21st Century Algorithms
Project Report

Prantar Ghosh and Maryam Negahbani

November 7, 2018

Paper: Linear Programming in the Semi-streaming Model with Application to the Maximum Matching Problem. Kook Jin Ahn and Sudipto Guha [AG13]

1 Background

This paper studies the Maximum Matching Problem, which is a classical and one of the most widely studied problems in Theoretical Computer Science. A Matching is a set of edges in a graph that do not share end-points. In the Maximum Cardinality Matching (MCM) problem, given an unweighted graph, we need to find a matching of maximum size. A generalization of this problem is the Maximum Weighted Matching (MWM) problem, where the input is a weighted graph and we are to find a matching of maximum total weight. Both of these problems have also been studied specifically for the class of Bipartite Graphs, as various problems that we commonly encounter in Computer Science naturally reduce to finding Bipartite Matchings. This paper considers both the MCM and MWM problems in the *semi-streaming model*, and has qualitatively better results for the special case of Bipartite Graphs.

It is important to understand the model that this paper considers. The *streaming model* is a widely studied model for processing massive data sets. In this model, the input is defined by a stream of data that arrives sequentially. For instance, for graph problems, the stream could be the stream of edges of the graph. An algorithm in this model has to process this input stream in the order it arrives, using a limited amount of memory. The Streaming model in general also allows multiple passes over the data. It has been explored for various problems, that are apparently hard in limited space for a single pass, whether they can be solved with multiple (but small) number of passes over the data stream. For approximation algorithms, it has been shown in several works that there is often a natural trade-off between number of passes and the approximation factor.

Solving most graph problems in the streaming model is difficult if the space allowed is sublinear in the number of vertices as in that case, it is not even possible to store some individual information about all vertices simultaneously. On the other hand, there isn't enough space to store the entire graph, i.e. all edges of the graph. Hence, having space roughly equal to the size of the vertex set, which is essentially sublinear in the number of edges, is an ideal middle ground. The *semi-streaming model* allows $O(n \cdot \text{polylog}(n))$ space, where n is the number of vertices in the graph, and hence is the most extensively studied model for solving graph problems.

Massive graphs arise naturally in various domains where there is data about certain entities as well as the relationships between those entities, e.g. people and their friendships on a social network, web-pages and hyperlinks, information retrieval, neurons and synapses, papers and their citations, just to name a few. So with limited memory, we need to process such graphs without storing all the edges. Combinatorial optimization problems on such graphs are well-studied and are often challenging, even when we have unrestricted access to the entire graph. So it is interesting to bring these two challenges together and aim to design algorithms for combinatorial optimization problems in graphs by storing only limited number of edges. The maximum matching problem, with all its generalizations, is one of the most celebrated combinatorial optimization problems in Computer Science and so whether it can be solved without storing all the edges is a very interesting question to explore. So one can say that the maximum matching problem in the streaming model is both practically and theoretically intriguing.

The Maximum Matching problem has a rich literature. Lots of work have considered the problem in the offline setting and while some of them explore algorithms that solve it exactly, multiple others look for fast approximation algorithms that focus on getting a close approximate solution in small runtime. Feigenbaum et al. initiated the study of this problem in the semi-streaming model. Note that a simple greedy algorithm that constructs a maximal matching of the graph gives a $1/2$- approximation for MCM and that is the best approximation factor known for single pass algorithms for the problem. Konrad et al. considered the problem in a setting where the edges arrive in a random-order and designed an algorithm that achieves a $1/1.98$-approximation factor in expectation. Most notable works on MCM and MWM prior to this paper and their comparison with the results here are given in the following table:

Problem	Approximation factor	Passes	Paper
Bipartite MCM	$2/3-\epsilon$	$O(\frac{1}{\epsilon}\log\frac{1}{\epsilon})$	[FKM+05]
	$1-\epsilon$	$O(\frac{1}{\epsilon^5})$	[EKMS12]
	$1-\epsilon$	$O(\frac{1}{\epsilon^2}\log\log\frac{1}{\epsilon})$	this paper
General MCM	$1-\epsilon$	$O((\frac{1}{\epsilon})^{1/\epsilon})$	[McG05]
Bipartite MWM	$1-\epsilon$	$O(\frac{1}{\epsilon^2}\log\frac{1}{\epsilon})$	this paper
General MWM	$1/6$	1	[FKM+05]
	$1/4.91$	1	[ELMS11]
	$1/2-\epsilon$	$O(\frac{1}{\epsilon^3})$	[McG05]
	$2/3-\epsilon$	$O(\frac{1}{\epsilon^2}\log\frac{1}{\epsilon})$	this paper
	$1-\epsilon$	$O(\frac{1}{\epsilon^4}\log n)$	this paper

As for lower bounds, Goel, Kapralov and Khanna [GKK12] have proved a lower bound of a $\frac{e-1}{e}$-approximation factor for MCM with single pass, even for Bipartite Graphs. Multi-pass lower bounds for MCM have been shown by Guruswami and Onak [GO16]. They proved that finding the size of the MCM exactly in p passes requires $n^{1+\Omega(1/p)}/p^{O(1)}$ space.

Future Work: In later years, there have been considerable improvement over the results for the Maximum Matching Problem that were known prior to this paper. Crouch and Stubbs [CS14] improved the approximation factor for single pass algorithims for MWM to $1/(4+\epsilon)$. Grigorescu et al. [GMZ16] improved the analysis of the last algorithm and showed that it achieves a $1/(3.5+\epsilon)$-approximation ratio. Paz and Schwartzman [PS17] further improved this factor to $1/(2+\epsilon)$ and this is the currently the best known result for MWM in a single pass.

2 Main Results

This paper largely improves the previously known results for MWM and Bipartite MCM both in terms of number of passes and the approximation factor. The main results of the paper are as follows:

- $(1-\epsilon)$-approximation for bipartite MCM in only $O(\frac{1}{\epsilon^2}\log\log\frac{1}{\epsilon})$ passes.

- A generalization of the former result for bipartite b-matching to get a $(1-\epsilon)$-approximation in $O(\frac{1}{\epsilon^3}\log n)$ many passes.
- $(1-\epsilon)$-approximation for bipartite MWM in $O(\frac{1}{\epsilon^2}\log\frac{1}{\epsilon})$ passes.
- $(2/3-\epsilon)$-approximation for MWM for general graphs in $O(\frac{1}{\epsilon^2}\log\frac{1}{\epsilon})$ passes.
- $O(1-\epsilon)$-approximation for MWM for general graphs in $O(\frac{1}{\epsilon^4}\log n)$ passes.

Even though a streaming algorithm with number of passes dependent on $\log n$ is not always practical, in theory, it is interesting to know that allowing this dependency on $\log n$ helps you to get arbitrarily close to the optimal solution. However, if you limit the number of passes to constant, you can still get arbitrarily close to 2/3-approximation.

To the best of our knowledge, this is one of the first papers that considers approximating the problem through looking at the LP. One can say, in addition to the main results mentioned earlier, this paper introduces a new framework for simulating primal-dual in semi-streaming model through MWU technique, which can be further investigated for approaching other combinatorial optimization problems.

3 Techniques and Overview of the Paper

The main theme of this paper is: "simulating primal-dual in semi-streaming through MWU". To elaborate more on the intuitions and ideas behind the proofs, we first need to describe how primal-dual and MWU methods work at a high level. Then we would proceed with stating the main lemmas and theorems of the paper and discuss our understanding of the proofs and what we think could be the ideas that lead to them.

3.1 Problem Statement

Formally the MCM and MWM problems on graph $G(V,E,w)$ are defined as the following LP's

$$\text{MCM}: \begin{cases} \max & \sum_{(i,j)\in E} y_{ij} \\ \text{s.t.} & \sum_{j:(i,j)\in E} y_{ij} \le 1 \ \ \forall i\in V \\ & y_{ij}\ge 0 \ \ \forall (i,j)\in E \end{cases} \quad \text{MWM}: \begin{cases} \max & \sum_{(i,j)\in E} w_{ij}y_{ij} \\ \text{s.t.} & \sum_{j:(i,j)\in E} y_{ij} \le 1 \ \ \forall i\in V \\ & y_{ij}\ge 0 \ \ \forall (i,j)\in E \end{cases}$$

In the offline setting both these problems can be solved by solving the LP using the MWU method in []. Unfortunately this method cannot be applied directly in the semi-streaming model because of the space restrictions (the oracle needs to store one variable for each edge). In the next section, we will discuss a popular framework for solving/approximating combinatorial optimization problems that can be described as LP's. This framework gives us hints on how to design a space efficient oracle that can be used in streaming.

3.2 Primal-Dual and MWU

Suppose we are given the following LP (called primal) and its dual LP:

$$\text{primal LP}: \begin{cases} \min & b^Tx \\ \text{s.t.} & A^Tx\ge c, \ \ x\ge 0 \end{cases} \quad \text{dual LP}: \begin{cases} \max & c^Ty \\ \text{s.t.} & Ay\le b, \ \ y\ge 0 \end{cases}$$

The whole idea behind the primal-dual approach is based on the following theorem: given solutions x and y for primal and dual respectively, x and y are optimal, if and only if both of the following "complementary slackness" conditions hold:

1. for any constraint i of the dual: $x_i(A_iy-b_i)=0$.

2. for any constraint j of the primal: $y_j(A_j^T x - c_j) = 0$.

In a primal-dual algorithm, the goal is to find a low cost solution x with the help of a solution y for the dual. Thus, we would start with some solutions x and y for the primal and the dual respectively, and try to modify them to make the previous conditions hold.

In most text-book primal-dual algorithms, you would start with *non-feasible* x with better than optimal objective value, and *feasible* y with sub-optimal objective. Throughout the run of the algorithm, you would keep the first condition while "approximating" the second one. Then, the steps you take towards improving the objective function for y, gives hints on how you should improve the x towards feasibility. The key aspect is that in the end, you could use the complementary slackness conditions to bound the cost of the primal solution constructed in this algorithm (the tightness of this bound is related to how closely you have approximated the slackness conditions).

For the purpose of this report the reader should remember two things about the classic primal-dual framework that was briefly discussed. First, the creative part of these primal-dual algorithms is in the choice of the direction you take to improve y. Different directions result in different approximations in the end. Second, after choosing the direction, we move across that direction as far as possible (until we hit a facet of the dual polytope and a constraint gets tight). Basically, this is what gives a bound on the running time of the algorithm.

The first complementary slackness constraint might look familiar to a reader who knows about the MWU method. In the MWU method, we are given some $x^{(0)}$ and an oracle that: in iteration t for $x^{(t)}$, computes $y^{(t)}$ that satisfies the first condition "in average". Then, for each i, $x_i^{(t)}$ is *increased* relative to the violation of the ith dual constraint. At the end of the algorithm, it can be shown that the output y "approximately" satisfies the first complementary slackness condition.

Of course, the creative part of a MWU algorithm is in designing the oracle. The second complementary slackness condition suggests that: When the primal constraint holds (i.e. $A_j^T x - c_j$ is big), y_j should be small and when the jth constraint of the primal is violated (i.e. $A_j^T x - c_j$ is small), y_j could be larger (but not too large). Interestingly, the oracle discussed in this paper seems to be following this hint.

A few remarks to contrast MWU and the described primal-dual framework: Firstly, through the run of the MWU algorithm, neither x nor y are necessarily feasible for their corresponding LP's but both of them are promised to have an objective value better than optimal. Secondly, after we decide the direction in which we want to change y or x, we do not necessarily make changes up to the point that some constraint becomes tight. In other words, the algorithm might have the option to move along *the same direction* for several iterations without making a constraint tight. We will discuss later how choosing a "good" direction for changing x and y can lead to better performance guarantees.

3.3 Designing the Oracle

In this section, we describe how to use the hints from the last section to design the oracle for the bipartite MCM. The following is the oracle for the MCM LP on input $x^{(t)}$ and a guess of optimal value α (the parameter δ is chosen later):

1. Let E_v be the set of violated primal constraints (i.e. $\{(i,j) : x_i + x_j < 1\}$)

2. Find a *maximal* matching S in E_v in a single pass

3. if $|S| < \delta\alpha$ report failure and return an "approximate" certificate of α being higher than optimal.

4. else return $y^{(t)}$ where $y_{ij}^{(t)} = \alpha/|S|$ for $(i,j) \in S$ and 0 otherwise.

Notice that the oracle only runs on E_v which is aligned with the hint from the last section on approximately satisfying the second complementary slackness condition. In other words, the oracle only sets y to be non-zero on edges that are violated. But again, we do not have a bound on size of E_v so it is not feasible to keep y for each edge on E_v. Instead, we compute a maximal matching in the second line which is a) easy to compute in one pass and b) is enough to improve the feasibility of x. This might not be "the best" direction to improve on x and y but it is a good starting point.

As mentioned in the last section, the promise is to maintain x and y to always have optimal values. But the problem is that we do not know the optimal value. So the algorithm takes a "guess" of the optimal value as input and then, we need to make sure there is a mechanism to determine if this guess is close to the actual value or not. This is exactly what line 3 of the oracle does. Basically, if $|S| < \delta\alpha$ means that there is not much room for improvement and in that case, x is close to being feasible. Then with a small addition to its cost (by $2|S| < 2\delta\alpha$) we can make it feasible. Of course, x had α cost in the beginning so in this case, the oracle has found a feasible x with cost $< (1 + 2\delta)\alpha$ which means that the optimal value must be $< (1 + 2\delta)\alpha$. Ideally, to rule out all the α's bigger than optimum, we would have wanted feasible x with value strictly less than α but we are only getting an "approximate" certificate that rules out guesses that are bigger than $1/(1 + 2\delta)$ times the actual optimal objective.

As for the 4th line of the algorithm, note that when $|S| \geq \delta\alpha$ we have $y_{ij}^{(t)} \leq 1/\delta$ for any edge $(i,j) \in S$. This is another place where we use the property that S is a matching: Because we can argue that the violation on each dual constraint is at most $1/\delta$. Indeed, this is called the "width" of the oracle and directly affects the number of times this oracle is called by the main algorithm.

3.4 The Initial Algorithm

The algorithm first makes a pass over the stream and computes a maximal matching. It is known that maximal matching is a 2-approximation for MCM. So we set our initial α to be twice the size of this maximal matching. To maintain our guess throughout the run of the algorithm, it is sufficient to scale down α whenever the oracle fails and try again (it can be proved that all the x's and y's can be reused for lower guesses). As the oracle only validates α approximately, it is natural to set the scaling factor to be proportional to δ.

Next we decide on how to set $x^{(t)}$. Let $u^{(t)}$ be the weight distribution the MWU algorithm keeps over the dual constraints, starting with $u^{(0)} = 1$. We set $x^{(t)}$ to be $\alpha p^{(t)}$ where $p^{(t)} = u^{(t)}/\sum_i u_i^{(t)}$. As promised, the objective value for $x^{(t)}$ is equal to α. Based on the analysis of MWU, the dependency of number of calls to the oracle, depends on the choice of $u^{(0)}$. To be more precise, the number of calls depends on $\max_i \psi_i/p_i^{(0)}$ where ψ_i is an upper-bound on all $p_i^{(t)}$'s. This basically says that for any iteration t, the probability for any i, namely $p_i^{(t)}$, should not be too different from its initial probability $p_i^{(0)}$ (the fraction is $1/n$ in the worst case). In the next section we will briefly discuss how this $u^{(0)}$ can be chosen smartly to bound $\max_i \psi_i/p_i^{(0)}$.

The rest of the algorithm is just running standard MWU. Observe that, every time we call the oracle, we need to run one more pass over the data. In the end, we get a solution y which is the average of the $y^{(t)}$'s and each constraint is slightly violated. Fortunately, with a simple scaling of this y we can get a feasible solution incurring a small hit to the objective function. Next, to extract an integral solution from this y, it is sufficient to solve the MCM problem on support of y (we can use the offline algorithm because support of y has at most nT many edges).

Note that the scaling part at the end of the algorithm is highly problem dependent. For the case of the MWM this rounding part is a whole other paper [DP14]. But the rest of the analysis goes through for the weighted version with some modification in the oracle. For example in line 3 of the oracle is modified as follows: Divide the edges into weight classes based on their weight and α and then compute S by greedily picking edges from maximal matchings computed in each weight class separately.

With carefully setting the parameters we get the following theorem (theorem 11 in the paper):

Theorem 1. *For any $\epsilon \leq \frac{1}{2}$ in $T = O(\frac{1}{\epsilon^3}\log n)$ calls to the oracle and $O(n(T + \log n))$ space we can compute a $(1-\epsilon)$ approximation for MWM in bipartite graphs.*

3.5 Improving the Initial Algorithm

The next interesting idea in this paper is that we can save on the number of passes with carefully choosing the direction of improving y. The key observation as stated previously is that the way x is modified by the MWU algorithm does not necessarily satisfy a new primal constraint in each iteration. But can we bound the number of iterations of MWU it takes to satisfy some constraint? Note that answering this question is important because if no new constraint is satisfied, E_v will not change and the output of the oracle will remain the same. Thus we can still re-use the $y^{(t)}$ from the previous iteration without calling the oracle again (and save on the number of passes).

Intuitively, it seems that increasing y_{ij} relative to the violation of the primal constraint is a good idea because the constraints that are less violated will not immediately get satisfied. This can be formulated as some other weights on the edges for which we can use the MWM oracle (so there is no need to worry about whether this change increases the number of iterations etc.). Surprisingly, it can be shown that for any $q \leq 1/\delta$, after using the output of the oracle from iteration t all the way up to iteration $t+q$, the initial $y^{(t)}$ still satisfies the constraints $x_i^{(t+q)}(A_i y^{(t)} - b_i) = 0$ in expectation. So it is safe to use the same oracle output for $1/\delta$ many iterations of MWU.

Putting this all together we have (theorem 15 of the paper):

Theorem 2. *For any $\epsilon \leq \frac{1}{2}$ in $T = O(\frac{1}{\epsilon^2}\log n)$ calls to the oracle and $O(n(T + \log n))$ space we can compute a $(1-\epsilon)$ approximation for MWM in bipartite graphs.*

The next technique is on removing the dependency of the number of passes on n. It can be shown that for fixed i, in any iteration t, value of $x_i^{(t)}$ is upper-bounded by a constant factor of $u_i^{(0)}$. This is good news because $x_i^{(t)} = \alpha p_i^{(t)}$ which implies that we can bound ψ_i by a constant factor of $u_i^{(0)}/\alpha$. Hence we get the bound of $O(\frac{\sum_i u_i^{(0)}}{\text{OPT}})$ for $\max_i \psi_i / p_i^{(0)}$, where OPT is the value of the optimal solution. This yet gives us another hint for improving the algorithm: If the number of vertices is proportional to OPT, even using the uniform $u^{(0)}$ removes the dependency on n; So the solution is finding a subgraph with fewer vertices that has approximately the same MCM optimal value as the original graph.

The idea is, instead of running the algorithm on the entire graph, we run it on a cleverly constructed subgraph, and setting the initial values of u_i^1 to an appropriate non-uniform distribution. Then, we prove that we still get a $(1-\epsilon)$-approximation for maximum weighted matching in bipartite graphs in $O(\frac{1}{\epsilon^2}\log\frac{\sum_i u_i^1}{OPT})$ many passes. For the case of bipartite MCM, $\sum_i u_i^1$ can be shown to be $O(OPT(\log 1/\epsilon))$, which gives an $O(\frac{1}{\epsilon^2}\log\log 1/\epsilon)$ pass $(1-\epsilon)$-approx algorithm for bipartite MCM. For the case of MWM, using an involced charging argument, $\sum_i u_i^1$

can be bounded by $OPT(1/\epsilon)$, which implies an $O(\frac{1}{\epsilon^2}\log 1/\epsilon)$ pass $(1-\epsilon)$-approx algorithm for bipartite MWM.

On a high level, the subgraph is constructed as follows: Starting with a maximal matching, repeatedly find maximal matchings between the vertices of the set of matched edges collected so far and the remaining unmatched vertices, and keep updating the set of edges. The graph induced by the set of edges collected after a certain number of iterations is the required subraph. The intuition behind construction of the subgraph and the charging argument used for MWM is however not very clear to us.

We end this section with a comment on MCM and MWM on general graphs. So far, we have been implicitly using the fact that the LP's we are working with are integral for the case of bipartite graphs (in the last step, where we extract an integral solution from the fractional solution of MWU). Of course, since the same LP has an integrality gap of $2/3$ for general graphs, all of the discussed results translate to $2/3(1-\epsilon)$-approximations for the general graph case.

But then, the question is: can we get $(1-\epsilon)$-approximation in number of passes independent of n? In order to address this question, we need to start over with an integral LP and do all the analysis done so far over again. But unfortunately, since the integral matching LP has exponentially many constraints, the number of iterations is linear in n. To overcome this problem, the trick is to ignore some of the constraints and then show the final scaling takes care of satisfying them. This trick reduces the number of iterations from linear in n to $O(\frac{1}{\epsilon^4}\log n)$. Eventually, we get the following theorem (theorem 28 in the paper):

Theorem 3. *In $T = O(\frac{1}{\epsilon^4}\log n)$ passes and $\tilde{O}(\frac{n^{O(1/\epsilon)}m}{\epsilon^9})$ time, we can compute a $(1-\epsilon)$-approximation for maximum weighted matching in general graphs. The algorithm uses $O(nT)$ space.*

4 Generalizations

This paper generalizes their techniques for MWM to the problem of (capacitated) b- matching, which is defined as follows:

Each vertex i has demand b_i and each edge (i,j) has capacity c_{ij} and weight w_{ij} . A multiset of edges is a b-matching if the multiplicity of each edge (i,j) is at most c_{ij} and i is the endpoint of at most b_i edges (counting the multiplicity of the edges) in the set. The maximum (capacitated) b-matching problem is to find a b-matching that maximizes the total weight of edges (keeping the multiplicity of the edges in account).

Their algorithm achieves an approximation factor of $(1-\epsilon)$ for the b-matching problem in $O(\frac{1}{\epsilon^3}\log n)$ passes.

This leads us to the question of what other problems can these techniques be generalized for? We see that in [ACG+15], similar MWU techniques have used to solve convex and semidefinite progams in small space.

References

[ACG+15] Kook Jin Ahn, Graham Cormode, Sudipto Guha, Andrew McGregor, and Anthony Wirth. Correlation clustering in data streams. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 2237–2246. JMLR.org, 2015.

[AG13] Kook Jin Ahn and Sudipto Guha. Linear programming in the semi-streaming model with application to the maximum matching problem. *Inf. Comput.*, 222:59–79, January 2013.

[CS14] Michael Crouch and Daniel M. Stubbs. Improved Streaming Algorithms for Weighted Matching, via Unweighted Matching. In Klaus Jansen, José D. P. Rolim, Nikhil R. Devanur, and Cristopher Moore, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2014)*, volume 28 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 96–104, Dagstuhl, Germany, 2014. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[DP14] Ran Duan and Seth Pettie. Linear-time approximation for maximum weight matching. *J. ACM*, 61(1):1:1–1:23, January 2014.

[EKMS12] Sebastian Eggert, Lasse Kliemann, Peter Munstermann, and Anand Srivastav. Bipartite matching in the semi-streaming model. *Algorithmica*, 63(1-2):490–508, June 2012.

[ELMS11] Leah Epstein, Asaf Levin, Julián Mestre, and Danny Segev. Improved approximation guarantees for weighted matching in the semi-streaming model. *SIAM J. Discrete Math.*, 25:1251–1265, 2011.

[FKM$^+$05] 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.

[GKK12] Ashish Goel, Michael Kapralov, and Sanjeev Khanna. On the communication and streaming complexity of maximum bipartite matching. In *Proceedings of the Twenty-third Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '12, pages 468–485, Philadelphia, PA, USA, 2012. Society for Industrial and Applied Mathematics.

[GMZ16] Elena Grigorescu, Morteza Monemizadeh, and Samson Zhou. Streaming weighted matchings: Optimal meets greedy. *CoRR*, abs/1608.01487, 2016.

[GO16] Venkatesan Guruswami and Krzysztof Onak. Superlinear lower bounds for multipass graph processing. *Algorithmica*, 76(3):654–683, November 2016.

[McG05] 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, pages 170–181, Berlin, Heidelberg, 2005. Springer-Verlag.

[PS17] Ami Paz and Gregory Schwartzman. A (2 + &#1013;)-approximation for maximum weight matching in the semi-streaming model. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '17, pages 2153–2161, Philadelphia, PA, USA, 2017. Society for Industrial and Applied Mathematics.