endpoints u_1 and u_2, or $\mathcal{U}$ has only one edge with endpoints u_1 and u_2 but S_i is a 2-cycle, then there is no way to transform S_i into a cycle of $\mathcal{U}$.
- If all edges with endpoints u_1 and u_2 in $\mathcal{U}$ have the same weight, then the only choice is to let e have the same weight as the edges.
- If $\mathcal{U}$ has two edges f_1 and f_2 with endpoints u_1 and u_2 such that f_1 and f_2 have different weights (namely, 1 and 2), then we have two choices of assigning a weight (namely, 1 or 2) to e.

Note that even if we obtain a cycle C_i of $\mathcal{U}$ after handling each edge e of S_i as above, C_i may not be an even cycle and we just discard it if so.

To transform S_i into a path of $\mathcal{U}$, we first have n_i choices to break S_i into a path P_i, and then have at most 2^{n_i-1} ways to transform P_i into a path of $\mathcal{U}$ because P_i has n_i-1 edges and we have at most two choices to handle each edge e of P_i as in the case of transforming S_i into a cycle of $\mathcal{U}$.

In total, there are at most $2^{n_i} + 2^{n_i-1}n_i$ ways to transform S_i into a path or an even cyle of $\mathcal{U}$. Thus, in total, there are at most $\prod_{i=1}^{h}\left(2^{n_i} + 2^{n_i-1}n_i\right) \leq 2^{k-1-s}\prod_{i=1}^{h}\left(1+\frac{n_i}{2}\right)$ ways to transform $\mathcal{P}$ into an even 2-matching of $\mathcal{U}$, where the inequality holds because $\sum_{i=1}^{h} n_i \leq |U| - s < k - s$.

Recall that for each $i \in \{1,\ldots,h\}$, $n_i \geq 2$. Moreover, $\sum_{i=1}^{h} n_i \leq k-1-s$. We claim that these facts imply that $\prod_{i=1}^{h}\left(1+\frac{n_i}{2}\right) \leq 2^{\frac{k-1-s}{2}}$ if $k-1-s$ is even, while $\prod_{i=1}^{h}\left(1+\frac{n_i}{2}\right) \leq 2.5 \cdot 2^{\frac{k-4-s}{2}}$ if $k-1-s$ is odd. To see this claim, first note that for every even integer $m \geq 2$, $1+\frac{m}{2} \leq \left(1+\frac{2}{2}\right)^{\frac{m}{2}}$. Moreover, for every odd integer $m \geq 3$, $1+\frac{m}{2} \leq \left(1+\frac{2}{2}\right)^{\frac{m-3}{2}}\left(1+\frac{3}{2}\right)$. Thus, under the conditions that $n_1 \geq 2$, $\ldots$, $n_h \geq 2$, and $\sum_{i=1}^{h} n_i \leq k-1-s$, the value of $\prod_{i=1}^{h}\left(1+\frac{n_i}{2}\right)$ is maximized

- at $(n_1, n_2, \ldots, n_h) = (2, 2, \ldots, 2)$ if $k-1-s$ is even,
- at $(n_1, n_2, \ldots, n_h) = (3, 2, \ldots, 2)$ if $k-1-s$ is odd.

Therefore, the claim holds. By the claim, $\prod_{i=1}^{h}\left(1+\frac{n_i}{2}\right) \leq 2^{\frac{k-1-s}{2}}$ no matter whether $k-1-s$ is even or odd. Hence, there at most $\left(2\sqrt{2}\right)^{k-1-s}$ ways to transform $\mathcal{P}$ into an even 2-matching of $\mathcal{U}$. Obviously, for each properly marked even 2-matching $\mathcal{C}$ transformed from $\mathcal{P}$, there are exactly 2^s ways to properly mark $\mathcal{C}$. So, there are at most $\left(2\sqrt{2}\right)^{k-1-s} \cdot 2^s \leq \left(2\sqrt{2}\right)^{k-1}$ ways to transform $\mathcal{P}$ into a properly marked even 2-matching of $\mathcal{U}$. Now, since there are $(k-1)!$ $\mathcal{P}$'s in total, the lemma holds. ∎

Finally, our algorithm uses $\mathcal{U}$ to check if G has an even 2-matching with at least k edges by performing the following two steps:

7. For each properly marked even 2-matching $\mathcal{C}$ of $\mathcal{U}$, perform the following steps:
   (a) Construct an edge-weighted simple bipartite graph $B_{\mathcal{C}}$ as follows:

- The vertex set of $B_{\mathcal{C}}$ is $I \cup U_0 \cup U_1 \cup U_2$, where U_0 (respectively, U_1) consists of all $u \in U$ whose degree in $\mathcal{C}$ is 0 (respectively, 1) and $U_2 = \{u_e \mid e \text{ is an edge of weight 2 in } \mathcal{C}\}$.
- For each edge e of weight 2 in $\mathcal{C}$ and for each vertex $v \in I$ such that v is adjacent to both endpoints of e in G, $B_{\mathcal{C}}$ has an edge of weight 0 between v and u_e.
- For each edge e of G such that one endpoint of e is in I and the other is a vertex of U_1 or an unmarked vertex of U_0, $B_{\mathcal{C}}$ has an edge of weight 1 between the endpoints of e.
- For each 2-cycle C of G such that one vertex of C is in I and the other is a marked vertex of U_0, $B_{\mathcal{C}}$ has an edge of weight 2 between the vertices of C.

(b) Compute a maximum-weight subgraph $S_{\mathcal{C}}$ of $B_{\mathcal{C}}$ such that (1) the degree of each vertex $v \in I \cup U_1$ in $S_{\mathcal{C}}$ is at most 1, (2) the degree of u_e in $S_{\mathcal{C}}$ is exactly 1 for each edge e of weight 2 in $\mathcal{C}$, (3) the degree of each marked vertex $u \in U_0$ in $S_{\mathcal{C}}$ is exactly 1, and (4) the degree of each unmarked vertex $u \in U_0$ in $S_{\mathcal{C}}$ is at most 2.

(c) If $S_{\mathcal{C}}$ was found in Step 7b and the sum of the weights of $\mathcal{C}$ and $S_{\mathcal{C}}$ is at least k, then output "yes" and halt.

8. Output "no" and halt.

Lemma 4. *G has an even 2-matching with at least k edges if and only if our algorithm outputs "yes" in Step 7c for some properly marked even 2-matching $\mathcal{C}$ of $\mathcal{U}$.*

Theorem 1. *Given a nonnegative integer k and a graph G with n vertices and m edges, it takes $O\left(m + k \cdot k! \cdot \left(2\sqrt{2}\right)^k n^2 \log n\right)$ time to decide whether G has two disjoint matchings M_1 and M_2 such that $|M_1| + |M_2| \geq k$.*

4 The Approximation Algorithm for MWTM

Throughout this section, fix an instance (G, w) of MWTM, where G is an n-vertex m-edge graph and w is a function mapping each edge e of G to a nonnegative real number $w(e)$. After a simple $O(m + n^2)$-time preprocessing, we can assume that for every two vertices u and v of G, there are exactly two edges between u and v in G. To see that no generality is lost with this assumption, first observe that if there are three or more edges between two vertices u and v in G, then we can delete all but the heaviest two edges between u and v from G. On the other hand, if there is at most one edge between two vertices u and v in G, then we can add one or more edges of weight 0 between u and v so that G has exactly two edges between them. The *mate* of an edge e in G is the other edge in G that has the same endpoints as e. We may further assume that n is even, because if n is odd, we can add a new vertex and connect it to each of the other vertices with two edges of weight 0. For a subset F of $E(G)$, $w(F)$ denotes $\sum_{e \in F} w(e)$. The *weight* of a subgraph H of G is $w(H) = w(E(H))$.

Note that MWTM is equivalent to the problem of computing a maximum-weight even 2-matching of a given graph.

For a random event A, $\Pr[A]$ denotes the probability that A occurs. For two random events A and B, $\Pr[A \mid B]$ denotes the (conditional) probability that A occurs given the known occurrence of event B. For a random variable X, $\mathcal{E}[X]$ denotes the expected value of X.

In the remainder of this section, we first design a randomized approximation algorithm for MWTM and then derandomize it. Section 4.1 gives an outline of the randomized algorithm. Section 4.2 then describes the details that are missing in the outline. Section 4.3 estimates the time complexity and the expected approximation ratio achieved by the randomized algorithm. Finally, Section 4.4 derandomizes the algorithm.

4.1 Outline of the Algorithm

Our algorithm starts by computing a maximum-weight 2-matching $\mathcal{C}$ and a maximum-weight matching M of G. We may assume that M is a perfect matching of G because n is even. Our algorithm then uses $\mathcal{C}$ and M to perform a preprocessing as follows.

1. Construct a graph K as follows: Initially, K is the graph $(V(G), M)$. Next, for every 2-cycle C in $\mathcal{C}$, add the heavier edge of C to K. (*Comment:* Each connected component of K is a path, a 2-cycle, or an even cycle of length 4 or more.)
2. Modify $\mathcal{C}$ by performing the following step for every cycle C' of K with $|C'| \geq 4$:
   (a) Delete all 2-cycles C from $\mathcal{C}$ such that one edge of C appears in C'.
   (b) Add C' to $\mathcal{C}$.

Obviously, $\mathcal{C}$ remains to be a 2-matching of G after the preprocessing. So, the preprocessing does not increase $w(\mathcal{C})$ because $\mathcal{C}$ was originally a maximum-weight 2-matching of G. The preprocessing does not decrease $w(\mathcal{C})$ either, because otherwise we would be able to obtain a heavier matching of G than M by modifying it by deleting the edges added to $\mathcal{C}$ in the preprocessing while adding the edges deleted from $\mathcal{C}$ in the preprocessing.

We hereafter assume that our algorithm has done the preprocessing. If $\mathcal{C}$ is now even, then our algorithm just outputs $\mathcal{C}$ and stops. In the remainder of this paper, we assume that $\mathcal{C}$ is not an even 2-matching of G. Then, $\mathcal{C}$ has at least two connected components. Suppose that T is a maximum-weight even 2-matching of G. Let T_{int} denote the set of all edges $\{u, v\}$ of T such that some cycle C in $\mathcal{C}$ contains both u and v. Let T_{ext} denote the set of edges in T but not in T_{int}. Let $\beta = w(T_{\mathrm{int}})/w(T)$.

Our algorithm then computes three even 2-matchings T_1, T_2, T_3 of G, outputs the heaviest one among them, and stops. T_1 is computed by modifying the odd cycles in $\mathcal{C}$ as follows. Fix a parameter $0 < \epsilon < 1$. For each odd cycle C in $\mathcal{C}$, if C has more than ϵ^{-1} edges, then remove the minimum-weight edge; otherwise,

replace C by a maximum-weight even 2-matching of the graph obtained from G by deleting all vertices not in C. Then, $\mathcal{C}$ becomes an even 2-matching. Obviously, we have the following fact:

Fact 2. $w(T_1) \geq (1-\epsilon)w(T_{\text{int}}) = (1-\epsilon)\beta w(T)$.

When $w(T_{\text{ext}})$ is large, $w(T_{\text{int}})$ is small and $w(T_1)$ may be small, too. The two even 2-matchings T_2 and T_3 together are aimed at the case where $w(T_{\text{ext}})$ is large. Their computation is given in the next subsection.

4.2 Computation of T_2 and T_3

To compute T_2 and T_3, we first perform the following two steps:

1. Compute a maximum-weight matching M' in an auxiliary graph H, where $V(H) = V(G)$ and $E(H)$ consists of those $\{u, v\} \in E(G)$ such that u and v belong to different connected components of $\mathcal{C}$.
2. Fix an arbitrary ordering $C_1, \ldots, C_r$ of the connected components of $\mathcal{C}$ such that the 2-cycles precede the others. (*Comment:* Since $\mathcal{C}$ is a maximum-weight 2-matching of G and the weight of each edge in G is nonnegative, the weight of the edge between the endpoints of each path P that is a connected component of $\mathcal{C}$ is 0. So, we can change P into a cycle by adding the edge between its endpoints without changing its weight. In the remainder of this paper, for ease of explanation, we assume that each connected component of $\mathcal{C}$ is a cycle.)

We then process the cycles $C_1, \ldots, C_r$ in turn. Roughly speaking, the processing can be sketched as follows:

3. For $i = 1, 2, \ldots, r$ (in this order), process C_i by performing the following:
   (a) Mark some suitable edges $\{u, v\} \in M'$ with $\{u, v\} \cap V(C_i) \neq \emptyset$.
   (b) Move some suitable edges of C_i to M while always maintain that M is a semi-path set of G.

To detail Substeps 3a and 3b, we need several definitions and lemmas. In the remainder of this paper, for each integer $i \in \{1, \ldots, r\}$, the phrase "at time i" means the time at which $C_1, \ldots, C_{i-1}$ have been processed and C_i is the next cycle to be processed. For each integer $i \in \{1, \ldots, r\}$, let M_i be the set M at time i. For convenience, let M_{r+1} and $\mathcal{C}_{r+1}$ be the values of M and $\mathcal{C}$ immediately after Step 3, respectively.

A set F of edges in G is *available at time* i if $F \subseteq E(C_i)$, $F \cap M_1 = \emptyset$, and $M_i \cup F$ is a semi-path set of G. Since $M_1 \subseteq M_i$ and M_1 is a perfect matching of G, each set available at time i is a matching in C_i. A *matching-pair* in C_i is an (unordered) pair $\{A, B\}$ such that both A and B are (possibly empty) matchings in C_i. An *available matching-pair at time* i is a matching-pair $\{A, B\}$ in C_i such that both A and B are available at time i. A matching-pair $\{A, B\}$ in C_i *covers* a vertex u of C_i if at least one edge in $A \cup B$ is incident to u.

If C_i is a 2-cycle, then we can obtain an available matching-pair $\{A_i, B_i\}$ at time i that covers both vertices of C_i, by simply letting A_i consist of one of the edges of C_i and letting B_i consist of the other. $\{A_i, B_i\}$ is available because every cycle C_j of $\mathcal{C}$ with $j < i$ is a 2-cycle and we have done the preprocessing. Thus, if C_i is a 2-cycle, the details of Substeps 3a and 3b are as follows (i.e., they are replaced by the following three substeps):

(a) Compute an available matching-pair $\{A_i, B_i\}$ at time i by letting A_i consist of one of the edges of C_i and letting B_i consist of the other.
(b) For each $v \in V(C_i)$, if some edge e of M' is incident to v, then mark e.
(c) Select one of A_i and B_i uniformly at random and move its edges from $\mathcal{C}$ to M.

If C_i is a cycle of length 3 or more, then it is easy to modify the proof of Lemma 1 in [9] to obtain a subroutine for computing an available matching-pair at time i that covers all vertices of C_i. Moreover, using the famous union-find data structure, we can implement this subroutine in $O(|C_i| \cdot \alpha(n))$ time. The following lemma summarizes this result:

Lemma 5. *If C_i is a cycle of length 3 or more, then we can compute an available matching-pair $\{A_i, B_i\}$ at time i in $O(|C_i| \cdot \alpha(n))$ time that covers all vertices of C_i.*

A *maximal available set at time i* is a set F available at time i such that for every $e \in E(C_i) - F$, $F \cup \{e\}$ is not available at time i. Now, we are ready to describe the details of Substeps 3a and 3b for those cycles C_i of $\mathcal{C}$ with $|C_i| \geq 3$. Indeed, they are replaced by the following four substeps:

(a) Compute an available matching-pair $\{A_i, B_i\}$ at time i in $O(|C_i| \cdot \alpha(n))$ time that covers all vertices of C_i.
(b) Extend both A_i and B_i to maximal available sets at time i. (*Comment:* Using the famous union-find data structure, we can implement this substep in $O(|C_i| \cdot \alpha(n))$ time by scanning the edges of C_i in an arbitrary order and checking if each of them can be added to A_i and/or B_i.)
(c) For each vertex $v \in V(C_i)$ such that both A_i and B_i have an edge incident to v, if some edge e of M' is incident to v, then mark e.
(d) Select one of A_i and B_i uniformly at random and move its edges from $\mathcal{C}$ to M.

We finish computing T_2 and T_3 by performing the following three steps:

4. Add to $\mathcal{C}$ those edges $\{u, v\} \in M'$ such that both u and v are of degree at most 1 in $\mathcal{C}$. (*Comment:* Let M'_4 denote the set of edges in M' that are added to $\mathcal{C}$ at this step. For each cycle C in $\mathcal{C}$, $|E(C) \cap M'_4| \geq 2$.)
5. For each odd cycle C in $\mathcal{C}$, if $|E(C) \cap M'| = 2$ and exactly one edge in $E(C) \cap M'$ is marked, then delete one edge in $E(C) \cap M'$ from C at random in such a way that the marked edge is deleted with probability 2/3; otherwise, select one edge in $E(C) \cap M'$ uniformly at random and delete it from C. (*Comment*: Let M'_5 denote the set of edges in M' that remain in $\mathcal{C}$ immediately after this step.)
6. Set T_2 and T_3 to be $\mathcal{C}$ and the graph $(V(G), M)$, respectively.

4.3 Analysis of the Approximation Ratio

Our algorithm is clearly correct. We next analyze its approximation ratio.

Lemma 6. *Let F be an available set at time i. Suppose that $e_1 = \{u_1, u_2\}$ and $e_2 = \{u_2, u_3\}$ are two adjacent edges in C_i such that F contains no edge incident to u_1, u_2, or u_3. Then, $F \cup \{e_1\}$ or $F \cup \{e_2\}$ is available at time i.*

Corollary 1. *Suppose that F is a maximal available set at time i. Then, $C_i - F$ is a collection of vertex-disjoint paths each of length 1, 2, or 3.*

A matching-pair $\{A, B\}$ in C_i *favors* a vertex u of C_i if A contains an edge $e_1 \in E(C_i)$ incident to u and B contains an edge $e_2 \in E(C_i)$ incident to u (possibly $e_1 = e_2$). An available set F at time i is *dangerous* for an (unordered) pair $\{e_1, e_2\}$ of edges in M' if $C_i - F$ contains a connected component that is a length-2 path one of whose endpoints is an endpoint of e_1 and the other is an endpoint of e_2.

Lemma 7. *Let $\{A, B\}$ be an arbitrary matching-pair in C_i that covers all vertices of C_i. If A (respectively, B) is dangerous for a pair $\{e_1, e_2\}$ of edges in M', then $\{A, B\}$ favors exactly one endpoint of the length-2 path in $C_i - A$ (respectively, $C_i - B$) between an endpoint of e_1 and an endpoint of e_2.*

A *maximal available matching-pair at time i* is an available matching-pair $\{A_1, A_2\}$ at time i such that both A_1 and A_2 are maximal available sets at time i.

Lemma 8. *Let $e = \{v_1, v_2\}$ be an edge in M'. Then, $\Pr[e \in M'_{\overline{5}}] \geq \frac{1}{6}$.*

Recall T, T_{int}, T_{ext}, and β (they are defined in Section 4.1).

Lemma 9. *Let $\delta w(T)$ be the expected total weight of edges moved from $\mathcal{C}$ to M at Step 3. Then, $\mathcal{E}[w(T_3)] \geq (0.5 + \delta)w(T)$ and $\mathcal{E}[w(T_3)] \geq \left((1-\delta) + \frac{1}{12}(1-\beta)\right) w(T)$.*

Theorem 3. *For any fixed $\epsilon > 0$, there is an $O(m + n^3)$-time randomized approximation algorithm for* MWTM *achieving an expected approximation ratio of $\frac{19(1-\epsilon)}{25-24\epsilon}$.*

Proof. We first estimate the running time of our algorithm. The computation of $\mathcal{C}$, M, and M' can be done in $O(n^3)$ time [7]. Step 3 can be done in $O(n \cdot \alpha(n))$ total time. The other steps take $O(n)$ time. Thus, the time complexity is $O(n^3)$.

We next estimate its approximation ratio. By Fact 2 and Lemma 9, we have the following three inequalities:

$$w(T_1) \geq (1-\epsilon)\beta w(T), \tag{1}$$

$$\mathcal{E}[w(T_2)] \geq \left((1-\delta) + \frac{1}{12}(1-\beta)\right) w(T), \tag{2}$$

$$\mathcal{E}[w(T_3)] \geq (0.5+\delta)w(T). \tag{3}$$

Adding Inequalities 2 and 3, we have

$$\mathcal{E}[w(T_2)] + \mathcal{E}[w(T_3)] \geq \left(1.5 + \frac{1}{12}(1-\beta)\right) w(T). \tag{4}$$

Multiplying both sides of Inequality 4 by $12(1-\epsilon)$ and adding the resulting inequality to Inequality 1, we get

$$w(T_1) + 12(1-\epsilon)(\mathcal{E}[w(T_2)] + \mathcal{E}[w(T_3)]) \geq 19(1-\epsilon)w(T). \tag{5}$$

By Inequality 5, we have

$$\mathcal{E}[\max\{w(T_1), w(T_2), w(T_3)\}] \geq \frac{19(1-\epsilon)}{25-24\epsilon} \cdot w(T). \tag{6}$$

Therefore, the algorithm achieves an expected approximation ratio of $\frac{19(1-\epsilon)}{25-24\epsilon}$. ∎

4.4 Derandomization

The above randomized algorithm makes random choices only in Substep 3d and Step 5. To derandomize Step 5, we just modify it as follows:

5. For each odd cycle C in $\mathcal{C}$, delete one edge $e \in E(C) \cap M'$ from C such that the weight e is minimized over all edges in $E(C) \cap M'$.

When processing cycle C_i in Step 3, we need one random bit in Substep 3d. So, Step 3 needs r random bits in total. In the above analysis of the randomized algorithm, only the proof of Lemma 8 is based on the mutual independence between these random bits. Indeed, by carefully inspecting the proof, we can see that the proof is still valid even if the random bits are only pairwise independent. So, we can derandomize it via conventional approaches. Therefore, we have the following theorem:

Theorem 4. *For any fixed $\epsilon > 0$, there is an $O(m+n^3\alpha(n))$-time approximation algorithm for* MWTM *achieving a ratio of $\frac{19(1-\epsilon)}{25-24\epsilon}$.*

5 Open Problems

An obvious question is to ask if we can improve the running time of our parameterized algorithm for MTM. Preferably, we want a parameterized algorithm for MTM such that the exponent of the exponential factor in the time bound of the algorithm is linear in k. Another obvious question is to ask if we can design a combinatorial approximation algorithm for MWTM that achieves a better ratio than 0.769.

Feige *et al.* [5] have considered the problem of computing t disjoint matchings in a given graph G such that the total number of edges in the matchings is maximized, where t is a fixed positive integer. The special case of this problem where $t = 3$ and the input graph is *simple* has been considered by Kosowski [11] and Rizzi [13]. It seems interesting to design parameterized or approximation algorithms for this problem and its special cases.

References

1. Chen, Z.-Z., Konno, S., Matsushita, Y.: Approximating Maximum Edge 2-Coloring in Simple Graphs. Discrete Applied Mathematics 158, 1894–1901 (2010); A preliminary version appeared in Chen, B. (ed.) AAIM 2010. LNCS, vol. 6124, pp. 78–89. Springer, Heidelberg (2010)
2. Chen, Z.-Z., Tanahashi, R.: Approximating Maximum Edge 2-Coloring in Simple Graphs via Local Improvement. AAIM 2008 410, 4543–4553 (2009); A preliminary version appeared in Fleischer, R., Xu, J. (eds.) AAIM 2008. LNCS, vol. 5034, pp. 84–96. Springer, Heidelberg (2008)
3. Chen, Z.-Z., Tanahashi, R., Wang, L.: An Improved Approximation Algorithm for Maximum Edge 2-Coloring in Simple Graphs. Journal of Discrete Algorithms 6, 205–215 (2008); A preliminary version appeared in Kao, M.-Y., Li, X.-Y. (eds.) AAIM 2007. LNCS, vol. 4508, pp. 27–36. Springer, Heidelberg (2007)
4. Chen, Z.-Z., Wang, L.: An Improved Randomized Approximation Algorithm for Max TSP. Journal of Combinatorial Optimization 9, 401–432 (2005)
5. Feige, U., Ofek, E., Wieder, U.: Approximating Maximum Edge Coloring in Multigraphs. In: Jansen, K., Leonardi, S., Vazirani, V.V. (eds.) APPROX 2002. LNCS, vol. 2462, pp. 108–121. Springer, Heidelberg (2002)
6. Gabow, H.: Implementation of Algorithms for Maximum Matching on Nonbipartite Graphs. Ph.D. Thesis, Department of Computer Science, Stanford University, Stanford, California (1973)
7. Gabow, H.: 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 1983), pp. 448–456. ACM (1983)
8. Hartvigsen, D.: Extensions of Matching Theory. Ph.D. Thesis, Carnegie-Mellon University (1984)
9. Hassin, R., Rubinstein, S.: Better Approximation for Max TSP. Information Processing Letters 75, 181–186 (2000)
10. Hochbaum, D.: Approximation Algorithms for NP-Hard Problems. PWS Publishing Company, Boston (1997)
11. Kosowski, A.: Approximating the Maximum 2- and 3-Edge-Colorable Subgraph Problems. Discrete Applied Mathematics 157, 3593–3600 (2009)
12. Kosowski, A., Malafiejski, M., Zylinski, P.: Packing $[1, \Delta]$-Factors in Graphs of Small Degree. Journal of Combinatorial Optimization 14, 63–86 (2007)
13. Rizzi, R.: Approximating the Maximum 3-Edge-Colorable Subgraph Problem. Discrete Mathematics 309, 4166–4170 (2009)
14. Serdyukov, A.I.: An Algorithm with an Estimate for the Traveling Salesman Problem of Maximum. Upravlyaemye Sistemy 25, 80–86 (1984) (in Russian)