Improved Approximation Algorithms for (1,2)-TSP and Max-TSP Using Path Covers in the Semi-Streaming Model

Sharareh Alipour
Department of Computer Science, Tehran Institute for Advanced Studies (TeIAS), Khatam University, Tehran, Iran

Ermiya Farokhnejad
Department of Computer Science, University of Warwick, Coventry, UK

Tobias Mömke
Department of Computer Science, University of Augsburg, Germany

Abstract

We investigate semi-streaming algorithms for the Traveling Salesman Problem (TSP). Specifically, we focus on a variant known as the $(1,2)$-TSP, where the distances between any two vertices are either *one* or *two*. Our primary emphasis is on the closely related Maximum Path Cover Problem, which aims to find a collection of vertex-disjoint paths that covers the maximum number of edges in a graph. We propose an algorithm that, for any $\epsilon > 0$, achieves a $(\frac{2}{3} - \epsilon)$-approximation of the maximum path cover size for an n-vertex graph, using $\text{poly}(\frac{1}{\epsilon})$ passes. This result improves upon the previous $\frac{1}{2}$-approximation by Behnezhad et al. [3] in the semi-streaming model. Building on this result, we design a semi-streaming algorithm that constructs a tour for an instance of $(1,2)$-TSP with an approximation factor of $(\frac{4}{3} + \epsilon)$, improving upon the previous $\frac{3}{2}$-approximation factor algorithm by Behnezhad et al. [3][1].

Furthermore, we extend our approach to develop an approximation algorithm for the Maximum TSP (Max-TSP), where the goal is to find a Hamiltonian cycle with the maximum possible weight in a given weighted graph G. Our algorithm provides a $(\frac{7}{12} - \epsilon)$-approximation for Max-TSP in $\text{poly}(\frac{1}{\epsilon})$ passes, improving on the previously known $(\frac{1}{2} - \epsilon)$-approximation obtained via maximum weight matching in the semi-streaming model.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Theory of computation → Streaming, sublinear and near linear time algorithms; Theory of computation → Approximation algorithms analysis

Keywords and phrases $(1,2)$-TSP, Max-TSP, Maximum Path Cover, Semi-Streaming Algorithms, Approximation Algorithms, Graph Algorithms

Digital Object Identifier 10.4230/LIPIcs.STACS.2025.9

Related Version *Full Version*: `https://arxiv.org/abs/2501.04813` [2]

Funding *Tobias Mömke*: Partially supported by DFG Grant 439522729 (Heisenberg-Grant).

1 Introduction

The Traveling Salesman Problem (TSP) is a fundamental problem in combinatorial optimization. Given a graph $G = (V, E)$ with distances assigned to the edges, the objective is to find a Hamiltonian cycle with the lowest possible cost. The general form of TSP is known to be inapproximable unless $\mathsf{P} = \mathsf{NP}$ [22]. Consequently, research often focuses on specific types of

[1] Although Behnezhad et al. do not explicitly state that their algorithm works in the semi-streaming model, it is easy to verify.

distance functions, particularly the metric TSP, where distances satisfy the triangle inequality. Two notable metric versions of TSP are the graphic TSP, where distances correspond to the shortest path lengths in an unweighted graph, and $(1,2)$-TSP, a variant of TSP with distances restricted to either *one* or *two* [1, 4, 5, 6, 12, 14, 15, 17, 18, 19, 23, 24, 25].

Most research is conducted within the classic centralized model of computation. However, with the surge of large data sets in various real-world applications (as reviewed in [7]), there is a growing demand for algorithms capable of handling massive inputs. For very large graphs, classical algorithms are not only too slow but also suffer from excessive space complexity. When a graph's size exceeds the memory capacity of a single machine, algorithms that rely on random access to the data become impractical, necessitating alternative computational models. One such model that has gained significant attention recently is the graph stream model, introduced by Feigenbaum et al. [8, 9]. In this model, edges of the graph are not stored in memory but arrive sequentially in a stream, requiring processing in that order. The challenge is to design algorithms that use minimal space and ideally make only a small constant number of passes over the stream. A widely studied variant of this is the semi-streaming model. In the semi-streaming model, as outlined by Feigenbaum et al. [9], we consider a graph G with n vertices. The algorithm processes the graph's edges as they arrive in the stream and aims to compute results with minimal passes while using limited memory, constrained to $\tilde{O}(n) := O(n \cdot \text{polylog}(n))$.

It is straightforward to design a deterministic one-pass streaming algorithm to compute the cost of a Minimum Spanning Tree (MST) exactly, even in graph streams, which in turn immediately provides an $\tilde{O}(n)$ space algorithm to estimate TSP cost within a factor of 2. Thus, in the semi-streaming regime, the key challenge is to estimate TSP cost within a factor that is strictly better than 2. Recently, Chen, Khanna, and Tan [5] proposed a deterministic two-pass 1.96-approximation factor algorithm for metric TSP cost estimation in the semi-streaming model. For the case of $(1,2)$-TSP, using the approach of Behnezhad et al. [3], it is possible to provide a 1.5-approximation factor algorithm in the semi-streaming model. In [3], authors presented a sub-linear version of their algorithm; however, it is straightforward to implement their algorithm in the semi-streaming model.

In section 11 of [3], the authors showed a reduction from (1,2)-TSP to maximum matching and stated that achieving better approximation than 1.5 for (1,2)-TSP in the sub-linear model, solves an important open problem in sub-linear maximum matching. Considering the same reduction in semi-streaming model shows achieving non-trivial approximations for (1,2)-TSP in semi-streaming model is challenging. Since the maximum matching problem is studied in the semi-streaming model extensively, the following question naturally arises.

Question. What is the trade off between the approximation ratio and the number of passes for (1,2)-TSP in the semi-streaming model?

Maximum Path Cover. In an unweighted graph G, a subset of edges is called a *path cover* if it forms a union of vertex-disjoint paths. A maximum path cover (MPC[2]) in an unweighted graph is a path cover with the maximum number of edges (not paths) among all possible path covers in the graph. The problem of finding an MPC is known to be NP-complete. It is

[2] Throughout this paper we use this acronym for 'Maximum Path Cover'. Please note that we do **not** refer to the common abbreviation for 'Massively Parallel Computation'.

straightforward to see that a maximum matching provides a 1/2-approximation for MPC. Therefore, computing a maximal matching, which is a 1/2-approximation for maximum matching, yields a 1/4-approximate solution for MPC.

Behnezhad et al. [3] developed a 1/2-approximate MPC algorithm, which provides a 1.5-approximate solution for $(1,2)$-TSP. Their algorithm can be implemented in one pass within the semi-streaming model using $\tilde{O}(n)$ space to return the cost, and in two passes if the approximate solution itself is required. Our primary contribution is an improvement in the approximation factor of their algorithm.

Result 1 (Formally as Theorem 8). For a given unweighted graph G, there is a semi-streaming algorithm that returns a $(\frac{2}{3}-\epsilon)$-approximation of MPC in poly$(\frac{1}{\epsilon})$ passes.

$(1,2)$-TSP. The classical problem $(1,2)$-TSP is well-studied and known to be NP-hard [15], and even APX-hard [20]. One can easily observe that in an instance of $(1,2)$-TSP, the optimal tour is almost the same as finding the MPC of the induced subgraph on edges with weight 1 and then joining their endpoints with edges with weight 2, except for a possible difference of 1 (in the case that there exists a Hamiltonian cycle all of whose edges have weight 1). A simple computation shows that if one can find a set of vertex-disjoint paths that is at least α times the optimal size ($\alpha \leq 1$), then one can also find a tour whose cost is no more than $(2-\alpha)$ times the optimal cost for $(1,2)$-TSP. Thus, Result 1 implies the following result.

Result 2 (Formally as Theorem 10). For an instance of $(1,2)$-TSP, there is a semi-streaming algorithm that returns a $(\frac{4}{3}+\epsilon)$-approximation of $(1,2)$-TSP in poly$(\frac{1}{\epsilon})$ passes.

In the second part of the paper, we examine Max-TSP in the semi-streaming model.

Max-TSP. For a given complete weighted graph G, the goal of Max-TSP is to find a Hamiltonian cycle such that the sum of the weights of the edges in this cycle is maximized.

It is evident that a maximum weighted matching provides a $\frac{1}{2}$-approximation for the cost of Max-TSP. Consequently, the result of Huang and Saranurak [13], which computes a $(1-\epsilon)$-approximate maximum weight matching in the semi-streaming model, yields a $(\frac{1}{2}-\epsilon)$-approximation for Max-TSP. In this paper, we improve this bound to $\frac{7}{12}-\epsilon$. Our result is as follows.

Result 3 (Formally as Theorem 16). For a given weighted graph G, there is a semi-streaming algorithm that returns a $(\frac{7}{12}-\epsilon)$-approximation of Max-TSP in poly$(\frac{1}{\epsilon})$ passes.

To the best of our knowledge, this is the first non-trivial approximation algorithm for Max-TSP in the semi-streaming model.

Further related work

Our approach for computing MPC, $(1,2)$-TSP and Max-TSP mainly uses the subroutines for computing maximum matching in unweighted graphs and maximum weight matching in weighted graphs.

In the semi-streaming model, Fischer, Mitrovic and Uitto [10] gave a $(1-\epsilon)$-approximation for the maximum matching problem in $\text{poly}(1/\epsilon)$ passes. This result was an improvement over the $(1/\epsilon)^{O(1/\epsilon)}$ passes algorithm by McGregor [16].

For the maximum weight matching in the semi-streaming model, Paz and Schwartzman gave a simple deterministic single-pass $(1/2-\epsilon)$-approximation algorithm [21]. Gamlath, Kale, Mitrovic, and Svensson gave a $(1-\epsilon)$-approximation streaming algorithm that uses $O_\epsilon(1)$ passes and $O_\epsilon(n \cdot \text{poly}(\log n))$ memory. This was the first $(1-\epsilon)$-approximation streaming algorithm for weighted matching that uses a constant number of passes (only depending on ϵ) [11]. Also, Huang and Su in [13], gave a deterministic $(1-\epsilon)$-approximation for maximum weighted matching using $\text{poly}(1/\epsilon)$ passes in the semi-streaming model. When ϵ is smaller than a constant $O(1)$ but at least $1/\log^{o(1)} n$, their algorithm is more efficient than [11].

1.1 Notation

Let G be a simple graph. We denote the set of vertices and edges of G by $V(G)$ and $E(G)$ respectively. We also denote the maximum matching size in G by $\mu(G)$ and the size of the MPC in G by $\rho(G)$.

For a subset of edges $T \subseteq E(G)$, we denote G/T as the contraction of G on T, which is the graph derived by repeatedly removing edges of T (it is well-known that the order does not matter) from the graph and merging its endpoints to be a single node in the new graph. Note that after contraction the graph might have parallel edges, but this does not interfere with our algorithm. In the weighted case, if $w(e)$ is the weight of edge e, then we define $w(T)$ to be the sum of weights of the elements of T i.e. $\sum_{e \in T} w(e)$. Let $P = (u_1, u_2, \ldots, u_k)$ be a path of length $k-1$ ($u_i \in V$ for $1 \leq i \leq k$). We call u_1 and u_k *end points* of P and u_i for $2 \leq i \leq k-1$ *middle points* of P.

2 Technical Overview and Our Contribution

We propose a simple algorithm that constructs a path cover with an approximation factor of almost $\frac{2}{3}$ for MPC. This new algorithm merely depends on basic operations and computing matching and approximate matching.

Our algorithm for MPC is as follows. Assume that MMϵ is a $(1-\epsilon)$-approximation algorithm for computing maximum matching in an unweighted graph G. We use MMϵ as a subroutine in our algorithm, which has two phases. In the first phase, we run MMϵ to compute a $(1-\epsilon)$-approximate maximum matching, denoted by M_1. In the next phase, we contract all edges in M_1 to obtain a new graph $G' = G/M_1$. Then, we compute a $(1-\epsilon)$-approximate maximum matching, denoted by M_2, for G' using MMϵ. Finally, we return the edges of M_1 and M_2 as a path cover for G (see Algorithm 1).

We will show that the output of our algorithm is a collection of vertex-disjoint paths, i.e., a valid path cover (see Lemma 1).

The algorithm is simple, but proving that its approximation factor is $\frac{2}{3}-\epsilon$ is challenging. As a warm-up, it is straightforward to see that by computing a maximum matching in the first phase, we achieve a $\frac{1}{2}$-approximate MPC. However, the challenge lies in the second phase, which helps to improve the approximation factor.

Now we explain the idea of our proof to find the approximation factor of Algorithm 1. For the matching M_1 in graph G, we provide a lower bound for $\mu(G/M_1)$. We show that if we consider a maximum path cover P^* and contract P^* on M_1, the contracted graph becomes a particular graph in which we can find a lower bound on the size of its maximum matching. Let M_2 be a maximum matching of G/M_1, then this results in a lower bound for $|M_2|$. Finally, we exploit this lower bound for $|M_2|$ together with a lower bound for $|M_1|$, to come up with the approximation factor of Algorithm 1.

We explain how to implement this algorithm in the semi-streaming model, achieving an improved approximation factor for $(1,2)$-TSP within this model.

For the Max-TSP, we use a similar algorithm, except we compute maximum weight matching instead of maximum matching (see Algorithm 3). By computing the approximation factor of this algorithm, we provide a non-trivial approximation algorithm for Max-TSP in the semi-streaming model. Despite the extensive study of the weighted version of the maximum matching problem, Max-TSP has not been studied extensively in the literature within the semi-streaming model. One reason could be that it is not possible to extend the approaches for the unweighted version to the weighted version. Fortunately, we can extend our algorithm to the weighted version and improve the approximation factor of Max-TSP in the semi-streaming model. However, our method for analyzing the approximation factor of Algorithm 1 does not apply to the weighted version, so we present a different proof approach for computing the approximation factor of Algorithm 3.

3 Improved Approximation Factor Semi-Streaming Algorithm for MPC

In Algorithm 1, we presented our novel algorithm for MPC. This section provides an analysis of its approximation factor, followed by a detailed explanation of its streaming implementation.

Algorithm 1 Approximating maximum path cover on a graph G.

```
1: Run $\mathrm{MM}_\epsilon$ on $G$ to find a matching $M_1$.
2: Contract $G$ on $M_1$ to get a new graph $G' = G/M_1$.
3: Run $\mathrm{MM}_\epsilon$ on $G'$ to find another matching $M_2$.
4: return $M_1 \cup M_2$.
```

We start by proving the correctness of this algorithm.

▶ **Lemma 1.** *If M_1 and M_2 are the matchings obtained in Algorithm 1, then $M_1 \cup M_2$ forms a path cover for G.*

Proof. We claim that $M_1 \cup M_2$ is a vertex-disjoint union of paths of length $1, 2$ or 3. As a result, it is a path cover. Suppose $M_1 = \{u_1v_1, u_2v_2, \dots, u_kv_k\}$. Let us denote the vertices of G/M_1 by $\{(uv)_1, (uv)_2, \dots, (uv)_k, w_1, w_2, \dots, w_l\}$ where $(uv)_i$ represents the vertices u_i and v_i, merged in the contracted graph G/M_1 and w_j's for $1 \le j \le l$ are the rest of the vertices. Let $xy \in M_2$ be an arbitrary edge. By symmetry between x and y, there are three cases as follows:

1. $x, y \in \{w_1, w_2, \dots, w_l\}$.
   In this case, x and y are intact vertices after contraction, which means there are no edges in M_1 adjacent to x and y. Since $xy \in M_2$ and M_2 is a matching, there are no other edges in $M_1 \cup M_2$ adjacent to x and y in G. As a result, xy would be a path of length 1 in $M_1 \cup M_2$.
2. $x \in \{(uv)_1, (uv)_2, \dots, (uv)_k\}$ and $y \in \{w_1, w_2, \dots, w_l\}$.
   In this case, $x = (uv)_i$ for some $1 \le i \le k$. As a result, xy would be u_iy or v_iy in G. By symmetry, assume that $xy = u_iy$ in G. Since M_1 is a matching, no other edges in M_1 are adjacent to u_i and v_i. No edge in M_1 is adjacent to y. Since M_2 is a matching, the only edge in M_2 adjacent to at least one of u_i, v_i and y in G is $xy = u_iy$. Finally, we can see that (v_i, u_i, y) is a path of length 2 in $M_1 \cup M_2$.

3. $x, y \in \{(uv)_1, (uv)_2, \dots, (uv)_k\}$.
   In this case, let $x = (uv)_i$ and $y = (uv)_j$ and by symmetry, assume that xy is the edge connecting u_i to u_j in G. Since M_1 is a matching, the only edges in M_1 adjacent to at least one of u_i, u_j, v_i, v_j are u_iv_i and u_jv_j. Since M_2 is a matching, the only edge in M_2 adjacent to at least one of u_i, u_j, v_i, v_j is $(uv)_i(uv)_j$. As a result, (v_i, u_i, u_j, v_j) would be a path of length 3 in $M_1 \cup M_2$.

So, $M_1 \cup M_2$ is a union of vertex-disjoint paths of length $1, 2$ or 3. ◀

3.1 Analysis of the Approximation Factor of Algorithm 1

We start with a simple and basic lemmas that is crucial in our proof.

▶ **Lemma 2.** *Let G be an arbitrary graph. We have:*

$$\rho(G) \geq \mu(G) \geq \frac{1}{2}\rho(G).$$

Proof. Since every matching is a path cover, we have $\rho(G) \geq \mu(G)$. Also, given an MPC, we can select every other edge in this MPC to obtain a matching that contains at least half of its edges, which implies $\mu(G) \geq \frac{1}{2}\rho(G)$. ◀

▶ **Corollary 3.** *If M is a $(1-\epsilon)$-approximation of maximum matching in graph G, then*

$$|M| \geq \frac{1}{2}(1-\epsilon)\rho(G).$$

We now present a lemma regarding the size of the maximum matching in a specific type of graph. This lemma may be of independent interest. In this paper we utilize this lemma on G/M_1 to derive a lower bound for $|M_2|$.

▶ **Lemma 4.** *Assume G is a graph without loops such that each vertex v of G has degree 1,2, or 4. If $V_4(G)$ denotes the set of vertices of degree 4 in G, then we have*

$$\mu(G) \geq \frac{|E(G)| - |V_4(G)|}{3}.$$

Proof. The proof of this lemma is a bit long and technical. Here, we provide the proof sketch of the lemma due to page limit constraints and we refer the reader to the full version of the paper to see the complete proof.[3] The main tools for the proof are induction and a charging scheme. The induction is on the number of edges between nodes of degree 2 or 4 in graph G denoted by $|E_{2,4}(G)|$.

In a high level, we provide some special local topological properties of the graph. If G does not satisfy at least one of the properties, then we do some delicate local changes on G to get G' such that $|E_{2,4}(G')| \leq |E_{2,4}(G)|$. Next, we improve a matching in G' to a matching for G with the desired size.

In the case that G satisfies all of the properties, we provide a charging scheme that takes some tokens for each edge in an arbitrary maximum matching and spread these tokens between incident edges, which proves the desired inequality. ◀

Using above lemma, we have the main lemma of this section as follows.

[3] A full version of this extended abstract can be found at [2].

▶ **Lemma 5.** *If M is an arbitrary matching in a graph G, then*

$$\mu(G/M) \geq \frac{\rho(G) - |M|}{3}.$$

Proof. Assume P^* is a maximum path cover in G such that $P^* \cap M$ is maximal. We claim that every $e \in M \setminus P^*$ connects two middle points of P^*. The proof of this claim follows from a case by case argument. For the sake of contradiction, assume $e = uv \in M \setminus P^*$ does not connect two middle points of P^*. We have three cases for u and v as follows.

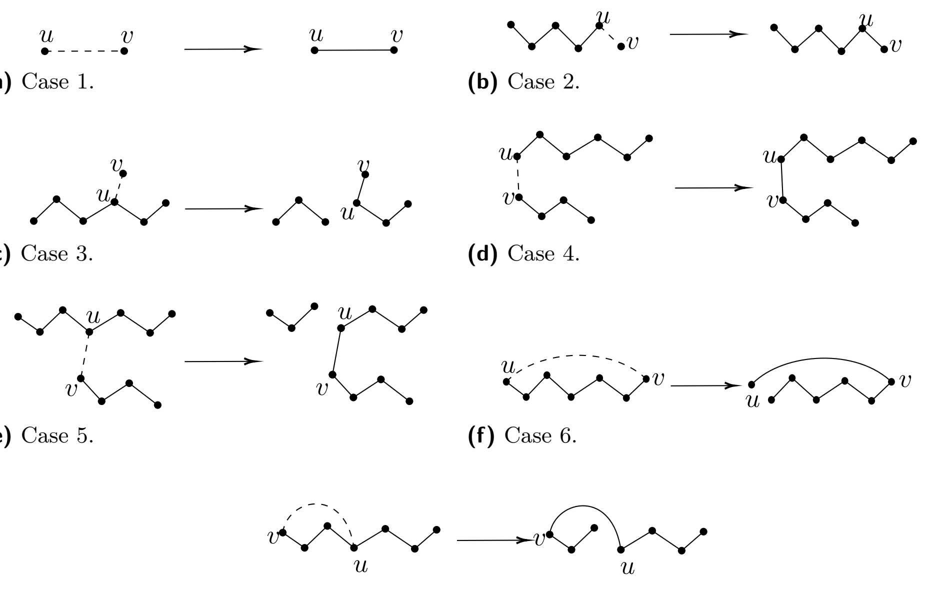
- None of u and v belong to P^* (case 1).
- Exactly one of them (say u) belongs to P^*. Then, u is an end point (case 2), or u is a middle point (case 3).
- Both u and v belong to P^*. Then, we have two sub cases.

  u and v are on different paths. Then either they are both end points (case 4), or one is a middle point (say u) and the other one is an end point (case 5). Note that we have considered that both of u and v are not middle points at the same time.

  u and v belong to the same path. Then, either they are both end points (case 6), or one is a middle point (say u) and the other one is an end point (case 7). Note that we have assumed both of them are not middle points at the same time.

Now, we explain each case in detail.

1. Neither u nor v belongs to $\tilde{P}$.
   This case is impossible because $P^* + e$ is a path cover with a size larger than $|P^*|$, which is in contradiction with P^* being MPC (see Figure 1a).
2. u is an end point of a path in P^* and v is not contained in P^*.
   Again, this case is impossible since $P^* + e$ is a path cover with a size larger than $|P^*|$, which is in contradiction with P^* being MPC (see Figure 1b).
3. u is a middle point of a path in P^* and v is not contained in P^*.
   Let $(p_1, p_2, \dots, p_k)$ be the path in P^* containing $u = p_i$. Replace P^* by $P^* - p_{i-1}p_i + e$ which is an MPC of G (see Figure 1c). Since $e \in M$, we have $p_{i-1}p_i \notin M$. Therefore, $|\tilde{P} \cap M|$ increments. This is in contradiction with $|P^* \cap M|$ being maximal.
4. u and v are end points of different paths in P^*.
   In this case, let $(p_1, p_2, \dots, p_k)$ and $(q_1, q_2, \dots, q_l)$ be the paths in P^* containing $u = p_1$ and $v = q_1$, respectively. $P^* + e$ would be a path cover of size greater than $|P^*|$ which is in contradiction with P^* being MPC (see Figure 1d).
5. u and v are the middle and end points of different paths in P^*, respectively.
   In this case, let $(p_1, p_2, \dots, p_k)$ and $(q_1, q_2, \dots, q_l)$ be the paths in P^* containing $u = p_i$ and $v = q_1$ respectively. Replace P^* by $P^* - p_{i-1}p_i + e$ which is an MPC of G (see Figure 1e). Since $e \in M$, we have $p_{i-1}p_i \notin M$. Therefore, $|P^* \cap M|$ increments. This is in contradiction with $|P^* \cap M|$ being maximal.
6. u and v are end points of the same path in P^*.
   In this case, let $(p_1, p_2, \dots, p_k)$ be the path in P^* containing $u = p_1$ and $v = p_k$. Replace P^* by $P^* - p_1p_2 + e$ which is an MPC of G (see Figure 1f). Since $e \in M$ we have $p_1p_2 \notin M$. Therefore, $|P^* \cap M|$ increments. This is in contradiction with $|P^* \cap M|$ being maximal.
7. u and v are the middle and end points of the same path in P^*, respectively.
   In this case, let $(p_1, p_2, \dots, p_k)$ be the path in P^* containing $u = p_i$ and $v = p_1$ (since $e \notin P^*$, we have $2 < i$). Replace P^* by $P^* - p_{i-1}p_i + e$ which is an MPC of G (see Figure 1g). Since $e \in M$, we have $p_{i-1}p_i \notin M$. Therefore, $|P^* \cap M|$ increments. This is in contradiction with $|P^* \cap M|$ being maximal.



(a) Case 1. **(b)** Case 2. **(c)** Case 3. **(d)** Case 4. **(e)** Case 5. **(f)** Case 6. **(g)** Case 7.

Figure 1 Different possible cases for u and v.

Each case leads to a contradiction, implying that every $e \in M \setminus P^*$ connects two middle points of P^*.

Now, contraction of each $e \in M \setminus P^*$ makes a vertex of degree 4 in P^*/M. Contraction of each $e \in M \cap P^*$ makes a vertex of degree 2 and decrements the number of edges in P^*. As a result, P^*/M is a graph whose vertices' degrees are 1,2 or 4, $|E(P^*/M)| = |P^*| - |P^* \cap M|$ and $|V_4(P^*/M)| = |M \setminus P^*|$. Finally, using Lemma 4 for P^*/M we have

$$\mu(P^*/M) \geq \frac{|E(P^*/M)| - |V_4(P^*/M)|}{3} = \frac{|P^*| - |P^* \cap M| - |M \setminus P^*|}{3} = \frac{|P^*| - |M|}{3}.$$

Since P^*/M is a subgraph of G/M we have

$$\mu(G/M) \geq \mu(P^*/M) \geq \frac{|P^*| - |M|}{3} = \frac{\rho(G) - |M|}{3}.$$ ◀

Using the above results, we compute the approximation factor of Algorithm 1.

▶ **Theorem 6.** *The approximation factor of Algorithm 1 is $\frac{2}{3}(1-\epsilon)$, i.e.,*

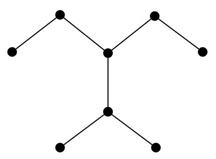
$$\rho(G) \geq |M_1 \cup M_2| \geq \frac{2}{3}(1-\epsilon)\rho(G). \tag{1}$$

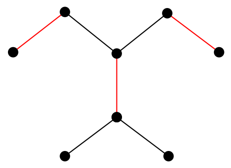
Proof. By Corollary 3 and, Lemma 5, we have

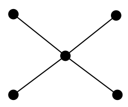
$$\begin{aligned}
|M_1 \cup M_2| &= |M_1| + |M_2| \\
&\geq |M_1| + (1-\epsilon)\mu(G/M_1) \\
&\geq |M_1| + \frac{1-\epsilon}{3}(\rho(G) - |M_1|) \\
&\geq \frac{1-\epsilon}{3}\rho(G) + \frac{2}{3}|M_1| \\
&\geq \frac{1-\epsilon}{3}\rho(G) + \frac{1-\epsilon}{3}\rho(G) = \frac{2}{3}(1-\epsilon)\rho(G).
\end{aligned}$$

Since $M_1 \cup M_2$ is a path cover, we have $\rho(G) \geq |M_1 \cup M_2|$. Hence, the approximation factor of Algorithm 1 is at least $\frac{2}{3}(1-\epsilon)$. ◀

Now, we show that our analysis of the approximation factor of Algorithm 1 is tight. Consider the graph in Figure 2a and denote it by $\tilde{G}$. If we run Algorithm 1 on $\tilde{G}$, then the edges of M_1 could be the red edges shown in Figure 2b. After contracting $\tilde{G}$ on M_1, we have $\tilde{G}/M_1$ shown in Figure 2c. Finally, the second matching M_2 found by Algorithm 1 in $\tilde{G}/M_1$ contains at most one edge which implies $|M_1 \cup M_2| \leq 4$. On the other hand, maximum path cover P^* in $\tilde{G}$ contains 6 edges shown in Figure 2d.


(a) Graph $\tilde{G}$.


(b) Matching M_1 (red edges) found by Algorithm 1.


(c) $\tilde{G}/M_1$.

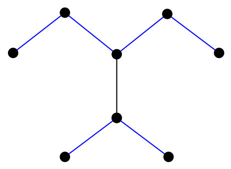

(d) P^* (blue edges) in $\tilde{G}$.

Figure 2 An example of a graph $\tilde{G}$ for which Algorithm 1 produces a path cover whose size is $\frac{2}{3}$ times the size of the MPC.

As a result, $\dfrac{|M_1 \cup M_2|}{|P^*|} \leq \frac{2}{3}$, so this example and Theorem 6 imply that the approximation factor of Algorithm 1 is $\frac{2}{3} - \epsilon$.

3.2 Implementation of Algorithm 1 in the Semi-Streaming Model

Now we explain how to implement Algorithm 1 in the semi-streaming model. We start with the following theorem by Fischer, Mitrovic and Uitto [10].

▶ **Theorem 7** (Theorem 1.1 in [10]). *Given a graph on n vertices, there is a deterministic $(1-\epsilon)$-approximation algorithm for maximum matching that runs in poly$(\frac{1}{\epsilon})$ passes in the semi-streaming model. Furthermore, the algorithm requires $n \cdot$ poly$(\frac{1}{\epsilon})$ words of memory.*

To implement Algorithm 1 in the semi-streaming model, we proceed as follows: In the first phase, by applying Theorem 7, we compute a $(1-\epsilon)$-approximate matching for the graph G, denoted as M_1. At the end of this phase, we have the edges of this matching. In the second phase, we again apply Theorem 7 to compute a matching for G/M_1 in the streaming model.

During the second phase, when we apply the algorithm of Theorem 7, while processing each edge (v_i, v_j) in the stream, we follow these rules: If $(v_i, v_j) \in M_1$, we ignore this edge. If $(v_i, v_j) \notin M_1$, but one of v_i or v_j is an endpoint of an edge in M_1 (e.g., $v_i, v_k \in M_1$), then since (v_i, v_j) is contracted, we consider v_i and v_j as a single vertex, v_{ij}. In this case, v_k is considered adjacent to the new vertex v_{ij}. Consequently, we can compute a $(1-\epsilon)$-approximation matching for G/M_1 in the next poly$(1/\epsilon)$ passes.

Thus, combining the results of Algorithm 1, Theorem 6, and Theorem 7, we have the main result of this section:

▶ **Theorem 8.** *Given an unweighted graph G on n vertices, there is a deterministic algorithm that returns a $(\frac{2}{3}-\epsilon)$-approximate MPC in the semi-streaming model in poly$(\frac{1}{\epsilon})$ passes.*

4 $(1,2)$-TSP

In this section, we present our algorithm for the $(1,2)$-TSP, detailed in Algorithm 2, and analyze its approximation factor. We also provide an explanation of how to implement this algorithm in the semi-streaming model.

Algorithm 2 Our algorithm for $(1,2)$-TSP.

```
1: Let $G_1$ be the subgraph of $G$ consisting of edges with weight 1.
2: Run Algorithm 1 on $G_1$ to get a path cover $\tilde{P}$.
3: Arbitrarily extend $\tilde{P}$ to a Hamiltonian cycle $\tilde{C}$ by adding edges between end points of $\tilde{P}$
   or/and existing vertices not in $\tilde{P}$.
4: return $\tilde{C}$.
```

▶ **Theorem 9.** *The approximation factor of Algorithm 2 for $(1,2)$-TSP is $\frac{4}{3}+\epsilon+\frac{1}{n}$.*

Proof. Let T^* be the optimal solution of (1,2)-TSP, ρ^* be the size of an MPC in G_1, and n be the number of vertices of G. Since every Hamiltonian cycle contains n edges with weights 1 or 2, we have $n \leq T^* \leq 2n$. We also have $T^* = 2n - \rho^* - 1$ or $T^* = 2n - \rho^*$, where $T^* = 2n - \rho^* - 1$ occurs only when G contains a Hamiltonian cycle consisting solely of edges with weight 1.

Let $\tilde{\rho}$ be the size of the path cover obtained by Algorithm 1 in G_1. The Hamiltonian cycle obtained by Algorithm 2 has a cost of at most $2(n-\tilde{\rho}) + \tilde{\rho} = 2n - \tilde{\rho}$. Let $\alpha \leq 1$ be the approximation factor of Algorithm 1, then we have $\alpha\rho^* \leq \tilde{\rho} \leq \rho^*$. As a result, $T^* \leq 2n - \rho^* \leq 2n - \tilde{\rho}$. We also have:

$$\begin{aligned} 2n - \tilde{\rho} &\leq 2n - \alpha\rho^* = (2-2\alpha)n + \alpha(2n - \rho^* - 1) + \alpha \\ &\leq (2-2\alpha)T^* + \alpha T^* + \alpha = (2-\alpha)T^* + \alpha \\ &\leq (2-\alpha)T^* + 1 \\ &\leq (2-\alpha)T^* + \frac{T^*}{n} = \left(2 - \alpha + \frac{1}{n}\right)T^*. \end{aligned} \tag{2}$$

By Theorem 6, we have $\alpha \geq \frac{2}{3} - \epsilon$. Using Equation (2), we conclude that:

$$\begin{aligned} 2n - \tilde{\rho} &\leq \left(2 - \alpha + \frac{1}{n}\right)T^* \\ &\leq \left(2 - \left(\frac{2}{3} - \epsilon\right) + \frac{1}{n}\right)T^* = \left(\frac{4}{3} + \epsilon + \frac{1}{n}\right)T^*. \end{aligned}$$

Hence, $T^* \leq 2n - \tilde{\rho} \leq \left(\frac{4}{3}+\epsilon+\frac{1}{n}\right)T^*$. So, the approximation factor of our algorithm for (1,2)-TSP is $\frac{4}{3}+\epsilon+\frac{1}{n}$. ◀

4.1 Implementation of Algorithm 2 in the Semi-Streaming Model

For a given instance of $(1,2)$-TSP in the streaming model, we compute an approximate MPC for the induced subgraph on the edges of weight 1 as explained in Theorem 8, then we add extra edges to connect these paths and vertices not in these paths arbitrarily to construct a Hamiltonian cycle, which gives us a $(4/3+\epsilon+1/n)$-approximate tour for $(1,2)$-TSP. So, we have the main result of this section as follows.

▶ **Theorem 10.** *Given an instance of $(1,2)$-TSP on n vertices, there is a deterministic algorithm that returns a $(\frac{4}{3}+\epsilon+\frac{1}{n})$-approximate $(1,2)$-TSP in the semi-streaming model in $O(poly(\frac{1}{\epsilon}))$ passes.*

5 Max-TSP

In this section, we introduce our algorithm for Max-TSP, which closely resembles our approach for MPC. The key difference is that, instead of using MM_ϵ, we employ a subroutine to compute an approximate maximum weight matching in a weighted graph.

Let MWM_ϵ be a subroutine for computing a $(1-\epsilon)$-approximate maximum weighted matching in a weighted graph G. First, we compute a matching M_1 for G using MWM_ϵ. Then, we contract the edges of M_1 to obtain another graph $G' = G/M_1$ and compute another matching, M_2, for G' using MWM_ϵ again. We derive the union of the two weighted matchings, $M_1 \cup M_2$. Similar to Lemma 1, it is evident that $M_1 \cup M_2$ forms a union of vertex-disjoint paths in G. Finally, since the graph is complete, there can be only one vertex that is not in $M_1 \cup M_2$. In this case we connect this vertex to one of the paths in $M_1 \cup M_2$. Now, we add edges arbitrarily between the endpoints of the paths in $M_1 \cup M_2$ to obtain a Hamiltonian cycle C for G.

Algorithm 3 Our algorithm for Max-TSP on a complete weighted graph G.

```
1: Run $\mathrm{MWM}_\epsilon$ on $G$ to find a matching $M_1$.
2: Contract $G$ on $M_1$ to get a new graph $G' = G/M_1$.
3: Run $\mathrm{MWM}_\epsilon$ on $G'$ to find another matching $M_2$.
4: Arbitrarily extend $M_1 \cup M_2$ to a Hamiltonian cycle $C$ by adding edges between end
   points of $M_1 \cup M_2$ or/and existing vertices not in $M_1 \cup M_2$.
5: return $C$.
```

Note that after contracting G on M_1 to obtain $G' = G/M_1$, this new graph might have parallel edges between to vertices. Since we aim to find a maximum matching in G', we can simply consider the edge with the largest weight for parallel edges and ignore the rest.

5.1 Analysis of the Approximation Factor of Algorithm 3

To analyze the approximation factor of Algorithm 3, we begin with a series of lemmas.

▶ **Lemma 11.** *Suppose C is a cycle of length k in a weighted graph G. Then, there exists a matching $M \subseteq C$ such that $w(M) \geq \frac{k-1}{2k} w(C)$.*

Proof. Assume that $e \in C$ is the edge with the minimum weight. Hence, $w(e) \leq w(C)/k$. Since $C - e$ is a path, there is a matching $M \subseteq C - e$ (which is also a subset of C) whose weight is at least $w(C-e)/2$. Finally,

$$w(M) \geq \frac{1}{2} w(C-e) = \frac{1}{2}(w(C) - w(e)) \geq \frac{1}{2}\left(w(C) - \frac{w(C)}{k}\right) = \frac{k-1}{2k} w(C). \qquad ◀$$

▶ **Lemma 12.** *Suppose T is a path or a cycle in a weighted graph G. Then there exists a matching $M \subseteq T$ such that $w(M) \geq \frac{1}{3} w(T)$.*

Proof. We have two cases

- T is a path. Enumerate the edges of T from one end point to the other. The odd numbered edges form a matching called M_{odd}. The same applies for even numbered edges, which form a matching called M_{even}. Since $T = M_{\mathrm{odd}} \cup M_{\mathrm{even}}$, at least one of these two matchings has weight no less than $w(T)/2$.
- T is a cycle. If it is a cycle of length 2 (i.e. T consists of two parallel edges), then obviously we can pick the edge e with bigger weight that satisfies $w(e) \geq \frac{1}{2} w(T) \geq \frac{1}{3} w(T)$. If the length of T is at least 3, then using Lemma 11, we conclude that there is a matching $M \subseteq T$ such that $w(M) \geq \frac{k-1}{2k} w(T) \geq \frac{1}{3} w(T)$. ◀

Now, we provide a lemma similar to Lemma 5 which works for the weighted version.

▶ **Lemma 13.** *Suppose M is a matching in a weighted graph G and C^* is a maximum weight Hamiltonian cycle of G. Then,*

$$\mu(G/M) \geq \frac{w(C^*) - w(M)}{6} - \frac{1}{3n}w(C^*).$$

Proof. We contract C^* on M in two steps. First, we contract C^* on the edges in $C^* \cap M$. Next, we contract the resulting graph on $M' = M \backslash C^*$. After the first step, $C' = C^*/(C^* \cap M)$ is a cycle with weight $w(C^*) - w(C^* \cap M)$ (see Figure 3).

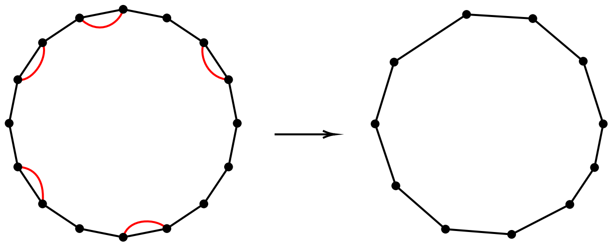


Figure 3 C^* remains a path cover after contraction on $C^* \cap M$ (red edges).

Here M' is a matching that connects some vertices of C' together (see Figure 4a, Figure 4b and Figure 4c).

Assume that the length of C' is k. Using Lemma 11, there is a matching $M^* \subseteq C'$ whose weight is at least $\frac{k-1}{2k}w(C')$ (see Figure 4d). Since the matching M_1 contains at most half of the edges of C^*, we conclude that $k \geq n/2$. As a result,

$$\begin{aligned} w(M^*) &\geq \frac{k-1}{2k}w(C') \geq \frac{n-2}{2n}(w(C^*) - w(C^* \cap M)) \\ &\geq \frac{w(C^*) - w(C^* \cap M)}{2} - \frac{1}{n}w(C^*). \end{aligned} \tag{3}$$

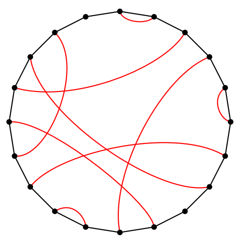
Since $M' \cap C' = \emptyset$, we conclude that $M' \cap M^* = \emptyset$. Because M^* and M' are matchings, it follows that $M^* \cup M'$ is a union of disjoint paths and cycles (see Figure 4e). As a result, after doing the second step of contraction, M^*/M' would also be a disjoint union of paths and cycles (whose number of edges is equal to $|M^*|$ since $M^* \cap M' = \emptyset$) in C'/M'. For instance, if $M^* \cup M'$ contains a cycle of length 4, then M^*/M' would contain a cycle of length 2 which contains parallel edges.

Now, consider each connected component of M^*/M'. This component is either a path or a cycle. Hence, by Lemma 12, We obtain a matching with a weight of at least one-third of the weight of the component.

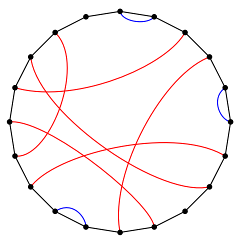
Finally, since these components are vertex-disjoint, the union of obtained matching would be a matching whose weight is at least $w(M^*)/3$. Note that this matching is also a matching in G/M. Hence, using Equation (3), we have

$$\begin{aligned} \mu(G/M) &\geq \frac{w(M^*)}{3} \geq \frac{w(C^*) - w(C^* \cap M)}{6} - \frac{1}{3n}w(C^*) \\ &\geq \frac{w(C^*) - w(M)}{6} - \frac{1}{3n}w(C^*). \end{aligned}$$ ◀

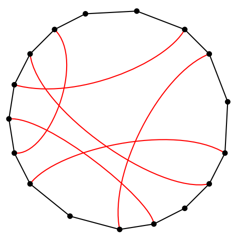
So, we have the following theorem which is a lower bound for the approximation factor of Algorithm 3.



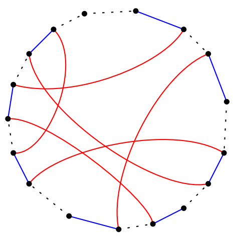
(a) An example of C^* and M (red edges) illustrating the proof of Lemma 13.



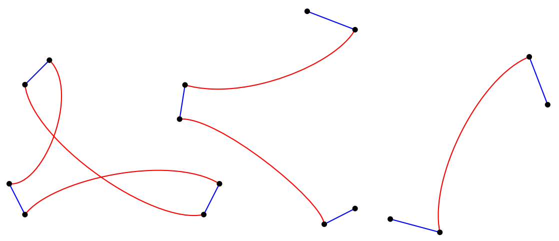
(b) Distinguishing $C^* \cap M$ (blue edges) and $M' = M \setminus C^*$ (red edges).



(c) C' is the contraction of C^* on $C^* \cap M$ and M' (red edges) connects some of its vertices together.



(d) the weight of M^* (blue edges) is at least $\frac{k-1}{2k} w(C')$.



(e) $M^* \cup M'$ is a disjoint union of paths and cycles (M^* and M' correspond to blue and red edges respectively).

Figure 4 An example of C^* and M illustrating the steps in the proof of Lemma 13.

▶ **Theorem 14.** *The approximation factor of Algorithm 3 is at least* $\left(\frac{7}{12} - \frac{3}{4n}\right)(1-\epsilon)$.

Proof. Let C^* be a maximum weight Hamiltonian cycle in G. By Lemma 11, there exists at least one matching $M \subseteq C^*$ whose weight is at least

$$\frac{n-1}{2n} w(C^*).$$

Since M_1 is a $(1-\epsilon)$-approximation of the maximum weighted matching in G we have

$$w(M_1) \geq \frac{(1-\epsilon)(n-1)}{2n} w(C^*).$$

By using Lemma 13 for M_2 on G/M_1, we have

$$\begin{aligned} w(M_1 \cup M_2) &= w(M_1) + w(M_2) \\ &\geq w(M_1) + (1-\epsilon)\mu(G/M_1) \\ &\geq w(M_1) + \frac{1-\epsilon}{6}(w(C^*) - w(M_1)) - \frac{1-\epsilon}{3n} w(C^*) \\ &\geq (1-\epsilon)\left(\frac{1}{6} - \frac{1}{3n}\right) w(C^*) + \frac{5}{6} w(M_1) \\ &\geq (1-\epsilon)\left(\frac{1}{6} - \frac{1}{3n}\right) w(C^*) + \frac{5(1-\epsilon)(n-1)}{12n} w(C^*) \\ &= \left(\frac{7}{12} - \frac{3}{4n}\right)(1-\epsilon) w(C^*). \end{aligned}$$

Since the weight of the edges of G are nonnegative, we have

$$w(C) \geq w(M_1 \cup M_2) \geq \left(\frac{7}{12} - \frac{3}{4n}\right)(1-\epsilon) w(C^*).$$

Finally, C is a Hamiltonian cycle which means $w(C^*) \geq w(C)$. Hence, the approximation factor of Algorithm 1 is at least $\left(\frac{7}{12} - \frac{3}{4n}\right)(1-\epsilon)$. ◀

5.2 Implementation of Algorithm 3 in the semi-streaming model

The implementation of Algorithm 3 in the semi-streaming model follows a similar approach as described in the previous section. Therefore, we omit a detailed explanation here. However, note that in this part, we should use a subroutine for computing a $(1-\epsilon)$-approximate maximum weight matching in the semi-streaming model. First, we recall the following theorem from [13]. We use the algorithm of this theorem as MWM_ϵ in our semi-streaming implementation of Algorithm 3.

▶ **Theorem 15** (Theorem 1.3 in [13]). *There exists a deterministic algorithm that returns a $(1-\epsilon)$-approximate maximum weight matching using $poly(\frac{1}{\epsilon})$ passes in the semi-streaming model. The algorithm requires $O(n \cdot \log W \cdot poly(\frac{1}{\epsilon}))$ words of memory where W is the maximum edge weight in the graph.*

Thus, Algorithm 3, Theorem 14, and Theorem 15 present the following theorem for Max-TSP in the semi-streaming model.

▶ **Theorem 16.** *Given an instance of Max-TSP on n vertices, there is an algorithm that returns a $(\frac{7}{12} - \frac{3}{4n})(1-\epsilon)$-approximate Max-TSP the semi-streaming model in $O(poly(\frac{1}{\epsilon}))$ passes. The algorithm requires $O(n \cdot \log W \cdot poly(\frac{1}{\epsilon}))$ words of memory where W is the maximum edge weight in the graph.*

6 Future Work

As a future work we propose the following algorithm that can help to improve the approximation factor for MPC in the semi-streaming model. The algorithm improves Algorithm 1 by iteratively finding new matchings and contracting the graph over these matchings. It is crucial to ensure that this process preserves the path cover property. Hence, during the kth iteration of our loop, we must remove all the edges in G that are incident to a middle point of any path (connected component) within $\cup_{i=1}^{k} M_i$. This is because $\left(\cup_{i=1}^{k} M_i\right) \cup M_{k+1}$ must remain a path cover, which means M_{k+1} cannot include any edge incident to a middle point of a path in $\cup_{i=1}^{k} M_i$. See Algorithm 4.

Algorithm 4 Extension of Algorithm 1.

```
1: Run $\mathrm{MM}_\epsilon$ (or $\mathrm{MWM}_\epsilon$ for weighted version) on $G$ to find a matching $M_1$.
2: Let $i = 1$.
3: while $M_i \neq \emptyset$:
4:     Let $G^{(i)} = G$.
5:     Remove all $e \in E(G^{(i)}) \setminus (\cup_{k=0}^{i} M_k)$ from $E(G^{(i)})$ that are incident to at least one
       middle point of a path (connected component) in $\cup_{k=0}^{i} M_k$.
6:     Contract $G^{(i)}$ on $\cup_{k=0}^{i} M_k$.
7:     Run $\mathrm{MM}_\epsilon$ (or $\mathrm{MWM}_\epsilon$ for weighted version) on $G^{(i)}$ to find a matching $M_{i+1}$.
8:     $i = i + 1$
9: return $\cup_{k=1}^{i} M_k$.
```

We leave the computation of the approximation factor of Algorithm 4 as a challenging open problem. Currently, we know that the approximation factor is at most $3/4$. Consider the graph in Figure 5a: the algorithm may select the red edges as M_1. After contraction, it might select the red edge in Figure 5b as M_2. In the next iteration, the graph becomes empty, as we must remove any edge incident to a middle point of $M_1 \cup M_2$. Thus, the algorithm terminates with a path of length 3. However, the MPC has 4 edges (see Figure 5c).

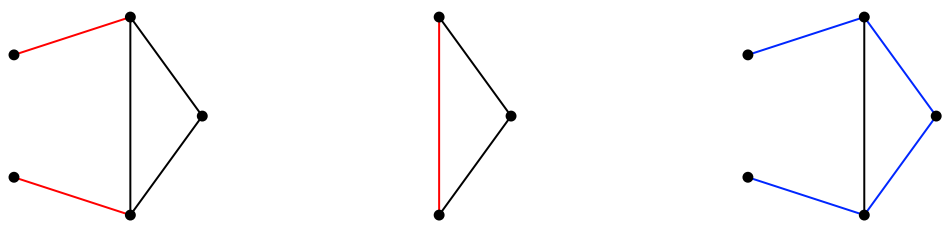

(a) M_1 (red edges). **(b)** M_2 (red edges). **(c)** MPC (blue edges).

Figure 5 An example of a graph where Algorithm 4 terminates after two iterations. The algorithm produces a $\frac{3}{4}$-approximation of the Maximum Path Cover (MPC).

The main bottleneck to find the approximation ratio of Algorithm 4 is that after the second iteration, there might be a lot of edges that we have to remove from the contracted graph in order to make sure that the union of matchings remains a path cover. More precisely, while running line 5 of Algorithm 4, a bunch of edges that are contained in every maximum path cover might be removed. We are not aware of any argument how to bound the number of these edges. This prevents us to provide an argument like Lemma 5 and Lemma 13. Finding the exact approximation ratio of Algorithm 4 seems to require clever new ideas, already for three matchings.

References

1 Anna Adamaszek, Matthias Mnich, and Katarzyna Paluch. New approximation algorithms for (1, 2)-tsp. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, volume 107 of *LIPIcs*, pages 9:1–9:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ICALP.2018.9.

2 Sharareh Alipour, Ermiya Farokhnejad, and Tobias Mömke. Improved approximation algorithms for (1,2)-tsp and max-tsp using path covers in the semi-streaming model, 2025. arXiv:2501.04813.

3 Soheil Behnezhad, Mohammad Roghani, Aviad Rubinstein, and Amin Saberi. Sublinear algorithms for TSP via path covers. *CoRR*, abs/2301.05350, 2023. doi:10.48550/arXiv.2301.05350.

4 Yu Chen, Sampath Kannan, and Sanjeev Khanna. Sublinear algorithms and lower bounds for metric TSP cost estimation. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPIcs*, pages 30:1–30:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ICALP.2020.30.

5 Yu Chen, Sanjeev Khanna, and Zihan Tan. Sublinear algorithms and lower bounds for estimating MST and TSP cost in general metrics. In Kousha Etessami, Uriel Feige, and Gabriele Puppis, editors, *50th International Colloquium on Automata, Languages, and Programming, ICALP 2023, July 10-14, 2023, Paderborn, Germany*, volume 261 of *LIPIcs*, pages 37:1–37:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ICALP.2023.37.

6 Artur Czumaj and Christian Sohler. Estimating the weight of metric minimum spanning trees in sublinear-time. In László Babai, editor, *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 175–183. ACM, 2004. doi:10.1145/1007352.1007386.

7 Doratha E. Drake and Stefan Hougardy. Improved linear time approximation algorithms for weighted matchings. In Sanjeev Arora, Klaus Jansen, José D. P. Rolim, and Amit Sahai, editors, *Approximation, Randomization, and Combinatorial Optimization: Algorithms and Techniques, 6th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2003 and 7th International Workshop on Randomization and Approximation Techniques in Computer Science, RANDOM 2003, Princeton, NJ, USA, August 24-26, 2003, Proceedings*, volume 2764 of *Lecture Notes in Computer Science*, pages 14–23. Springer, 2003. doi:10.1007/978-3-540-45198-3_2.

8 Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. Graph distances in the streaming model: the value of space. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2005, Vancouver, British Columbia, Canada, January 23-25, 2005*, pages 745–754. SIAM, 2005. URL: http://dl.acm.org/citation.cfm?id=1070432.1070537.

9 Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. *Theor. Comput. Sci.*, 348(2-3):207–216, 2005. doi:10.1016/J.TCS.2005.09.013.

10 Manuela Fischer, Slobodan Mitrovic, and Jara Uitto. Deterministic $(1+\epsilon)$-approximate maximum matching with poly$(1/\epsilon)$ passes in the semi-streaming model and beyond. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 248–260. ACM, 2022. doi:10.1145/3519935.3520039.

11 Buddhima Gamlath, Sagar Kale, Slobodan Mitrovic, and Ola Svensson. Weighted matchings via unweighted augmentations. In Peter Robinson and Faith Ellen, editors, *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019*, pages 491–500. ACM, 2019. doi:10.1145/3293611.3331603.

12 Shayan Oveis Gharan, Amin Saberi, and Mohit Singh. A randomized rounding approach to the traveling salesman problem. In Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 550–559. IEEE Computer Society, 2011. doi:10.1109/FOCS.2011.80.

13 Shang-En Huang and Hsin-Hao Su. (1-1013)-approximate maximum weighted matching in poly(1/1013, log n) time in the distributed and parallel settings. In Rotem Oshman, Alexandre Nolin, Magnús M. Halldórsson, and Alkida Balliu, editors, *Proceedings of the 2023 ACM Symposium on Principles of Distributed Computing, PODC 2023, Orlando, FL, USA, June 19-23, 2023*, pages 44–54. ACM, 2023. doi:10.1145/3583668.3594570.

14 Anna R. Karlin, Nathan Klein, and Shayan Oveis Gharan. A (slightly) improved approximation algorithm for metric TSP. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 32–45. ACM, 2021. doi:10.1145/3406325.3451009.

15 Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972. doi:10.1007/978-1-4684-2001-2_9.

16 Andrew McGregor. Finding graph matchings in data streams. In Chandra Chekuri, Klaus Jansen, José D. P. Rolim, and Luca Trevisan, editors, *Approximation, Randomization and Combinatorial Optimization, Algorithms and Techniques, 8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2005 and 9th InternationalWorkshop on Randomization and Computation, RANDOM 2005, Berkeley, CA, USA, August 22-24, 2005, Proceedings*, volume 3624 of *Lecture Notes in Computer Science*, pages 170–181. Springer, 2005. doi:10.1007/11538462_15.

17 Matthias Mnich and Tobias Mömke. Improved integrality gap upper bounds for traveling salesperson problems with distances one and two. *Eur. J. Oper. Res.*, 266(2):436–457, 2018. doi:10.1016/j.ejor.2017.09.036.

18 Tobias Mömke and Ola Svensson. Approximating graphic TSP by matchings. *CoRR*, abs/1104.3090, 2011. arXiv:1104.3090.

19 Marcin Mucha. 13/9-approximation for graphic TSP. In Christoph Dürr and Thomas Wilke, editors, *29th International Symposium on Theoretical Aspects of Computer Science, STACS 2012, February 29th - March 3rd, 2012, Paris, France*, volume 14 of *LIPIcs*, pages 30–41. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2012. doi:10.4230/LIPIcs.STACS.2012.30.

20 Christos H. Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. *J. Comput. Syst. Sci.*, 43(3):425–440, 1991. doi:10.1016/0022-0000(91)90023-X.

21 Ami Paz and Gregory Schwartzman. A (2+ ε)-approximation for maximum weight matching in the semi-streaming model. *ACM Transactions on Algorithms (TALG)*, 15(2):1–15, 2018. doi:10.1145/3274668.

22 Sartaj Sahni and Teofilo F. Gonzalez. P-complete approximation problems. *J. ACM*, 23(3):555–565, 1976. doi:10.1145/321958.321975.

23 András Sebö and Jens Vygen. Shorter tours by nicer ears: 7/5-approximation for the graph-tsp, 3/2 for the path version, and 4/3 for two-edge-connected subgraphs. *Comb.*, 34(5):597–629, 2014. doi:10.1007/s00493-014-2960-3.

24 Xianghui Zhong. On the approximation ratio of the k-opt and lin-kernighan algorithm for metric and graph TSP. In Fabrizio Grandoni, Grzegorz Herman, and Peter Sanders, editors, *28th Annual European Symposium on Algorithms, ESA 2020, September 7-9, 2020, Pisa, Italy (Virtual Conference)*, volume 173 of *LIPIcs*, pages 83:1–83:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ESA.2020.83.

25 Xianghui Zhong. On the approximation ratio of the 3-opt algorithm for the (1, 2)-tsp. *CoRR*, abs/2103.00504, 2021. arXiv:2103.00504.