# Constructing Large Matchings via Query Access to a Maximal Matching Oracle

**Lidiya Khalidah binti Khalil**
Department of Computer Science, University of Bristol, UK
lb17727@bristol.ac.uk

**Christian Konrad**[1]
Department of Computer Science, University of Bristol, UK
http://people.cs.bris.ac.uk/~konrad/
christian.konrad@bristol.ac.uk

## Abstract

Multi-pass streaming algorithm for Maximum Matching have been studied since more than 15 years and various algorithmic results are known today, including 2-pass streaming algorithms that break the 1/2-approximation barrier, and $(1-\epsilon)$-approximation streaming algorithms that run in $O(\text{poly}\,\frac{1}{\epsilon})$ passes in bipartite graphs and in $O((\frac{1}{\epsilon})^{\frac{1}{\epsilon}})$ or $O(\text{poly}(\frac{1}{\epsilon}) \cdot \log n)$ passes in general graphs, where $n$ is the number of vertices of the input graph. However, proving impossibility results for such algorithms has so far been elusive, and, for example, even the existence of 2-pass small space streaming algorithms with approximation factor 0.999 has not yet been ruled out.

The key building block of all multi-pass streaming algorithms for Maximum Matching is the Greedy matching algorithm. Our aim is to understand the limitations of this approach: How many passes are required if the algorithm solely relies on the invocation of the Greedy algorithm?

In this paper, we initiate the study of lower bounds for restricted families of multi-pass streaming algorithms for Maximum Matching. We focus on the simple yet powerful class of algorithms that in each pass run Greedy on a vertex-induced subgraph of the input graph. In bipartite graphs, we show that 3 passes are necessary and sufficient to improve on the trivial approximation factor of 1/2: We give a lower bound of 0.6 on the approximation ratio of such algorithms, which is optimal. We further show that $\Omega(\frac{1}{\epsilon})$ passes are required for computing a $(1-\epsilon)$-approximation, even in bipartite graphs. Last, the considered class of algorithms is not well-suited to general graphs: We show that $\Omega(n)$ passes are required in order to improve on the trivial approximation factor of 1/2.

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

**Keywords and phrases** Maximum matching approximation, Query model, Streaming algorithms

**Digital Object Identifier** 10.4230/LIPIcs.FSTTCS.2020.26

## 1 Introduction

The Greedy matching algorithm is the key building block of most published streaming algorithms for approximate Maximum Matching [16, 26, 13, 25, 2, 14, 21, 24]. Given a graph $G = (V, E)$, Greedy scans the set of edges $E$ in arbitrary order and inserts the current edge $e \in E$ into an initially empty matching $M$ if possible, i.e., if both endpoints of $e$ are not yet matched by an edge in $M$. Greedy produces a maximal matching, which is known to be at least half as large as a matching of largest size.

The Greedy matching algorithm is well-suited for implementation in the *streaming model of computation*. A streaming algorithm processing a graph $G = (V, E)$ with $|V| = n$ receives a potentially adversarially ordered sequence of the edges of the input graph, and the objective

[1] Corresponding author

© Lidiya Khalidah binti Khalil and Christian Konrad;
licensed under Creative Commons License CC-BY
40th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2020).
Editors: Nitin Saxena and Sunil Simon; Article No. 26; pp. 26:1–26:15
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

is to solve a graph problem using as little space as possible. Many graph problems require space $\Omega(n \log n)$ to be solved in the streaming model [28], and streaming algorithms that use space $\mathrm{O}(n \operatorname{poly} \log n)$ are referred to as *semi-streaming algorithms*. Multi-pass streaming algorithms process the input stream multiple times. Observe that Greedy constitutes a one-pass semi-streaming algorithm for Maximum Matching with approximation factor $\frac{1}{2}$.

The Maximum Matching problem is the most studied graph problem in the streaming model, and despite intense research efforts, the Greedy algorithm is the best one-pass streaming algorithm known today, even if space $\mathrm{O}(n^{2-\delta})$ is allowed, for any $\delta > 0$. Performing multiple passes over the input allows improving the approximation factor. The main questions addressed in the literature are: (1) What can be achieved in $p$ passes, for small $p$ (e.g. $p \in \{2, 3\}$), and (2) How many passes are required in order to obtain a $(1-\epsilon)$-approximation, for any $\epsilon > 0$. See Table 1 for an overview of the currently best results.

**Table 1** State of the art semi-streaming algorithms for Maximum Matching.

| # passes | Approximation | det/rand | Reference | See also |
|---|---|---|---|---|
| **Bipartite Graphs** | | | | |
| 1 | $\frac{1}{2}$ | deterministic | Greedy, folklore | |
| 2 | $2-\sqrt{2} \approx 0.5857$ | randomized | Konrad [24] | [25, 14, 21] |
| 3 | 0.6067 | randomized | Konrad [24] | [14, 21] |
| $\frac{2}{3\epsilon}$ | $\frac{2}{3}-\epsilon$ | deterministic | Kale and Tirodkar [21] | [16] |
| $\mathrm{O}(\frac{1}{\epsilon^2}\log\log\epsilon)$ | $1-\epsilon$ | deterministic | Ahn and Guha [2] | [13] |
| **General Graphs** | | | | |
| 1 | $\frac{1}{2}$ | deterministic | Greedy, folklore | |
| 2 | 0.53125 | deterministic | Kale and Tirodkar [21] | [25] |
| $\frac{1}{\epsilon}^{\mathrm{O}(\frac{1}{\epsilon})}$ | $1-\epsilon$ | deterministic | Tirodkar [29] | [26] |
| $\mathrm{O}(\frac{1}{\epsilon^4}\log n)$ | $1-\epsilon$ | deterministic | Ahn and Guha [2] | |

Only few lower bounds are known: We know that one-pass semi-streaming algorithms cannot have an approximation factor larger than $1-\frac{1}{e}$ [22] (see also [18]). The only multi-pass lower bound known addresses the exact version of Maximum Matching, showing that computing a maximum matching in $p$ passes requires space $n^{1+\Omega(1/p)}/p^{O(1)}$ [20]. No lower bound is known for multiple passes and approximations, and, for example, the existence of a 2-pass 0.999-approximation semi-streaming algorithm has not yet been ruled out.

The Greedy algorithm is the key building block of all algorithms referenced in Table 1 (including those mentioned in the "See also" column). In many cases, the presented algorithms collect edges by *solely* executing Greedy on specific subgraphs in each pass and output a large matching computed from the edges produced by Greedy. In this paper, we are interested in the limitations of this approach: How large a matching can be computed if Greedy is executed at most $p$ times?

Known streaming algorithms apply Greedy in different ways. For example, the 2-pass and 3-pass algorithms by Konrad [24] run Greedy on randomly sampled subgraphs that depend on a previously computed maximal matching. The multi-pass algorithms by Ahn and Guha [2] maintain vertex weights $\in [0, 1]$ over the course of the algorithm and run Greedy on a *threshold subgraph*, i.e., on the set of edges $uv$ so that the sum of the current weights associated with $u$ and $v$ is at most 1. The algorithm by Eggert et al. [13] runs Greedy on an edge-induced subgraph in order to find augmenting paths.

In this paper, we initiate the study of lower bounds for restricted families of multi-pass streaming algorithms for Maximum Matching that are based on Greedy. We start this line of research by addressing the probably simplest and most natural approach, which is

nevertheless surprisingly powerful: the class of deterministic algorithms that run Greedy on a vertex-induced subgraph in each pass. Two known streaming algorithms fit our model:

1. A 3-pass 0.6-approximation streaming algorithm for bipartite graphs that is implicit in [16], explicitly mentioned in [25], and analyzed in [21]. Given a bipartite graph $G = (A, B, E)$, the algorithm first computes a maximal matching in $G$, i.e., $M \leftarrow \textsc{Greedy}(G)$. Then, the algorithm attempts to find length-3 augmenting paths by invoking Greedy twice more: $M_L \leftarrow \textsc{Greedy}(G[A(M) \cup \overline{B(M)}])$, where $A(M)$ are the matched $A$-vertices and $\overline{B(M)}$ are the unmatched $B$-vertices. Last, $M_R \leftarrow \textsc{Greedy}(\overline{A(M)}, B')$, where $B' \subseteq B(M)$ are those matched $B$ vertices that are endpoints in length-2 paths in $M_L \cup M$. Kale and Tirodkar showed that $M \cup M_L \cup M_R$ contains a 0.6-approximate matching [21]. We will denote this algorithm by 3RoundMatch.
2. The $(1-\epsilon)$-approximation $\mathrm{O}(\frac{1}{\epsilon^5})$-passes streaming algorithm for bipartite graphs by Eggert et al. [13] can be adapted to fit our model using $\mathrm{O}(\frac{1}{\epsilon^6})$ invocations of Greedy.

We abstract this approach as a game between a player and an oracle: Let $G$ be a graph with vertex set $V$. The player initially knows $V$. In each round $i$ the player sends a query $\mathsf{query}(V_i)$ to the oracle, where $V_i \subseteq V$. The oracle returns a maximal matching in the vertex-induced subgraph $G[V_i]$. For this model to yield lower bounds for the streaming model, we impose that the oracle is *streaming-consistent*, i.e., there exists a stream of edges $\pi$ so that the oracle's answers to the queries $(\mathsf{query}(V_i))_i$ equal runs of Greedy on the respective substream of edges $G[V_i]$ of $\pi$ (see preliminaries for a more detailed definition). We denote this model as the *vertex-query model* (as opposed to an edge-query model, where the player may ask for maximal matchings in a subgraph spanned by a subset of edges).

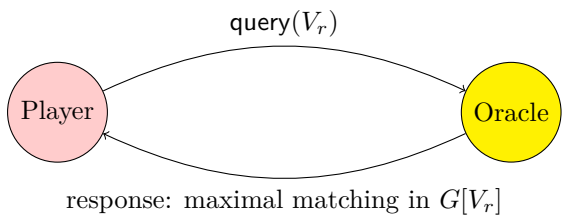



**Figure 1** Illustration of the game between the player and oracle in the vertex-query model.

**Our Results.** In bipartite graphs, we show that at least 3 rounds are required to improve on the approximation factor of 1/2, and we give a lower bound of 0.6 on the approximation factor of 3 round algorithms. This is optimal, as demonstrated by the previously mentioned algorithm 3RoundMatch. We also show that $\Omega(\frac{1}{\epsilon})$ rounds are required for computing a $(1-\epsilon)$-approximation. This polynomial lower bound is in line with the poly $\frac{1}{\epsilon}$ rounds upper bound by Eggert et al. [13]. Last, we demonstrate that our query model is not well-suited to general graphs: We show that improving on a factor of 1/2 requires $\Omega(n)$ rounds.

**Further Related Work.** Besides the adversarial one-pass and multi-pass streaming models, Maximum Matching has also been studied in the random order [25, 24, 17, 4, 15, 8] and the insertion-deletion settings [23, 9, 6, 12]. In the random order model, where edges arrive in uniform random order, Konrad et al. [25] were the first to give a semi-streaming algorithms with approximation ratio above 1/2. Very recently, Bernstein showed that an approximation ratio of 2/3 can be achieved in random order streams [8]. In light of the lower bound

by Kapralov [22], this result separates the adversarial and the random order settings. In insertion-deletion streams, edges that have previously been inserted may be deleted again. Assadi et al. [6] showed that, up to sub-polynomial factors, space $n^{2-3\epsilon}$ is necessary and sufficient for computing a $n^{\epsilon}$-approximation (see [12] for a slightly improved lower bound).

Many works allow only query access to the input graph. For example, cross-additive queries, bipartite independent set queries, additive queries, cut-queries, and edge-detection queries have been considered [19, 3, 11, 10, 7, 27, 1], however, mainly for graph reconstruction problems. Very recently, linear queries and or-queries have been considered for graph connectivity [5].

**Outline.** In Section 2, we give notation and definitions. We also define the vertex-query model and provide a construction mechanism that ensures that our oracles are streaming-consistent. Then, in Section 3 we prove that 3 rounds are required to improve on 1/2 and give a lower bound of 0.6 on the approximation ratio achievable in three rounds. In Section 4, we show that $\Omega(\frac{1}{\epsilon})$ rounds are needed for computing a $(1-\epsilon)$-approximation, and in Section 5 we show that improving on $\frac{1}{2}$ in general graphs requires $\Omega(n)$ rounds. Finally, we conclude in Section 6 and give open questions.

## 2 Preliminaries

**Matchings.** Let $G = (V, E)$ be a graph with $|V| = n$. A *matching* $M \subseteq E$ is a subset of vertex-disjoint edges. Matching $M$ is *maximal* if for every $e \in E \setminus M : M \cup \{e\}$ is not a matching. A *maximum matching* is one of largest cardinality. If the size of a matching $M$ is $n/2$, i.e., it matches all vertices of the graph, then $M$ is a *perfect matching*.

**Notation.** We write $V(M)$ to denote the set of vertices incident to the edges of a matching $M$. For a subset of vertices $V' \subseteq V$, we denote by $G[V']$ the *vertex-induced subgraph* of $G$ by vertices $V'$, i.e., $G[V'] = (V', (V' \times V') \cap E)$. For a set of edges $E' \subseteq E$, we denote by $OPT(E')$ the size of a maximum matching in the subgraph of $G$ spanned by the edges $E'$. For an integer $n$, we define $[n] := \{1, 2, \ldots, n\}$.

**The Vertex-query Model.** In the *vertex-query model*, a player and an oracle play a rounds-based matching game on a vertex set $V$ of size $n$ that is initially known to both parties. Over the course of the game, the oracle makes up a graph $G = (V, E)$. The objective of the player is to learn a large matching in $G$. The way the player learns edges is as follows:

In each round $1 \leq i \leq r$, where $r$ is the total number of rounds played, the player submits a query $\mathsf{query}(V_i)$ to the oracle, for some $V_i \subseteq V$. The oracle then determines a set of edges $M_i$, which is guaranteed to be a maximal matching in the vertex-induced subgraph $G[V_i]$. Observe that in doing so, the oracle not only commits to the fact that $M_i \subseteq E$, but also that the vertices $V_i \setminus V(M_i)$ form an independent set (which follows from the fact that $M_i$ is maximal). Furthermore, we impose that the answers to all queries are consistent with graph $G$ and that $G$ has a perfect matching.

After the $r$ query rounds, the player reports a largest matching $M_P$ that can be formed using the edges $\cup_{i \leq r} M_i$. The *approximation ratio* of the solution obtained is $|M_P|/(\frac{1}{2}n)$.

We are interested in oracles that are consistent with the streaming model. We say that an oracle is *streaming-consistent*, if there exists an ordering $\pi$ of the edges $E$ so that, for every round $i$, $M_i$ is produced by running Greedy on the substream of $\pi$ consisting of the edges of $G[V_i]$. We will ensure that all our oracles are streaming-consistent.

**Construction of Streaming-consistent Oracles.** We will construct streaming-consistent oracles as follows. Upon query $V_1$, the oracle answers with $M_1$ and places $M_1$ in the beginning of the stream $\pi$. Next, given query $V_i$, for some $i \geq 2$, the oracle first runs GREEDY on the substream of $\pi$ consisting of the edges $G[V_i]$ which produces an intermediate matching $M'$, thereby attempting to match $V_i$ using edges of previous matchings $\cup_{j<i} M_j$. The oracle then extends $M'$ to a matching $M_i$. Edges $M_i \setminus M'$ are then introduced at the end of the stream $\pi$. This construction procedure guarantees that our oracles are streaming-consistent. Furthermore, it allows us to simplify our arguments, since it is enough to restrict our considerations to queries with the following property:

▶ **Observation 1.** *Suppose that the oracle is constructed as above. Then, given the sequence of queries $V_1, \ldots, V_r$ and matchings $M_1, \ldots, M_r$, there exists a sequence of queries $\tilde{V}_1, \ldots, \tilde{V}_r$ that produces matchings $\tilde{M}_1, \ldots, \tilde{M}_r$ such that:*

- *The player learns the same set of edges, i.e., for every $i \leq r : \bigcup_{j \leq i} M_j = \bigcup_{j \leq i} \tilde{M}_j$, and*
- *No query $\tilde{V}_i$ contains a pair of vertices $u, v$ such that $uv \in \cup_{j<i} \tilde{M}_j$.*

We can therefore assume that the player never includes a pair of vertices $u, v$ into a query so that the edge $uv$ is contained in a previous answer from the oracle.

## 3 Lower Bound for Few Round Algorithms in Bipartite Graphs

In this section, we show that the player cannot produce an approximation ratio better than $\frac{1}{2}$ in two rounds, even on bipartite graphs. We also show that three rounds do not allow for an approximation ratio better than 0.6, which is achieved by the algorithm 3ROUNDMATCHING.

In order to keep track of the information learned by the player, we will make use of *structure graphs*, which we discuss first.

### 3.1 Structure Graphs

Observe that when the oracle answers the query $\mathsf{query}(V_i)$ and returns a maximal matching $M_i$, the player not only learns that the edges $M_i$ are contained in the input graph $G$, but also learns that the vertices $V_i \setminus V(M_i)$ form an independent set in $G$ (due to the maximality of $M_i$). We maintain the structure learned by the player and the structure committed to by the oracle (which do not have to be identical) using *structure graphs*:

▶ **Definition 2** (Structure graph). *A 4-tuple $(A, B, E, F)$ is a **bipartite structure graph** if:*

- *$A, B$ are disjoint sets of vertices,*
- *$E, F$ are disjoint sets of edges such that $(A, B, E)$ and $(A, B, F)$ are bipartite graphs,*
- *The structure graph admits a perfect matching, i.e., there exists a set of edges $M^*$ such that $M^* \cap F = \emptyset$ and $M^*$ is a perfect matching in the bipartite graph $(A, B, E \cup M^*)$ .*

From the perspective of the player, the set $E$ corresponds to the edges returned by the oracle so far, i.e., $E = \cup_{j \leq i} M_j$, and the set $F$ corresponds to guaranteed *non-edges*, i.e., $F = \cup_{j \leq i} C(V_i \setminus V(M_i))$, where $C(V')$ denotes a biclique (respecting the bipartition $A, B$) among the vertices $V'$.

In the following, we will denote the structure graph after round $i$ learned by the player by $\tilde{H}_i = (A, B, \tilde{E}_i, \tilde{F}_i)$, i.e., $\tilde{E}_i = \cup_{j \leq i} M_j$ and $\tilde{F}_i = \cup_{j \leq i} C(V_i \setminus V(M_i))$. The oracle will also maintain a sequence of structure graphs $(H_i)_i$ with $H_i = (A, B, E_i, F_i)$ such that $H_i$ *dominates* $\tilde{H}_i$, for every $1 \leq i \leq r$. We say that a structure graph $H = (A, B, E, F)$ dominates a structure graph $\tilde{H} = (A, B, \tilde{E}, \tilde{F})$, if $\tilde{E} \subseteq E$ and $\tilde{F} \subseteq F$. This notion allows the oracle to commit to edges and non-edges that the player has not yet learned. This domination property allows us to simplify our arguments.

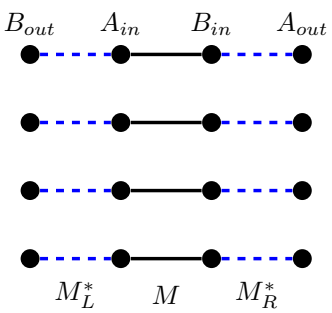



**Figure 2** Illustration of the structure graph $H_1$ on a graph on 16 vertices. The matching $M$ is half the size of the matching $M^* = M_L^* \cup M_R^*$.

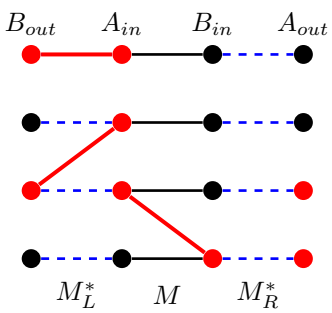



**Figure 3** Matching $M_2$ (in red) returned by the oracle. The red vertices constitute $A_2 \cup B_2$, i.e., the vertices of the second query. The case $|A_2^{in}| \geq |B_2^{in}|$ is illustrated here. We see that no edges from $B_{in} \times A_{out}$ are returned, and that $M_2$ does not allow us to increase the size of $M$.

In our lower bound arguments, we make use of the following two assumptions:

▶ **Assumption 1.** *After round $i$, the player knows the structure graph $H_i$.*

This is a valid assumption since $H_i$ dominates $\tilde{H}_i$ and thus contains at least as much information as $\tilde{H}_i$. This assumption therefore only strengthens the player. Furthermore, we will also assume a slightly strengthened property of the property discussed in Observation 1:

▶ **Assumption 2.** *For every $1 \leq i \leq r$, we assume that query $V_i$ does not contain a pair of vertices $u, v \in V_i$ such that $uv \in E_{i-1}$.*

This is a valid assumption, since if such a pair $u, v$ of vertices existed in $V_i$, the oracle could simply match $u$ to $v$ in $M_i$ and the algorithm would not learn any new information.

Last, observe that the approximation ratio of the player's strategy is completely determined by $H_r$, the oracle's structure graph after the last round. Since $H_r$ dominates $\tilde{H}_r$, the player's largest matching is of size at most $OPT(E_r)$. Since by definition of a structure graph, $H_r$ admits a perfect matching, the approximation ratio achieved is $2 \cdot OPT(E_r)/n$.

## 3.2 Lower Bound for Two Rounds

Assume that $n$ is a multiple of 4. The player and the oracle play the matching game on a bipartite vertex set $V = A \dot{\cup} B$ with $|A| = |B| = n/2$. Consider the structure graph:

$$H_1 = (A_{in} \cup A_{out}, B_{in} \cup B_{out}, M, A_{out} \times B_{out}) \ ,$$

where $|A_{in}| = |A_{out}| = |B_{in}| = |B_{out}| = n/4$, and $M$ is a perfect matching between $A_{in}$ and $B_{in}$. Observe that there exists an $M^*$ outside $A_{out} \times B_{out}$ such that $M^*$ is a perfect matching in $(A, B, M \cup M^*)$, namely, $M^*$ consists of the two perfect matchings $M_L^*$ connecting $B_{out}$ to $A_{in}$ and $M_R^*$ connecting $B_{in}$ to $A_{out}$. See Figure 2 for an illustration.

We have:

▶ **Lemma 3.** *There is a structure graph isomorphic to $H_1$ that dominates $\tilde{H}_1$.*

**Proof.** Denote the first query by $A_1, B_1$ ($A_1 \subseteq A$, and $B_1 \subseteq B$). We will argue that we can relabel the sets $A_{in}, A_{out}, B_{in}, B_{out}$ so that $H_1$ dominates $\tilde{H}_1$:

If $A_1 \leq n/4$ then let $A_{in}$ be an arbitrary subset of the $A$ vertices of size $n/4$ that contains $A_1$, and let $A_{out}$ be the remaining $A$-vertices. If $A_1 > n/4$ then let $A_{out}$ be an arbitrary subset of $A$ vertices of size $n/4$ that contains $A \setminus A_1$, and let $A_{in}$ be the remaining $A$-vertices. Proceed similarly for $B_1$. The oracle returns the subset $M_1 \subseteq M$ where each edge has one endpoint in $A_1$ and one endpoint in $B_1$, which is clearly maximal given that edges in $A_{out} \times B_{out}$ are forbidden. ◀

Since $OPT(M) = |M| = \frac{1}{4}n$, Lemma 3 implies the unsurprising fact that no one round algorithm has an approximation ratio better than $\frac{2 \cdot \frac{1}{4}n}{n} = \frac{1}{2}$. We argue now that an additional round does not help with increasing the approximation factor.

▶ **Theorem 4.** *The best approximation ratio achievable in two rounds is* $1/2$.

**Proof.** Let $A_2, B_2$ be the vertices of the second query. By Lemma 3, $H_1$ dominates $\tilde{H}_1$, and by Assumption 1 we can assume that the player already knows $H_1$. Let $A_2^{in} = A_2 \cap A_{in}$, $A_2^{out} = A_2 \cap A_{out}$ and define $B_2^{in}$ and $B_2^{out}$ similarly.

Suppose first that $|A_2^{in}| \geq |B_2^{in}|$. Then the oracle returns a matching $M_2$ that matches an arbitrary subset of $A_2^{in}$ of size $|B_2^{in}|$ to $B_2^{in}$, and matches $\max\{|B_2^{out}|, |A_2^{in}| - |B_2^{in}|\}$ of the remaining $A_2^{in}$ vertices arbitrarily to vertices in $B_2^{out}$. In doing so, either all $A_2^{in}$ vertices or all $B_2$ vertices are matched. Since $H_1$ indicates that there are no edges connecting the "out"-vertices, $M_2$ is therefore maximal.

Observe further that $M \cup M_2$ does not match any vertex in $A_{out}$, and, hence, only half of the $A$-vertices are matched in $M \cup M_2$. The player thus cannot report any matching of size larger than $|M|$, which constitutes a 1/2-approximation.

Last, the case $|A_2^{in}| < |B_2^{in}|$ is identical with roles of $A$ and $B$ vertices reversed. ◀

## 3.3 Lower Bound for Three Rounds

In this section, we work with a vertex set $V = A \dot{\cup} B$ with $|A| = |B| = 5$ (and thus $|V| = n = 10$). By choosing disjoint copies of this vertex set, our result can be extended to graphs with an arbitrarily large number of vertices.

**First Query.** Similar to the two round case, we define the structure graph $H_1 = (A_{in} \cup A_{out}, B_{in} \cup B_{out}, M, A_{out} \times B_{out})$, however, this time $|A_{in}| = |B_{in}| = 3$ and $|A_{out}| = |B_{out}| = 2$. The matching $M$ matches $A_{in}$ to $B_{in}$, see Figure 4a.

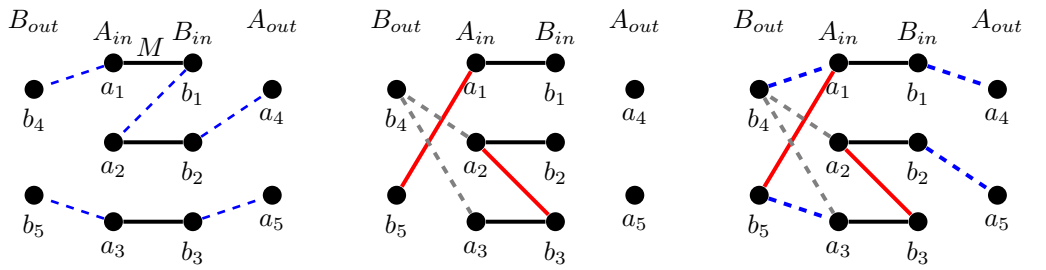
It shall be convenient to assign labels to the vertices in our structure graph. In our arguments below, in order to avoid symmetric cases, we relabel the vertices of our structure graph as we see fit, however, we always ensure that the structure graph after relabeling is isomorphic to the structure graph before the relabeling.

First, similar to Lemma 3, it is not hard to see that a structure graph isomorphic to $H_1$ dominates $\tilde{H}_1$ (proof omitted).

▶ **Lemma 5.** *There is a structure graph isomorphic to* $H_1$ *that dominates* $\tilde{H}_1$.

**Second Query.** We assume that the player knows $H_1$ after the first query (Assumption 1). Next, we define structure graph $H_2 = (A_{in} \cup A_{out}, B_{in} \cup B_{out}, M \cup E_2, A_{out} \times B_{out} \cup F_2)$, where $E_2 = \{a_1b_5, a_2b_3\}$, and $F_2 = \{a_2b_4, a_3b_4\}$. It is easy to see that $H_2$ is indeed a structure graph (see Figures 4b and 4c).

We shall prove that there is a structure graph isomorphic to $H_2$ that dominates $\tilde{H}_2$. Lemma 6 considers the case when the second query $V_2$ contains exactly three "in"-vertices, i.e., vertices from $A_{in} \cup B_{in}$, and Lemma 7 considers the case when there are fewer "in"-vertices.




**(a)** $H_1$: The blue edges constitute a perfect matching that does not use any edges connecting $A_{out}$ to $B_{out}$.

**(b)** $H_2$: black edges are in $M$, red edges in $E_2$, gray edges in $F_2$.

**(c)** $H_2$: The blue dotted edges and the edge $a_2b_3$ constitute a maximum matching.

**Figure 4** Illustrations of structure graphs $H_1$ and $H_2$.

By Assumption 2, we do not need to consider the cases when more than three "in"-vertices are contained in $V_2$ since then $V_2$ necessarily contains a pair of vertices $u, v$ such that $uv \in M$.

▶ **Lemma 6.** *If the player queries exactly 3 "in"-vertices (i.e., vertices from $A_{in} \cup B_{in}$) in their second query then there exists a structure graph isomorphic to $H_2$ that dominates $\tilde{H}_2$.*

**Proof.** The player can either query more vertices in $A_{in}$ or in $B_{in}$, and these cases are symmetrical. Hence we only consider the case when the player queries more vertices in $A_{in}$. Due to Assumption 2, for queries that contain vertices in both $A_{in}$ and $B_{in}$, we assume these vertices do not form any edges seen in $M$.

Since we will not match any vertices in $A_{out}$, we do not need to distinguish between cases where the player queries different numbers of vertices in $A_{out}$. We distinguish between the following cases:

1. Player queries all vertices in $A_{in}$ and the query includes $b_5$: the oracle returns $M_2 = \{a_1b_5\}$.
2. Player queries all vertices in $A_{in}$ and only $b_4$ in $B_{out}$: relabel $b_4$ as $b_5$ and proceed as in case (1).
3. Player queries all vertices in $A_{in}$ and no vertices in $B_{out}$: the oracle returns $M_2 = \emptyset$.
4. Player queries two vertices in $A_{in}$, one vertex in $B_{in}$ and the query includes $b_5$: relabel the "in" vertices so that after relabeling the vertices $a_1$, $a_2$ and $b_3$ are included in the query. The oracle returns $M_2 = E_2$.
5. Player queries two vertices in $A_{in}$, one vertex in $B_{in}$ and only $b_4$ in $B_{out}$: relabel $b_4$ as $b_5$ and proceed as in case (4).
6. Player queries two vertices in $A_{in}$, one vertex in $B_{in}$ and no vertices in $B_{out}$: relabel "in" vertices so that after relabeling the vertices $a_2$ and $b_3$ are included in the query. The oracle returns $M_2 = \{a_2b_3\}$.

In all cases considered, observe that $M_2 \subseteq E_2$. Further, edges $F_2$ ensure that $M_2$ is maximal. ◀

We argue now that querying three "in"-vertices in the second round is best possible in the sense that querying fewer (or more) "in"-vertices does not yield more information.

▶ **Lemma 7.** *If the player queries fewer than 3 "in"-vertices (i.e., vertices from $A_{in} \cup B_{in}$) then there exists a structure graph isomorphic to $H_2$ that dominates $\tilde{H}_2$.*

**Proof.** Clearly if the player does not query any "in"-vertices, no matching will be found i.e. $M_2 = \emptyset$. If the player queries exactly one vertex in $A_{in}$, we can relabel this vertex as $a_1$ and if the query contains a vertex in $B_{out}$, relabel this one to be $b_5$. Then the matching

found will be a subset of $E_2$. If the player queries exactly two "in" vertices there are two cases to consider. If they are both in $A_{in}$, we ensure one of these vertices is $a_1$ by relabeling, and, if at least one vertex in $B_{out}$ is queried, potentially relabel this vertex to be $b_5$ and return the edge $a_1b_5$. If the player queried one vertex in $A_{in}$ and one in $B_{in}$, we relabel these vertices as $a_2, b_3$ and return the edge between them, $a_2b_3$. Hence the edges learned by the player are always a subset of $E_2$. In all cases considered, edges $F_2$ ensure that matching $M_2$ is maximal. ◀

**Third Query.** We assume that the player knows structure graph $H_2$. Similar to the second query, we distinguish between the cases where the player queries exactly three "in"-vertices and fewer "in"-vertices. Again, by Assumption 2, we do not need to consider the case where the player queries more than three "in"-vertices. In the following proofs, we will define different structure graphs $H_3$ that depend on the individual query.

▶ **Lemma 8.** *If the player queries exactly 3 "in"-vertices in the third round, then the player cannot output a matching of size larger than* 3.

**Proof.** We provide the oracle's answers when the player queries exactly three "in"-vertices. Among those cases, there are three cases to consider where the player queries more vertices in $B_{in}$ than in $A_{in}$:

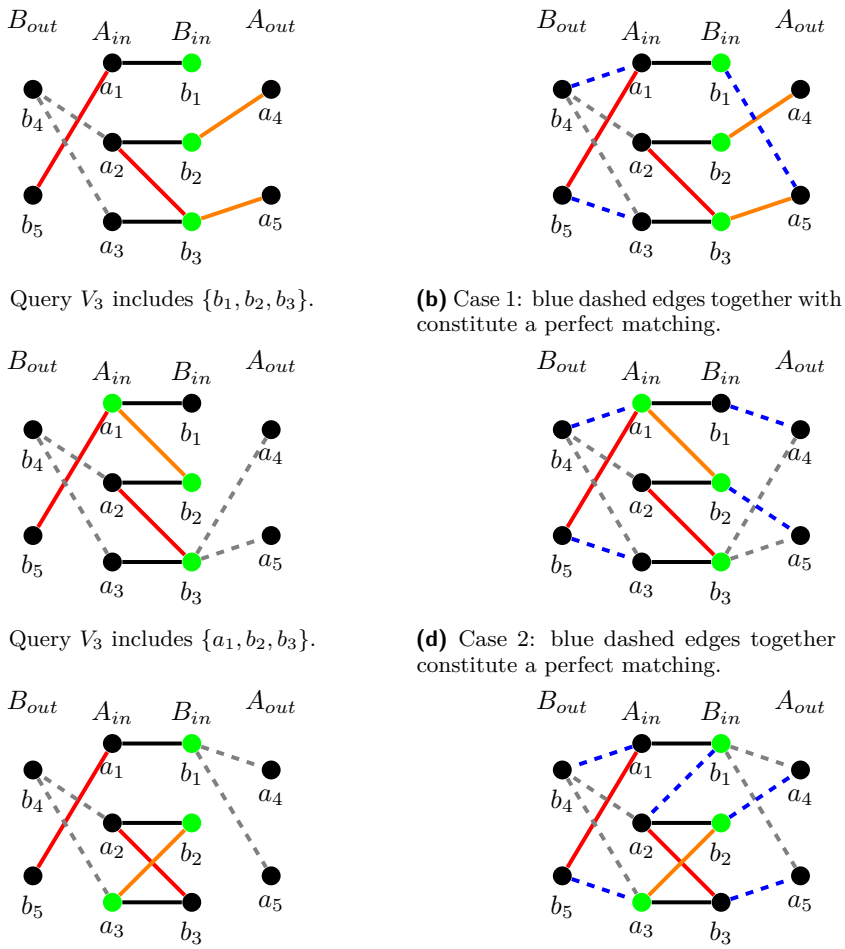
1. **Case 1:** Player queries $b_1, b_2, b_3$. The oracle defines $H_3 = (A, B, E_3, F_3)$ such that $E_3 = M \cup E_2 \cup \{a_4b_2, a_5b_3\}$ and $F_3 = A_{out} \times B_{out} \cup F_2$. If the player queried both vertices in $A_{out}$, the oracle returns $M_3 = \{a_4b_2, a_5b_3\}$. Otherwise $M_3$ would consist of one or zero edges depending on the player's query. In particular, we have $M_3 \subset E_3$.
   In cases 2 and 3, we do not define any edges involving vertices from $A_{out}$ or $B_{out}$, so the oracle proceeds regardless of which vertices in $A_{out}, B_{out}$ the player queried.
2. **Case 2:** Player queries $a_1, b_2, b_3$. The oracle defines $H_3 = (A, B, E_3, F_3)$ such that $E_3 = M \cup E_2 \cup \{a_1b_2\}$ and $F_3 = A_{out} \times B_{out} \cup F_2 \cup \{a_4b_3, a_5b_3\}$. The oracle returns $M_3 = \{a_1b_2\}$.
3. **Case 3:** Player queries $b_1, b_2, a_3$. The oracle defines $H_3 = (A, B, E_3, F_3)$ such that $E_3 = M \cup E_2 \cup \{a_3b_2\}$ and $F_3 = A_{out} \times B_{out} \cup F_2 \cup \{a_4b_1, a_5b_1\}$. The oracle returns $M_3 = \{a_3b_2\}$.

Observe that the case $b_1, a_2, b_3 \in V_3$ is not relevant, since $a_2b_3 \in M_2$ and Assumption 2. Figure 5 shows that in these three cases, $H_3$ is a structure graph and the largest matching that the player thus able to return is of size 3.

If the player queries more vertices in $A_{in}$ than in $B_{in}$, we will argue that the player will not learn any edges connecting to vertices in $A_{out}$, and since the player then only holds edges incident to 3 of the 5 $A$-vertices, the player cannot report a matching larger than of size 3.

If the player queries all three vertices in $A_{in}$ then he clearly cannot learn any edges connecting to $A_{out}$. If the player queries a vertex in $B_{in}$, note that we can match it with a vertex queried in $A_{in}$, and there will be no vertices left to match with vertices in $A_{out}$ (see Figure 6). Since no more non-edges are defined, it is easy to see that edges can be added to create a perfect matching. ◀

▶ **Lemma 9.** *If the player queries fewer than three "in"-vertices in the third round, then the player cannot output a matching of size larger than* 3.




**(a)** Case 1: Query $V_3$ includes $\{b_1, b_2, b_3\}$.

**(b)** Case 1: blue dashed edges together with $a_4b_2, a_2b_3$ constitute a perfect matching.

**(c)** Case 2: Query $V_3$ includes $\{a_1, b_2, b_3\}$.

**(d)** Case 2: blue dashed edges together with $a_2b_3$ constitute a perfect matching.

**(e)** Case 3: Query $V_3$ includes $\{b_1, b_2, a_3\}$.

**(f)** Case 3: blue dashed edges form a perfect matching.

**Figure 5** Round 3 cases. Green vertices are queried by the player in round 3. Red edges are in $E_2 \setminus E_1$, orange is $E_3 \setminus E_2$, grey is $F_3$. The blue dashed edges can be added to the graph to create a perfect matching.

**Proof.** We distinguish the following cases:

1. If the player queries no "in" vertices, this is obvious, and we would have $M_3 = \emptyset$.
2. If the player queries exactly one "in" vertex, the only possible way to obtain a larger matching than one of size 3 is to find an edge incident to $b_1$, i.e., by querying $b_1$, but we can define $F_3 = A_{out} \times B_{out} \cup F_2 \cup \{a_4b_1, a_5b_1\}$ and then $M_3 = \emptyset$.
3. If the player queries one vertex in $A_{in}$ and one in $B_{in}$, we can connect them by an edge, say $e$, and then $M_3 = \{e\}$ does not help increasing the size of a matching.
4. If the player queries two vertices in $A_{in}$, the player will not be able to learn any edges to vertices in $A_{out}$, and so $A_{out}$ remains unmatched, which implies that the player cannot return a matching of size larger than 3.
5. If the player queries two vertices in $B_{in}$, the oracle defines $H_3$ as in Case 1 of Lemma 8, and the matching returned is a subset of $E_3$. ◀

Hence we have shown that no matter what queries are made in the second and third rounds, the player cannot increase the size of the matching learned within the 10-vertex subgraph. This then holds for a graph with $|A| = |B| = n$ where $5|n$ and the theorem follows.

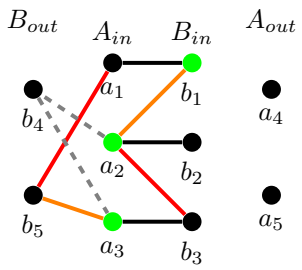



**Figure 6** An example of how the oracle behaves when the player queries more vertices in $A_{in}$ than in $B_{in}$ during the third round. Green vertices are queried by the player. Red edges are in $E_2 \setminus E_1$, orange is $E_3 \setminus E_2$, gray is $F_3$. The player learns no edges incident to $A_{out}$ and can therefore only report a matching of size 3.

▶ **Theorem 10.** *The best approximation factor achievable in three rounds is* $3/5$.

## 4 $(1-\epsilon)$-approximation in Bipartite Graphs Requires $\Omega(\frac{1}{\epsilon})$ Rounds

Let $G_c = (A, B, E)$ with $A = B = [c]$ be the *semi-complete* graph on $2c$ vertices, i.e., vertices $a \in A$ and $b \in B$ are connected if and only if $b \geq a$. Observe that $G_c$ has a unique perfect matching $M^* = \{(i, i) \in E \mid i \in [c]\}$.

Let $G$ be the disjoint union of $n/(2c)$ copies of $G_c$ (assuming for simplicity that $n$ is a multiple of $2c$). We will refer to a copy of $G_c$ in $G$ as a gadget. We now show that computing a $(1-\epsilon)$-approximation requires $\Omega(\frac{1}{\epsilon})$ queries on $G$.

▶ **Theorem 11.** *Any query algorithm with approximation factor* $1-\epsilon$ *requires at least* $\frac{1}{\epsilon} - 1$ *queries, even in bipartite graphs.*

**Proof.** Let $c = \frac{1}{\epsilon} - 1$. We consider the graph $G$. First, suppose that the algorithm does not compute a perfect matching in any of the $n/(2c)$ gadgets. Then, the computed matching is of size at most $\frac{c-1}{c}\frac{n}{2}$ and thus constitutes at best a $\frac{c-1}{c} = 1 - \frac{\epsilon}{1-\epsilon} < 1 - \epsilon$ approximation. The algorithm therefore needs to compute a perfect matching in at least one gadget. Since all gadgets are disjoint, we now argue that it requires at least $c$ queries in order to compute a perfect matching in one gadget. Consider thus the gadget $G_c$ and denote by $M^*$ the perfect matching in $G_c$. We claim that each query may produce at most one edge of the perfect matching $M^*$ in $G_c$:

Indeed, let $A' = \{a_1, a_2, \ldots, a_k\} \subseteq A$ and $B' = \{b_1, b_2, \ldots, b_\ell\} \subseteq B$ be so that $A' \cup B'$ is any query submitted to the oracle. Further, suppose that $a_1 < a_2 < \cdots < a_k$ and $b_1 < b_2 < \cdots < b_\ell$. The oracle will return the following matching $M$:

$$M = \{a_i b_{\ell+1-i} \mid i \in [\min\{k, \ell\}]\} \cap E \ .$$

We will now argue that $M$ is maximal and $|M \cap M^*| \leq 1$. To this end, let $j$ be the largest index such that $a_j b_{\ell+1-j} \in E$, which is equivalent to $j$ being the largest index so that $a_j \leq b_{\ell+1-j}$. Observe that since the $(a_i)_i$ and $(b_i)_i$ are increasing, we have $a_{j'} b_{\ell+1-j'} \in E \Leftrightarrow j' \leq j$, which also implies that vertices $a_{j'}$ are matched, for every $j' \leq j$. Consider now a vertex $a_q$, for some $q > j$. Since $a_{j+1} > b_{\ell-j}$ and $a_q \geq a_{j+1}$, it follows that there is no edge between $a_q$ and any of the unmatched $B'$-vertices $\{b_1, b_2, \ldots, b_{\ell-j}\}$. This implies that the matching $M$ is maximal. Next, suppose that $M$ contains at least one edge from $M^*$ and let $q$ be the smallest index such that $a_q = b_{\ell+1-q}$, i.e., $(a_q, b_{\ell+1-q}) \in M^*$. Then, for any $q' > q$, we have

$$a_{q'} > a_q = b_{\ell+1-q} > b_{\ell+1-q'} \ ,$$

which implies that $a_{q'} \neq b_{\ell+1-q'}$. Hence, at most one edge from $M^*$ is returned per query.

Last, we argue that the oracle can be made streaming-consistent: Consider any ordering of the edges so that edge $ij$ arrives before edge $ik$, for every $k < j$. ◀

Using the oracle described in the previous proof on a single gadget $G_{n/2}$, we obtain the following corollary:

▶ **Corollary 12.** *Any query algorithm that produces a maximum matching requires at least $n/2$ queries (on a graph on $n$ vertices), even on bipartite graphs.*

## 5 Improving on $1/2$ in General Graphs Requires $\Omega(n)$ Queries

Let $G$ be a *bomb graph* on $n$ ($n$ even) vertices $U \cup V$ with $|U| = |V| = [n/2]$, where $G[V]$ is a clique, $G[U]$ is an independent set, and $u \in U$ and $v \in V$ are connected if and only if $u = v$ ($U$ and $V$ are connected via a perfect matching). Denote by $M^*$ the perfect matching between $U$ and $V$ and by $C$ the edges of the clique $G[V]$.

In the next lemma, we show that any large matching in $G$ must contain a large number of edges from $M^*$.

▶ **Lemma 13.** *Let $M$ be a matching in $G$. Then: $|M| \leq \frac{n}{4} + \frac{1}{2}|M \cap M^*|$ .*

**Proof.** Observe that $|M| = |M \cap M^*| + |M \cap C|$, and since there are $n/2 - |M \cap M^*|$ vertices in $V$ that are not matched to a vertex in $U$, we have $|M \cap C| \leq (n/2 - |M \cap M^*|)/2$. Hence:

$$|M| = |M \cap M^*| + |M \cap C| \leq |M \cap M^*| + (n/2 - |M \cap M^*|)/2 = \frac{n}{4} + \frac{1}{2}|M \cap M^*| \ . \quad ◀$$

▶ **Theorem 14.** *Any $r$-round query algorithm on general graphs has approximation ratio at most $\frac{1}{2} + \frac{r}{n}$ (on an $n$-vertex input graph).*

**Proof.** Consider an arbitrary query $U' \cup V'$ so that $U' \subseteq U$ and $V' \subseteq V$. The oracle returns the following matching: First, the oracle arbitrarily pairs up all vertices of $V'$ except possibly one in case $|V'|$ is odd. Let $M$ denote this matching. If $|V'|$ is even then $M$ is returned. Suppose now that $|V'|$ is odd and let $v \in V'$ be the vertex that is not matched in $M$. Then, if $v$'s partner $u \in U$ in $M^*$ is contained in $U'$, then return $M \cup \{uv\}$, otherwise return $M$.

It is easy to see that, by construction, the returned matching is maximal and contains at most one edge from $M^*$. Hence, in $r$-rounds the algorithm can learn at most $r$ edges from $M^*$. By Lemma 13, the returned matching is therefore of size at most $\frac{n}{4} + \frac{1}{2}r$, which constitutes a $\frac{1}{2} + \frac{r}{n}$-approximation.

The oracle can be made streaming-consistent: Consider any edge order where we first have edges $C$ in arbitrary order followed by $M^*$ in arbitrary order. ◀

## 6 Conclusion

In this paper, we introduced a new query model that allows us to prove lower bounds for streaming algorithms for Maximum Matching that repeatedly run the Greedy matching algorithm on a vertex-induced subgraph of the input graph. We showed that the three rounds algorithm 3RoundMatch with approximation factor 0.6 is optimal for this class of algorithms. We also showed that computing a $(1 - \epsilon)$-approximation in bipartite graphs requires $\Omega(\frac{1}{\epsilon})$ rounds, and computing an approximation strictly better than $\frac{1}{2}$ in general graphs requires $\Omega(n)$ rounds. We conclude with open questions:

- Can we prove that computing a maximum matching in the vertex-query model in bipartite graphs requires $\Omega(n^2)$ rounds, or is there an algorithm that requires only $o(n^2)$ rounds?
- Can we prove a $\Omega(\frac{1}{\epsilon^2})$ lower bound for computing a $(1-\epsilon)$-approximation in bipartite graphs?

## References


1. Hasan Abasi and Nader H. Bshouty. On learning graphs with edge-detecting queries. In Aurélien Garivier and Satyen Kale, editors, *Algorithmic Learning Theory, ALT 2019, 22-24 March 2019, Chicago, Illinois, USA*, volume 98 of *Proceedings of Machine Learning Research*, pages 3–30. PMLR, 2019. URL: `http://proceedings.mlr.press/v98/abasi19a.html`.
2. Kook Jin Ahn and Sudipto Guha. Linear programming in the semi-streaming model with application to the maximum matching problem. *Information and Computation*, 222:59–79, 2013. 38th International Colloquium on Automata, Languages and Programming (ICALP 2011). `doi:10.1016/j.ic.2012.10.006`.
3. Noga Alon, Richard Beigel, Simon Kasif, Steven Rudich, and Benny Sudakov. Learning a hidden matching. *SIAM J. Comput.*, 33(2):487–501, 2004. `doi:10.1137/S0097539702420139`.
4. Sepehr Assadi, MohammadHossein Bateni, Aaron Bernstein, Vahab S. Mirrokni, and Cliff Stein. Coresets meet EDCS: algorithms for matching and vertex cover on massive graphs. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 1616–1635. SIAM, 2019. `doi:10.1137/1.9781611975482.98`.
5. Sepehr Assadi, Deeparnab Chakrabarty, and Sanjeev Khanna. Graph connectivity and single element recovery via linear and or queries, 2020. `arXiv:2007.06098`.
6. Sepehr Assadi, Sanjeev Khanna, Yang Li, and Grigory Yaroslavtsev. Maximum matchings in dynamic graph streams and the simultaneous communication model. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1345–1364. SIAM, 2016. `doi:10.1137/1.9781611974331.ch93`.
7. Paul Beame, Sariel Har-Peled, Sivaramakrishnan Natarajan Ramamoorthy, Cyrus Rashtchian, and Makrand Sinha. Edge Estimation with Independent Set Oracles. In Anna R. Karlin, editor, *9th Innovations in Theoretical Computer Science Conference (ITCS 2018)*, volume 94 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 38:1–38:21, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. `doi:10.4230/LIPIcs.ITCS.2018.38`.
8. Aaron Bernstein. Improved Bounds for Matching in Random-Order Streams. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*, volume 168 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 12:1–12:13, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.ICALP.2020.12`.
9. Rajesh Chitnis, Graham Cormode, Hossein Esfandiari, MohammadTaghi Hajiaghayi, Andrew McGregor, Morteza Monemizadeh, and Sofya Vorotnikova. Kernelization via sampling with applications to finding matchings and related problems in dynamic graph streams. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1326–1344. SIAM, 2016. `doi:10.1137/1.9781611974331.ch92`.
10. Sung-Soon Choi. Polynomial time optimal query algorithms for finding graphs with arbitrary real weights. In Shai Shalev-Shwartz and Ingo Steinwart, editors, *COLT 2013 - The 26th Annual Conference on Learning Theory, June 12-14, 2013, Princeton University, NJ, USA*, volume 30 of *JMLR Workshop and Conference Proceedings*, pages 797–818. JMLR.org, 2013. URL: `http://proceedings.mlr.press/v30/Choi13.html`.
11. Sung-Soon Choi and Jeong Han Kim. Optimal query complexity bounds for finding graphs. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC

'08, page 749–758, New York, NY, USA, 2008. Association for Computing Machinery. doi:10.1145/1374376.1374484.

12 Jacques Dark and Christian Konrad. Optimal lower bounds for matching and vertex cover in dynamic graph streams. In *35th Computational Complexity Conference, CCC 2020*, LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

13 Sebastian Eggert, Lasse Kliemann, Peter Munstermann, and Anand Srivastav. Bipartite matching in the semi-streaming model. *Algorithmica*, 63(1–2):490–508, June 2012.

14 Hossein Esfandiari, MohammadTaghi Hajiaghayi, and Morteza Monemizadeh. Finding large matchings in semi-streaming. In Carlotta Domeniconi, Francesco Gullo, Francesco Bonchi, Josep Domingo-Ferrer, Ricardo Baeza-Yates, Zhi-Hua Zhou, and Xindong Wu, editors, *IEEE International Conference on Data Mining Workshops, ICDM Workshops 2016, December 12-15, 2016, Barcelona, Spain*, pages 608–614. IEEE Computer Society, 2016. doi:10.1109/ICDMW.2016.0092.

15 Alireza Farhadi, MohammadTaghi Hajiaghayi, Tung Mai, Anup Rao, and Ryan A. Rossi. Approximate maximum matching in random streams. In *Proceedings of the Thirty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '20, page 1773–1785, USA, 2020. Society for Industrial and Applied Mathematics.

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

17 Buddhima Gamlath, Sagar Kale, Slobodan Mitrovic, and Ola Svensson. Weighted matchings via unweighted augmentations. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, PODC '19, page 491–500, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3293611.3331603.

18 Ashish Goel, Michael Kapralov, and Sanjeev Khanna. On the communication and streaming complexity of maximum bipartite matching. In Yuval Rabani, editor, *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 468–485. SIAM, 2012. doi:10.1137/1.9781611973099.41.

19 Vladimir Grebinski and Gregory Kucherov. Optimal reconstruction of graphs under the additive model. *Algorithmica*, 28(1):104–124, 2000. doi:10.1007/s004530010033.

20 Venkatesan Guruswami and Krzysztof Onak. Superlinear lower bounds for multipass graph processing. *Algorithmica*, 76(3):654–683, 2016. doi:10.1007/s00453-016-0138-7.

21 Sagar Kale and Sumedh Tirodkar. Maximum matching in two, three, and a few more passes over graph streams. In Klaus Jansen, José D. P. Rolim, David Williamson, and Santosh S. Vempala, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2017, August 16-18, 2017, Berkeley, CA, USA*, volume 81 of *LIPIcs*, pages 15:1–15:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.APPROX-RANDOM.2017.15.

22 Michael Kapralov. Better bounds for matchings in the streaming model. In Sanjeev Khanna, editor, *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 1679–1697. SIAM, 2013. doi:10.1137/1.9781611973105.121.

23 Christian Konrad. Maximum matching in turnstile streams. In Nikhil Bansal and Irene Finocchi, editors, *Algorithms - ESA 2015 - 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, volume 9294 of *Lecture Notes in Computer Science*, pages 840–852. Springer, 2015. doi:10.1007/978-3-662-48350-3_70.

24 Christian Konrad. A Simple Augmentation Method for Matchings with Applications to Streaming Algorithms. In Igor Potapov, Paul Spirakis, and James Worrell, editors, *43rd International Symposium on Mathematical Foundations of Computer Science (MFCS 2018)*, volume 117 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 74:1–74:16, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.MFCS.2018.74.

**25** Christian Konrad, Frédéric Magniez, and Claire Mathieu. Maximum matching in semi-streaming with few passes. In Anupam Gupta, Klaus Jansen, José D. P. Rolim, and Rocco A. Servedio, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 15th International Workshop, APPROX 2012, and 16th International Workshop, RANDOM 2012, Cambridge, MA, USA, August 15-17, 2012. Proceedings*, volume 7408 of *Lecture Notes in Computer Science*, pages 231–242. Springer, 2012. doi:10.1007/978-3-642-32512-0_20.

**26** Andrew McGregor. Finding graph matchings in data streams. In *Proceedings of the 8th International Workshop on Approximation, Randomization and Combinatorial Optimization Problems, and Proceedings of the 9th International Conference on Randamization and Computation: Algorithms and Techniques*, APPROX'05/RANDOM'05, page 170–181, Berlin, Heidelberg, 2005. Springer-Verlag. doi:10.1007/11538462_15.

**27** Aviad Rubinstein, Tselil Schramm, and S. Matthew Weinberg. Computing Exact Minimum Cuts Without Knowing the Graph. In Anna R. Karlin, editor, *9th Innovations in Theoretical Computer Science Conference (ITCS 2018)*, volume 94 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 39:1–39:16, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ITCS.2018.39.

**28** Xiaoming Sun and David P. Woodruff. Tight Bounds for Graph Problems in Insertion Streams. In Naveen Garg, Klaus Jansen, Anup Rao, and José D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2015)*, volume 40 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 435–448, Dagstuhl, Germany, 2015. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.APPROX-RANDOM.2015.435.

**29** Sumedh Tirodkar. Deterministic Algorithms for Maximum Matching on General Graphs in the Semi-Streaming Model. In Sumit Ganguly and Paritosh Pandya, editors, *38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2018)*, volume 122 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 39:1–39:16, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.FSTTCS.2018.39.