Improved Streaming Algorithms for Weighted Matching, via Unweighted Matching

Michael Crouch and Daniel M. Stubbs

University of Massachusetts, Amherst, U.S.

Abstract

We present a $(4+\epsilon)$ approximation algorithm for weighted graph matching which applies in the semistreaming, sliding window, and MapReduce models; this single algorithm improves the previous best algorithm in each model. The algorithm operates by reducing the maximum-weight matching problem to a polylog number of copies of the maximum-cardinality matching problem. The algorithm also extends to provide approximation guarantees for the more general problem of finding weighted independent sets in p-systems (which include intersections of p matroids and p-bounded hypergraph matching).

1998 ACM Subject Classification F.1.1 [Computation by Abstract Devices]: Models of Computation – relations between models, F.1.2 [Computation by Abstract Devices]: Modes of Computation – online computation, G.2.2 [Discrete Mathematics]: Graph Theory – graph algorithms, hypergraphs

Keywords and phrases Streaming Algorithms, Graph Matching, Weighted Graph Matching, MapReduce, Independence Systems

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2014.96

1 Introduction

Finding large matchings (that is, sets of edges which do not have any endpoints in common) is a pivotal problem in graph algorithms, and has seen significant recent work in a variety of "big data" models. In particular, there has been a series of papers in the semi-streaming graph model, where we have one-way read access to a stream of weighted edges, but have only $O(n \operatorname{polylog} n)$ memory (enough to store only a sparse subgraph of the input). The paper which first introduced this model [9] provided a 6-approximation algorithm for maximum weighted graph matching. This was improved to a 5.828-approximation in [15]; a 5.585-approximation in [17]; and finally the current best, a $4.911+\epsilon$-approximation in [8]. Other work has looked at maximum weighted matching in the MapReduce model [13] and the sliding-window stream model [6], and has examined more general submodular-function matching problems in the semistreaming model [2, 5].

We present an algorithm for maximum weighted matching which is applicable in all of these models and which improves on the best known approximation guarantees in all of them. Our algorithm also extends to a generalization of maximum weighted graph matching: the problem of finding maximum-weight independent sets in p-systems. p-systems are a type of independence system which generalize both matching on p-bounded hypergraphs and intersections of p matroids.

Our algorithm works by reducing a single maximum weighted matching problem to a number of unweighted matching problems, then combining the unweighted matchings according to a simple greedy heuristic. The structure of our reduction is related to an existing streaming algorithm for maximum weighted matching [8]. That algorithm partitions incoming edges into multiplicatively-spaced weight classes; it maintains a separate greedy

17th Int'l Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX'14) / 18th Int'l Workshop on Randomization and Computation (RANDOM'14).
Editors: Klaus Jansen, José Rolim, Nikhil Devanur, and Cristopher Moore; pp. 96–104

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

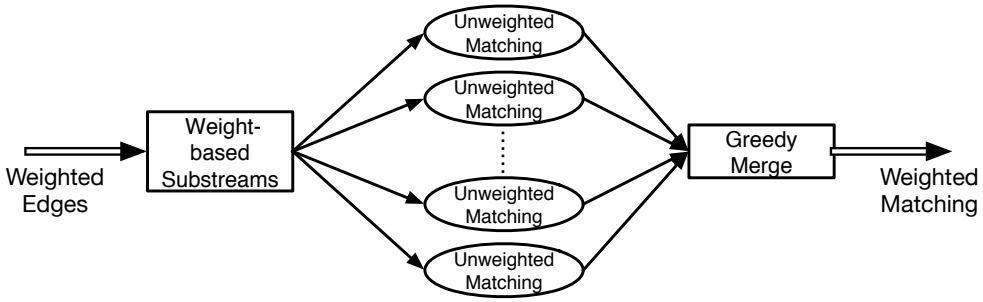

Figure 1 Block Diagram of the Weighted Matching Algorithm.

matching on the edges in each weight class. At the end of the stream, they greedily merge the matchings from largest to smallest.

Rather than partitioning the edges into disjoint weight classes, our algorithm uses weight classes that are unbounded above: classes that admit smaller-weight edges are supersets of all of the "more exclusive" classes. The stronger relationship between different weight classes created by this approach allows a more unified and global charging argument, giving an improved approximation ratio and broader applicability, including to the case of more general independence systems than just graph matchings.

In §2 we define our model more specifically and state our main results. In §3 we present the algorithm. In §4 we summarize the improvements that our that our algorithm yields for weighted matching and weighted independent set problems in the streaming, sliding window, and MapReduce models. In §6 we comment on improvements for specific cases and outline future work.

2 Definitions and Results

In this section, we define our model of "streaming reductions"; present our result for maximum weighted graph matching; recall the definitions of p-systems; and state the extensions of our results to p-systems.

2.1 Streaming Reductions

Our algorithm will operate by transforming a maximum-weight matching problem into a polylogarithmic number of maximum-cardinality matching problems (Fig. 1). This is an example of a class of reductions which we believe may be of particular interest in big data models: Turing reductions which make a polylogarithmically-bounded number of queries and which are "nonadaptive" (in the sense that the input to one query does not depend on the output of any other query).

In this model, a reduction from problem A to problem B consists of processing the input to problem A into the inputs to polylog instances of B; solving the B instances; then processing the B outputs into the output to problem A. These models form a restricted class of the approximation-preserving streaming reductions used in [4].[1]

[1] Other papers introducing reductions in the streaming model defined many-one reductions between decision problems via a generalization of string homomorphisms [3, 14]; it is not readily apparent how to apply this work to approximation problems.

These reductions are natural choices for models where resources of interest are closed under polylogarithmic blowup, including the semistreaming graph model. Requiring the subproblems to be nonadaptive allows them to be easily parallelized. The resources used by the preprocessing and postprocessing steps can be restricted to preserve the classes of interest; in our case, the preprocessing step is merely testing the weight of each edge, and the postprocessing step is a greedy merge. Many existing streaming algorithms operate by reducing to polylog nonadaptive subproblems, including the precision sampling framework [1] and Indyk/Woodruff L_p norm estimators [10].

We say that a reduction from A to B is p-approximate if, for any $\alpha \geq 1$, given an α-approximate solution to each B subproblem we can generate an αp-approximate solution to the A problem.

2.2 Main Result

Section 3 presents the proof of our main result:

▶ **Theorem 1.** *There is a $2(1+\epsilon)$-approximate nonadaptive Turing reduction from the problem of maximum-weight matching to the problem of maximum-cardinality matching. The reduction uses $O(\frac{1}{\epsilon}\log n)$ copies of maximum-cardinality matching.*

The reduction in Theorem 1 uses extremely minimal preprocessing (separating edges by weight) and minimal postprocessing (performing a greedy merge of the edge sets).

Since greedy matching provides a 2-approximation to maximum-cardinality matching, from Theorem 1 we immediately find:

▶ **Corollary 2.** *We can perform a $4(1+\epsilon)$-approximation to maximum-weight matching, using $O(\frac{1}{\epsilon}\log n)$ times the resources necessary to keep a greedy matching.*

The consequences of Corollary 2 in specific models are described in Section 4.

2.3 Independence Systems

Our algorithm extends to a class of independence systems called *p-systems*. An *independence system* is a pair (S, I) comprising a finite set S and a set I of subsets of S (the "independent sets") such that

1. $\emptyset \in I$
2. For $X \subseteq X'$, $X' \in S \Rightarrow X \in S$.

An independence system (S, I) is called a p-system if, for any $A \subseteq S$, the ratio between the largest and smallest maximal independent subsets of A is at most p. Graph matching forms a 2-system where S is the set of edges and where a set of edges is independent if no two edges share an endpoint. More generally, p-hypergraph matching is a p-system, as is the intersection of p matroids. For more detail on p-systems, see e.g. [11].

Given a p-system (S, I), the *maximum-cardinality independent set* problem is the problem of finding an independent set with the largest number of elements. Given a weight function $w : S \to \mathbb{R}_{\geq 0}$, the *maximum-weight independent set* problem is the problem of finding an independent set $X \in I$ which maximizes $\sum_{x \in X} w(x)$. These problems naturally extend the problems of finding maximum matchings on unweighted or weighted graphs.

Section 3.1 shows that Thm. 1 extends to:

▶ **Theorem 3.** *Let (S, I) be a p-system. Then there is a $p(1+\epsilon)$-approximate nonadaptive Turing reduction from the problem of maximum-weight independent set on (S, I) to the problem of maximum-cardinality independent set on (S, I). The reduction uses $O(\frac{1}{\epsilon}\log n)$ copies of maximum-cardinality independent set.*

From the definition of p-systems, a greedily maximized set is always a p-approximate maximum cardinality matching. From Theorem 3 we thus immediately find:

▶ **Corollary 4.** *We can perform a $p^2(1+\epsilon)$ approximation to maximum-weight independent set on any p-system, using $O(\frac{1}{\epsilon}\log n)$ times the resources necessary to greedily compute a maximal independent set on that p-system.*

The consequences of Corollary 4 in specific models are described in Section 4.

3 Algorithm

In this section we present a proof of Theorem 1.

Consider a graph G on vertex set V. Let $n = |V|$. Let the input E be a stream of edges from $V \times V$, where each $e \in E$ is annotated with its weight $w(e)$.

For $i \in \mathbb{Z}$, we define substreams E_i, each containing all edges with weight above threshold $(1+\epsilon)^i$:

$$E_i \triangleq \{e \in E \mid w(e) \geq (1+\epsilon)^i\} \tag{1}$$

Note that i can be negative, but we assume that the range of possible weights $w(e)$ is polynomially bounded in n, so that we only need to consider substreams for $O(\frac{1}{\epsilon}\log n)$ values of i.[2]

To perform the reduction, assume that for $\alpha > 1$ we have for each E_i some matching $C_i \subseteq E_i$ which contains at least $\frac{1}{\alpha}$ times as many elements as the maximum-cardinality matching on E_i. We then greedily construct a matching T by considering the edges in C_i in descending order of i, and at the end we output T. The top-level structure of the algorithm is summarized in Figure 1.

Consider a fixed maximum-weight matching Opt on E. For each class E_i let $T_i = T \cap E_i$ be the set of edges output from E_i.

▶ **Lemma 5.** *For each i, $|T_i| \geq \frac{1}{2\alpha}|E_i|$.*

Proof. For each class E_i let Opt_i be a maximum-cardinality matching on E_i. Our oracle returns C_i with $|C_i| \geq \frac{1}{\alpha}|\text{Opt}_i|$, and thus with $|C_i| \geq \frac{1}{\alpha}|\text{Opt} \cap E_i|$.

We greedily add as many edges as possible from C_i to T_i. Since T_i and C_i are both matchings, each edge in T_i can share endpoints with at most two edges of C_i. Thus, as long as $|T_i| < \frac{1}{2}|C_i|$, C_i contains at least one edge which is not adjacent to any edge yet in T_i.

The greedy merge can thus always add edges from C_i to T_i until $|T_i| \geq \frac{1}{2}|C_i|$. Combining with the above we then have $|T_i| \geq \frac{1}{2\alpha}|\text{Opt} \cap E_i|$. ◀

We now must argue that the cardinality constraint in Lemma 5 leads to the weight-based approximation ratio in Theorem 1.

▶ **Lemma 6.** *There exists a function f from Opt to T such that for each $e \in \text{Opt}$, $w(e) \leq (1+\epsilon)w(f(e))$ and for each $t \in T$, there are at most 2α edges $e \in \text{Opt}$ such that $f(e) = t$.*

[2] If we do not have this guarantee, we can keep track of the highest-weight edge seen so far, and discard any items with less than $2\epsilon/n$ times that weight. A matching made entirely of these discarded edges is then at most an ϵ fraction of the output weight (since the weight we output is at least the weight of the largest edge).

Proof. We define f inductively, considering $\text{OPT} \cap E_i$ in descending order by i and picking $f(e)$ from among the elements of T_i that have fewer than 2α edges already associated with them. This restriction will guarantee that $f(e)$ is from at least as high a class as e, which gives us that $w(e) \leq (1+\epsilon)w(f(e))$. By Lemma 5 there are always enough elements in T_i to avoid overcrowding.[3] ◀

Lemma 6 leads immediately to a charging argument which proves Theorem 1: every edge $e \in \text{OPT}$ is an element of the preimage $f^{-1}(t)$ for some t, and for each t

$$\sum_{e \in f^{-1}(t)} w(e) \leq \left|f^{-1}(t)\right|(1+\epsilon)w(t) \leq 2\alpha(1+\epsilon)w(t) \tag{2}$$

so we have

$$w(\text{OPT}) = \sum_{e \in \text{OPT}} w(e) = \sum_{\substack{e \in f^{-1}(t) \\ t \in T}} w(e) \leq \sum_{t \in T} 2\alpha(1+\epsilon)w(t) = 2\alpha(1+\epsilon)w(T) \tag{3}$$

3.1 Extension to p-Systems

The algorithm operates similarly for reducing maximum-weight independent sets in arbitrary p-systems to copies of the problem of finding maximum-cardinality independent sets (Theorem 3). Most of the proof is similar, with independent sets replacing matchings and with p replacing the multiplicative factor 2. The equivalent of Lemma 5 is somewhat more involved to prove:

▶ **Lemma 7.** *For each i, $|T_i| \geq \frac{1}{\alpha p}|E_i|$.*

Proof. For each class E_i let OPT_i be a maximum-cardinality independent set on E_i; we again consider an oracle which returns an independent set C_i of cardinality $|C_i| \geq \frac{1}{\alpha}|\text{OPT}_i| \geq \frac{1}{\alpha}|\text{OPT} \cap E_i|$.

E_i is a subset of a p-system and thus also forms a p-system. Now consider maximal independent sets on $C_i \cup T_i$ (recall $C_i \cup T_i \subseteq E_i$). We have that C_i is a maximal independent set of size $|C_i|$. Thus, by the definition of p-systems, no maximal independent subset of $C_i \cup T_i$ can have size less than $\frac{1}{p}|C_i|$.

The greedy merge can thus always add elements from C_i until $|T_i| \geq \frac{1}{p}|C_i|$, yielding $|T_i| \geq \frac{1}{\alpha p}|\text{OPT} \cup E_i|$. ◀

The remainder of the proof of Theorem 3 proceeds similarly to the proof of Theorem 1.

4 Extensions

The results of Corollaries 2 and 4 improve the best known algorithms for many matching problems. These are summarized in Fig. 2.

Maximum weighted graph matching (MWM) has been studied in a variety of models; the algorithm of Theorem 1 provides an approximation guarantee in any big data model where we are capable of performing greedy matching on weight-based substreams of the data. Several of these applications are explained below; each of these is an improvement of the previous best results in these models.

[3] In the case where 2α is fractional, we allow edges from OPT to be mapped "partially" to multiple edges from T so long as this doesn't result in more than 2α total OPT edges mapped to any edge of T.

Problem	Model	Previous	This Paper
MWM	One-pass streaming	4.911 [8]	4
MWM	One-pass sliding window	9.027 [6]	6
MWM	MapReduce	8 [13]	4
3-MWM	One-pass streaming	9.899 [5]	9
2-MWIS	One-pass streaming	5.828 [2]	4
3-MWIS	One-pass streaming	9.899 [2]	9

Figure 2 Approximation factor improvements over previous results. ϵ factors have been omitted.

The semi-streaming model (defined in [9]) allows one-way access to a stream of weighted edges on a machine limited to $O(n \operatorname{polylog} n)$ memory. A series of papers has provided improved approximation guarantees in this model [9, 15, 17, 8]; the current best is a $4.911 + \epsilon$ approximation [8]. Keeping a maximal matching in the semistreaming model is trivial (see e.g. [9]) and the machine has enough memory space to store one maximal matching for each of the $O(\log n)$ many weight classes, thus we find

▶ **Corollary 8.** *There is a $4 + \epsilon$ approximation algorithm for maximum weighted matching in the semistreaming model.*

Ashwinkumar 2011 [2] extended semistreaming matching algorithms to the more general case of finding maximum-weight independent sets in p-intersection systems (p-MWIS); this was further developed by Chakrabarti et al. 2013 to include weighted matching on hypergraphs of degree p (p-MWM). Our algorithm improves their approximation ratio of $2(p + \sqrt{p(p-1)}) - 1$ for the most practical $p = 2$ and $p = 3$ cases (see Figure 2).

▶ **Corollary 9.** *There is a semistreaming algorithm for finding the Maximum-Weight Independent Set on a p-system with approximation ratio $p^2 + \epsilon$.*

▶ **Corollary 10.** *There is a semistreaming algorithm for finding the Maximum-Weight Matching on a degree p hypergraph with approximation ratio $p^2 + \epsilon$.*

In the related semi-streaming "sliding window" graph model [7, 6], there is a fixed window length $L \in \omega(n \operatorname{polylog} n)$, and we are interested in maintaining (at all times) a maximum matching over the most recent L edges. We are again limited to $O(n \operatorname{polylog} n)$ memory space. In this model, only a $3 + \epsilon$ approximation to unweighted matching is known [6], and we thus find:

▶ **Corollary 11.** *There is a $6 + \epsilon$ approximation algorithm for maximum weighted matching in the semi-streaming sliding window model.*

The class $\mathcal{MRC}^0$ [12] is a theoretical model for MapReduce computations achievable with a constant number of rounds. In this model, even though the edge set does not fit on any individual processor, it is possible to find a maximal matching [13] (and thus a 2-approximation of the maximum unweighted matching). This immediately yields an improvement over the previous best-known 8-approximation algorithm for maximum weighted matching [13], with no additional communication cost (since the merge can be performed on a single processor).

▶ **Corollary 12.** *There is a constant-round $4 + \epsilon$ approximation algorithm for maximum weighted matching in the MapReduce model $\mathcal{MRC}^0$.*

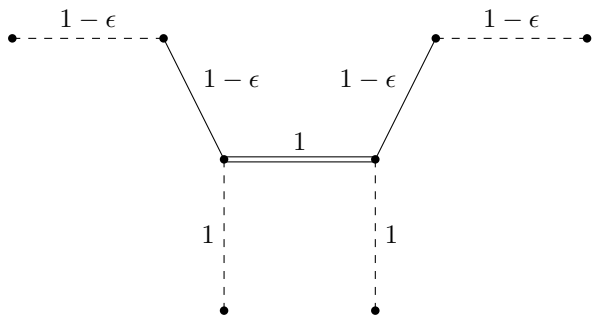

Figure 3 Graph with output weight 1 and optimum matching weight $4 - 2\epsilon$. In the weight class $[1, \infty)$ the double-lined edge is remembered; in the weight class $[1 - \epsilon, \infty)$ the two single-lined edges are remembered. The double-lined edge is output. The dashed edges are the optimum matching (and are not remembered in either class).

5 Lower Bounds for Graph Matching

In this section we consider the case of maximum weighted graph matching, and we present graph constructions which prove lower bounds on the approximation ratios achievable by our algorithm. These constructions extend to a general family of techniques which also includes previous weight-class-based approaches to maximum weighted matching.

The algorithm presented in §3 computes its output matching by performing a greedy matching on remembered edges, in decreasing order of weight. Our analysis showed that this was a $(4 + \epsilon)$-approximation. In Fig. 3 we present a graph where the algorithm's approximation ratio is $4 - 2\epsilon$, showing that our analysis is tight to within $1 + \epsilon$ factors.

In the graph of Fig. 3, the greedy matching on the remembered edges has weight 1, but the maximum weight matching on remembered edges has weight $2 - 2\epsilon$. In practice, many applications may be able to spend the post-processing time necessary to find the maximum-weight matching on the remembered edges (which are, after all, a sparse subgraph of the original graph). An obvious question is whether this post-processing can provide a stronger approximation guarantee.

The graph of Fig. 4 shows that our algorithm cannot achieve better than a 3.5 approximation, even when we output the maximum-weight matching on all of the remembered edges. Only remembered edges are drawn. Edges arrive in increasing order by weight, with the remembered edges appearing before other edges of the same weight. When the graph of Fig. 4 is extended upwards, any algorithm which uses greedy matchings on weight-based substreams cannot do better than a 3.5-approximation, because it is incapable of remembering any edge from Opt. This class of algorithms includes our algorithm and also the previous best algorithm of [8].

6 Conclusion

For specific systems of interest, we may be able to obtain stronger approximation guarantees, particularly by being more clever in our post-processing of memory. The case of one-pass streaming algorithms for graph matching is of particular interest. An obvious improvement to our algorithm is to calculate the maximum matching on all edges held in memory (via e.g. the Blossom algorithm [16]) rather than performing a greedy matching on edges held. We conjecture that this improvement yields a $(3.5 + \epsilon)$-approximation, tight to the lower bound shown in Fig. 4.

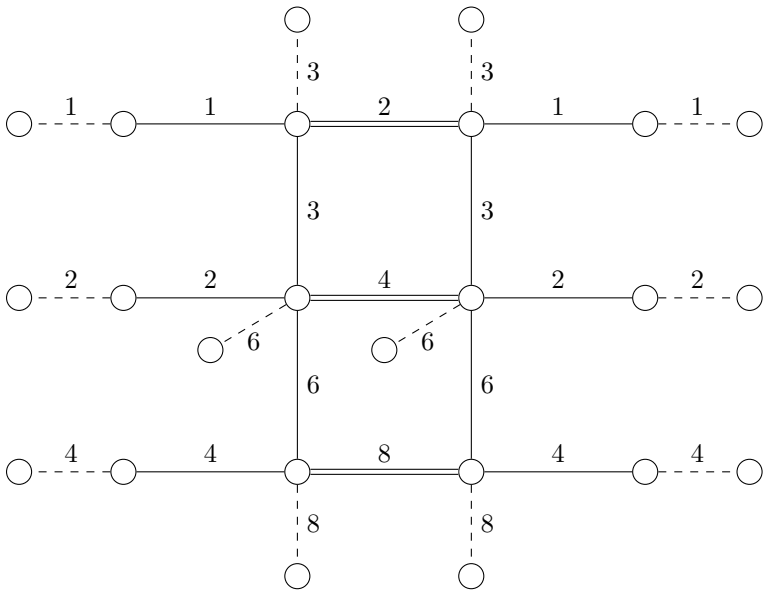

Figure 4 A graph which, when extended upwards, approaches approximation ratio 3.5. The dotted-line edges form the optimal matching, but are not remembered in any weight class. Solid edges are remembered but not output; double-lined edges are remembered and output. These remembered edges are produced by a stream where the edges arrive in order of increasing weight, with to-be-remembered edges arriving first. The reader can verify that within each weight class, the set of remembered edges is maximal.
On this graph, the output has weight 14, and the optimum matching has weight 48, for an approximation ratio of ≈3.429. The graph can be extended upwards, with each new layer including a single new output edge which decreases in weight by a power of 2; the approximation ratio quickly approaches 3.5.

Since each edge in our stream may fall in multiple weight classes, we may need to update $\Omega(\log n)$ matchings to process each incoming edge, leading to an $\Omega(\log n)$ sequential processing time. In contrast, the previous best semistreaming algorithm [8] used non-overlapping weight classes and required time $O(1)$, but obtained a worse approximation ratio. The trade-off between approximation quality and per-element processing time may be worth further study.

References

1 Alexandr Andoni, Robert Krauthgamer, and Krzysztof Onak. Streaming algorithms via precision sampling. *FOCS*, 2011.

2 B.V. Ashwinkumar. Buyback problem - approximate matroid intersection with cancellation costs. In *ICALP*, pages 379–390, 2011.

3 Ajesh Babu, Nutan Limaye, J Radhakrishnan, and Girish Varma. Streaming algorithms for language recognition problems. *Theoretical Computer Science*, 494:13–23, 2013.

4 Ziv Bar-Yossef, Ravi Kumar, and D Sivakumar. Reductions in streaming algorithms, with an application to counting triangles in graphs. In *SODA*, pages 623–632, January 2002.

5 Amit Chakrabarti and Sagar Kale. Submodular maximization meets streaming: Matchings, matroids, and more. *IPCO*, 2014. To appear.

6 Michael Crouch, Andrew McGregor, and Daniel Stubbs. Dynamic graphs in the sliding-window model. *ESA*, 2013.

7 Mayur Datar, Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Maintaining stream statistics over sliding windows. *SIAM Journal on Computing*, 31(6):1794, 2002.

8 Leah Epstein, Asaf Levin, Julián Mestre, and Danny Segev. Improved approximation guarantees for weighted matching in the semi-streaming model. *SIAM Journal on Discrete Mathematics*, 25(3):1251–1265, January 2011.

9 Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. *Theoretical Computer Science*, 348(2-3):207–216, December 2005.

10 Piotr Indyk and David P Woodruff. Optimal approximations of the frequency moments of data streams. In *STOC*, pages 202–208. ACM, 2005.

11 Thomas A Jenkyns. The efficacy of the "greedy" algorithm. *Proceedings of the 7th Southeastern Conference on Combinatorics, Graph Theory and Computing*, pages 341–350, 1976.

12 Howard J. Karloff, Siddharth Suri, and Sergei Vassilvitskii. A model of computation for MapReduce. In *SODA*, pages 938–948, 2010.

13 Silvio Lattanzi, Benjamin Moseley, Siddharth Suri, and Sergei Vassilvitskii. Filtering: a method for solving graph problems in MapReduce. In *SPAA*, New York, New York, USA, 2011. ACM Press.

14 Frédéric Magniez, Claire Mathieu, and Ashwin Nayak. Recognizing well-parenthesized expressions in the streaming model. In *STOC*, pages 261–270. ACM, 2010.

15 Andrew McGregor. Finding graph matchings in data streams. *APPROX-RANDOM*, 2005.

16 Silvio Micali and Vijay V. Vazirani. An O(sqrt(|V|) |E|) algorithm for finding maximum matching in general graphs. In *FOCS*, pages 17–27, 1980.

17 Mariano Zelke. Weighted matching in the semi-streaming model. *Algorithmica*, pages 669–680, 2012.