Maximum-Weight Matching in Sliding Windows and Beyond

Leyla Biabani
Department of Computer Science, TU Eindhoven, The Netherlands

Mark de Berg
Department of Computer Science, TU Eindhoven, The Netherlands

Morteza Monemizadeh
Department of Computer Science, TU Eindhoven, The Netherlands

Abstract

We study the maximum-weight matching problem in the sliding-window model. In this model, we are given an adversarially ordered stream of edges of an underlying edge-weighted graph $G(V, E)$, and a parameter L specifying the *window size*, and we want to maintain an approximation of the maximum-weight matching of the current graph $G(t)$; here $G(t)$ is defined as the subgraph of G consisting of the edges that arrived during the time interval $[\max(t - L, 1), t]$, where t is the current time. The goal is to do this with $\tilde{O}(n)$ space, where n is the number of vertices of G. We present a deterministic $(3.5 + \varepsilon)$-approximation algorithm for this problem, thus significantly improving the $(6 + \varepsilon)$-approximation algorithm due to Crouch and Stubbs [5].

We also present a generic machinery for approximating subadditve functions in the sliding-window model. A function f is called *subadditive* if for every disjoint substreams A, B of a stream S it holds that $f(AB) \leqslant f(A) + f(B)$, where AB denotes the concatenation of A and B. We show that given an α-approximation algorithm for a subadditive function f in the insertion-only model we can maintain a $(2\alpha + \varepsilon)$-approximation of f in the sliding-window model. This improves upon recent result Krauthgamer and Reitblat [14], who obtained a $(2\alpha^2 + \varepsilon)$-approximation.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases maximum-weight matching, sliding-window model, approximation algorithm, and subadditve functions

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2021.73

Funding The work in this paper is supported by the Dutch Research Council (NWO) through Gravitation-grant NETWORKS-024.002.003.

1 Introduction

A *matching* in a graph $G(V, E)$ is a subset $M \subseteq E$ of pairwise non-adjacent edges. Matchings play an important role in applications and they form a central concept in combinatorial and algorithmic graph theory. They have been studied extensively; there is even a book, by Lovasz and Plummer, dedicated completely to matching theory [16]. One of the most studied algorithmic problems concerning matchings is the *maximum-matching problem*. In the unweighted version, the goal is to compute a maximum-cardinality matching and in the weighted version – here every edge in the input graph has a non-negative weight – the goal is to compute a maximum-weight matching (MWM). In the classical offline setting we are given the graph G completely in advance and we have enough space to store all vertices and edges. In this setting, the fastest algorithm to compute a maximum matching is still the 30-years-old algorithm due to Micali and Vazirani [17] with running time $O(m\sqrt{n})$, where $n := |V|$ and $m := |E|$.

In this paper, we study the maximum-weight matching problem in a streaming setting. Here we are given a stream S of edges of an underlying graph $G(V, E)$ and we only have $\tilde{O}(n) := O(n \cdot \text{polylog}(n))$ storage available. This means that we cannot store all edges from G, and it becomes impossible to compute a solution that is guaranteed to be optimal. The goal thus becomes to maintain a matching on the current graph whose total weight is as close to optimal as possible. In the *insertion-only model*, the current graph $G(t)$ is the graph whose edge set consists of all edges that have arrived up to the current time t. We will be working in the *sliding-window model* [6], where we are given a window length L and the current graph $G(t)$ consists of the last L edges from the stream. Thus, at any time t we are interested in approximating the maximum-weight matching of the edges that are within the window $[\max(t - L, 1), t]$.

Our results. We present a $(3.5 + \varepsilon)$-approximation for the maximum-weight matching problem in the sliding-window model. The following theorem states our main result more formally.

▶ **Theorem 1.** *Let $G(V, E)$ be a graph with $n = |V|$ vertices and let $w : E \to \mathbb{R}^+$ be a function that assigns a weight $w(e)$ to each edge $e \in E$ such that the ratio of the maximum edge weight to the minimum edge weight is at most W for some $W = n^{O(1)}$. Then, for any given $\varepsilon > 0$, there exists a deterministic algorithm that maintains a $(3.5 + \varepsilon)$-approximate maximum-weight matching in the sliding-window model using $\tilde{O}(n)$ space.*

A comparison of our result with the existing streaming algorithms for maximum-weight matchings is given in Table 1.

The algorithm in Theorem 1 takes the algorithm by Paz and Schwartzman [18] as a starting point. Our $(3.5+\varepsilon)$-approximation is then based on two contributions: a mechanism to convert an algorithm in the insertion-only model into an algorithm in the sliding-window model, and an intricate analysis of the resulting algorithm in the case of maximum-weight matchings. The mechanism to obtain a sliding-window algorithm from an insertion-only algorithm actually works in a much more general setting, namely when we want to compute subadditive functions, defined as follows. Let f be a function that assigns a value to a data stream. Then f is called *subadditive* if for every disjoint substreams A, B of a stream S it holds that $f(AB) \leqslant f(A) + f(B)$, where AB denotes the concatenation of A and B. We say that f is *monotone* if for all $A \subseteq B$ we have $f(A) \leqslant f(B)$. Observe that (the cost of a) maximum weight matching is a subadditive and monotone function. We prove the following result in Section 4.

▶ **Theorem 2.** *Let $0 < \varepsilon \leqslant 1/2$ and $\alpha \geqslant 1$ be two parameters. Let f be a function defined on streams that is subadditive, non-negative, and monotone. Let $\sigma := f_{\max}/f_{\min^+}$, where $f_{\min^+} := \min\{f(X) : X \text{ is a substream of the input and } f(X) > 0\}$ and $f_{\max} := \max\{f(X) : X \text{ is a substream of the input}\}$. Suppose there is an algorithm in the insertion-only streaming model that α-approximates f using space s. Then, there is an algorithm in the sliding-window model that maintains a $(2\alpha + \varepsilon)$-approximation of f using $O(\varepsilon^{-1} s \cdot \log \sigma)$ space.*

Previous Work. Given the prominence of the matching problem, it is not surprising that it has already received considerable attention in the streaming setting.

In the insertion-only model, a maximal matching – a matching M such that adding any edge from E to M would violate the condition that the edges are pairwise non-adjacent – can be computed greedily using $O(n)$ space. This immediately gives a 2-approximation for maximum-cardinality matching (MCM) [9]. If the stream is in adversarial order, obtaining an

approximation ratio better than 2 for the MCM problem is one of the most important open problems in streaming algorithms. Interestingly, Kapralov [12] showed that any streaming algorithm that achieves an approximation ratio better than $e/(1-e) \approx 1.58$ must use $n^{1+\Omega(\frac{1}{\log\log n})}$ storage. In the *random-order model*, where the edges from E arrive in random order, better approximation ratios than 2 have been obtained [13, 10, 7, 8, 1]. In particular, very recently, Bernstein [1] showed that we can compute a $(\frac{3}{2}+\varepsilon)$-approximate matching in random order streams using $\tilde{O}(n)$ space.

Maximum-weighted matchings (MWM) have also been studied in the insertion-only model. Crouch and Stubbs [5] presented a technique that makes it possible to turn a c-approximation algorithm for the MCM problem into a $2(1+\varepsilon)c$-approximation algorithm for the MWM problem. A combination of this reduction and the greedy matching achieves a $(4+\varepsilon)$-approximation algorithm for the maximum-weight matching problem in the insertion-only model. In a breakthrough result, Paz and Schwartzman [18] developed a $(2+\varepsilon)$-approximation algorithm for the maximum-weight matching problem in the insertion-only model, using $O(n\log^2 n)$ bits of space. Later, Ghaffari and Wajc [11] improved the space of Paz and Schwartzman's algorithm to $O(n\log n)$ bits of space.

We now turn our attention to the sliding-window model. Braverman and Ostrovsky [2] introduced *smooth histograms*, a powerful framework to maintain a class of functions, called smooth function in the sliding-window model. Crouch, McGregor, and Stubbs [4] showed that the smooth-histograms technique can be applied to maintain an approximately maximal matching that achieves a $(3+\varepsilon)$-approximation of the maximum-cardinality matching. They also presented a 9.027-approximation algorithm for the maximum-weight matching in the sliding-window model. This can be improved using the already mentioned (more recent) technique by Crouch and Stubbs [5] to turn a c-approximation algorithm for the MCM problem into a $2(1+\varepsilon)c$-approximation algorithm for the MWM problem. Using this reduction and the algorithm of [4] we can achieve $(6+\varepsilon)$-approximation weighted matching in the sliding-window model. This reduction, which yields the best previously known result for the MWM problem in the sliding-window model, partitions the edges into weight classes and maintains a matching on the edges in each weight class. At the end of the stream, it greedily merges the matchings from largest weight class to smallest weight class. However, our algorithm is based on the algorithm by Paz and Schwartzman [18] that is explained in Section 2.

We next explain the previous work on subadditive functions in data streams. Let f be a function defined on streams, which is subadditive, non-negative, and monotone. Recently, Krauthgamer and Reitblat [14] showed that given a streaming algorithm $\mathcal{ALG}$ that α-approximates f using space s, we can develop a sliding-window algorithm that $(2\alpha^2+\varepsilon)$-approximates f using space $O(\varepsilon^{-1}s\cdot\log\sigma)$, where σ is ratio of the maximum value of f to the minimum value of f. They also showed that if $\mathcal{ALG}$ is monotone and subadditive, we can reduce the approximation factor of the sliding-window algorithm down to $(2\alpha+\varepsilon)$-factor. Unfortunately, it is not always easy to show a streaming algorithm is monotone and/or subadditive. In some cases, this is not in fact, the case. On the other hand, Theorem 2 shows that we can achieve $(2\alpha+\varepsilon)$-approximation factor independent of the monotonicity or subadditivity of the streaming algorithm $\mathcal{ALG}$. We should mention that there is a similar result for constrained submodular maximization in the sliding-window model due to Chen, Nguyen, and Zhang [3].

Notation and terminology. In this paper, we assume we are given a weighted graph $G(V,E)$ with $n=|V|$ vertices and a weight function $w: E\to\mathbb{R}^+$ that assigns a non-negative weight $w(e)$ to each edge $e=(u,v)\in E$, where the ratio of the maximum edge weight to the

Table 1 Overview of results on the maximum-weight matching problem.

Model	Problem	Approximation	Adversarial Order	Reference
insertion-only	MCM	2	✓	[9]
		$1.5+\varepsilon$	Random Order	[1]
	MWM	$4+\varepsilon$	✓	[5]
		$2+\varepsilon$	✓	[18]
sliding-window	MCM	$3+\varepsilon$	✓	[4]
	MWM	9.027	✓	[4]
		$6+\varepsilon$	✓	[5]
		$3.5+\varepsilon$	✓	[this paper]

minimum edge weight is upper-bounded by $W = n^{O(1)}$. We assume that the graph is simple, that is, it does not have parallel edges. We denote by $N_G(v) = \{u \in V : \exists (u,v) \in E\}$ the set of neighbors of a vertex v in G. We drop the subscript G and write $N(v)$ when G is clear from the context. Similarly, we denote by $N_G(e)$ the set of edges that are neighbors of an edge $e = (u,v)$, that is, those edges such that one of their endpoints is in the $\{u,v\}$. A *matching* M of G is a set of pairwise non-adjacent edges, that is, a set where no two edges share a common vertex. The weight of the matching M is defined as $w(M) = \sum_{e \in M} w(e)$. A *maximum matching* of graph $G(V,E)$ is a matching of maximum weight. Throughout the paper, when we fix a maximum-weight matching of a graph G, we denote it by $M_{\text{opt}}(G)$, or simply M_{opt} when G is clear from the context. We say a matching M is an *α-approximate weighted matching*, for some $\alpha > 1$, when $w(M) \geqslant (1/\alpha) \cdot w(M_{\text{opt}})$. Finally, with a slight abuse of notation, we will often not distinguish between sets of edges and streams of edges. As an example, given a stream S of edges of an underlying graph $G(V,E)$, we do not distinguish between $G(V,E)$ and $G(V,S)$.

2 Maximum-weight matching in the insertion-only model

As mentioned in the introduction, in a breakthrough paper [18], Paz and Schwartzman showed that there exists an algorithm in the insertion-only model that computes a $(2+\varepsilon)$-approximate weighted matching of a graph $G(V,E)$. Their algorithm forms the basis of our $(3.5+\varepsilon)$-approximation algorithm in the sliding-window model. In this section, we explain Paz and Schwartzman's algorithm and we give various properties – some already proved by Ghaffari and Wajc [11], some new – that we later use.

Overview of algorithm. Before the stream starts, we initialize an empty stack. The streaming algorithm then processes the edges in the stream one by one. While doing so, it maintains a *potential* $\phi(v)$ for every vertex $v \in V$ (which is initialized to zero before the stream starts). For each arriving edge $e = (u,v)$, it is decided whether or not to push e on the stack based on its weight $w(e)$ and on the potential of its endpoints. In particular, if $w(e) \geqslant (1+\varepsilon) \cdot (\phi(u) + \phi(v))$, then e is added to the stack. When e is added to the stack, we assign a *reduced weight* $w'(e)$ to it, defined as $w'(e) := w(e) - (\phi(u) + \phi(v))$; from now on we refer to $w(e)$ as the *original weight* of e. If e is added to the stack, we add $w'(e)$ to the potential of its endpoints. It will be convenient to set $w'(e) := 0$ when e is not added to the stack.

After all edges have been handled in this manner, the matching is computed greedily: we pop edges from the stack and add them to the matching $M_{\text{alg}}(G)$ if they have no neighboring edge in $M_{\text{alg}}(G)$.

Algorithm 1 MWM-Streaming [18, 11].

Streaming:

while a new edge $e = (u, v)$ of the stream S is revealed **do**
  if $w(e) < (1 + \varepsilon) \cdot (\phi(u) + \phi(v))$ **then** $w'(e) \leftarrow 0$
  else
    $w'(e) \leftarrow w(e) - (\phi(u) + \phi(v))$ ▷ $w'(e)$ is the *reduced weight* of e
    $\phi(u) \leftarrow \phi(u) + w'(e)$; $\phi(v) \leftarrow \phi(v) + w'(e)$ ▷ update potentials
    $Stack.push(e)$

Postprocessing:

Let $M_{\text{alg}}(G) \leftarrow \emptyset$ be an empty matching set.
while $Stack \neq \emptyset$ **do**
  $e \leftarrow Stack.pop()$
  if $M_{\text{alg}}(G) \cap N(e) = \emptyset$ **then** $M_{\text{alg}}(G) \leftarrow M_{\text{alg}}(G) \cup \{e\}$.
return $M_{\text{alg}}(G)$.

Properties of the algorithm. Let t_e be the arrival time of the edge e. Let $P(e) := \{e' \in E : e' \in N(e) \cup \{e\}$ and $t_{e'} \leqslant t_e\}$ be the set of neighbor edges of e that arrive in the stream before e plus the edge e itself. Observe that if e is not added to the stack, then $w(e) < (1 + \varepsilon) \cdot (\phi(u) + \phi(v))$; and if this happens, the reduced weight must be $w'(e) = 0$.

The first lemma shows that the original weight of every edge in the graph is upper-bounded by $(1 + \varepsilon)$-factor of the sum of the potentials that are assigned to its end-points regardless of the fact that the edge is pushed onto the stack or not. The following lemma was proved by Ghaffari and Wajc [11, Observation 3.2].

▶ **Lemma 3** ([11]). *After the algorithm has finished we have* $w(e) \leqslant (1 + \varepsilon) \cdot (\phi(v) + \phi(u))$ *for each* $e = (u, v)$ *in* E.

The next lemma gives a relation between the original weight $w(e)$ of an edge e and the reduced weights $w'(e')$ of the edges $e' \in P(e)$. It is a special case of Lemma 3.4 of [11], where it was stated that $w(e) \geqslant \sum_{e' \in P(e)} w'(e')$. As follows from their proof, we can only have $w(e) > \sum_{e' \in P(e)} w'(e')$ when there are parallel edges. As we assume in this paper that G is a simple graph (i.e., does not contain any parallel edge) we state the result as follows.

▶ **Lemma 4** ([11]). *Suppose that the graph* $G(V, E)$ *is a simple graph. Let* $e \in E$ *be an edge that is added to the stack. Then* $w(e) = \sum_{e' \in P(e)} w'(e')$.

Lemma 4 is about edges that are added to the stack. The next lemma states a similar result for any edge e, regardless of whether it is added to the stack.

▶ **Lemma 5.** *For each* $e \in E$ *we have* $w(e) \leqslant (1 + \varepsilon) \sum_{e' \in P(e)} w'(e')$.

Proof. If $e = (u, v)$ is added to the stack, the inequality holds according to Lemma 4. Suppose that e is not added to the stack. We use $\phi_e^-(u)$ and $\phi_e^-(v)$ for the potential values just before the arrival of e. As e is not added to the stack, we have $w(e) < (1 + \varepsilon) \cdot (\phi_e^-(u) + \phi_e^-(v))$. Let e' be an edge in $P(e) \setminus \{e\}$. As G does not contain parallel edges, e' increases exactly one of $\phi(u)$ or $\phi(v)$ by $w'(e')$. (This also holds when $w'(e') = 0$.) Therefore, $\phi_e^-(u) + \phi_e^-(v) = \sum_{e' \in P(e) \setminus \{e\}} w'(e')$. Thus,

$$w(e) < (1 + \varepsilon) \cdot (\phi_e^-(v) + \phi_e^-(u)) = (1 + \varepsilon) \cdot \sum_{e' \in P(e) \setminus \{e\}} w'(e') \leqslant (1 + \varepsilon) \cdot \sum_{e' \in P(e)} w'(e') \ . \quad \blacktriangleleft$$

We also need Lemma 3.1 of [11], which states that the weight of the reported matching $M_{\text{alg}}(G)$ is lower-bounded by the sum of reduced weights of all edges of the graph G, which is in fact, one half of the sum of potentials that we place on vertices. The latter is a $(1+\varepsilon)$-approximation of the optimal weight $w(M_{\text{opt}}(G))$.

▶ **Lemma 6** ([11]). *Let $M_{\text{alg}}(G)$ be the matching reported by Algorithm MWM-Streaming. Then,*

$$w(M_{\text{alg}}(G)) \geqslant \sum_{e\in E} w'(e) = \frac{1}{2}\sum_{v\in V}\phi(v) \geqslant \frac{1}{2(1+\varepsilon)}\cdot w(M_{\text{opt}}(G)) \ .$$

Suppose we have two substreams B and BC, i.e., B is a prefix of BC. The next lemma shows that the reduced weight of an edge $e \in B$ is the same when the algorithm is run on B as when it is run on BC. It immediately follows from the fact that $w'(e)$ is set when e is processed, and it will not be changed afterwards.

▶ **Lemma 7.** *Let S be a stream of edges of an underlying simple weighted graph $G(V,E)$. Let B, C be two disjoint segments of the stream S. Let $e \in B$ be an arbitrary edge in B. Let $w'_B(e)$ and $w'_{BC}(e)$ be the reduced weights of e when we run Algorithm MWM-Streaming on the streams B and BC, respectively. Then, $w'_B(e) = w'_{BC}(e)$.*

Making MWM-Streaming monotone. Later, when we develop our sliding-window algorithm, we need Algorithm MWM-Streaming to be monotone, that is, we need that the weight of the reported matching for a stream BC is at least the weight of the reported matching for B. Unfortunately, this need not be the case, because in the reporting phase the edges are popped from the stack in reverse order of their arrival and added to the matching greedily.

However, we can simply make Algorithm MWM-Streaming monotone by maintaining the best solution over all prefixes encountered so far. More precisely, while we run the algorithm on the input stream S as usual, we maintain $M_{\text{mon}}(S) := \arg\max\{w(M_{\text{alg}}(T)) : T \text{ is a prefix of } S\}$. To this end, we just run the Postprocessing subroutine after each arrival on a copy of the current stack, and update $M_{\text{mon}}(S)$ if the computed solution is better. We denote by MWM-Monotone this monotone version of Algorithm MWM-Streaming.

Next, we state two useful properties of the monotone matching $M_{\text{mon}}(S)$.

▶ **Lemma 8.** *Let S be a stream of edges of an underlying simple weighted graph $G(V,E)$. Suppose that we have two disjoint substreams B and C of the stream S. Then,*

- ***Monotonicity Property 1:*** $w(M_{\text{alg}}(B)) \leqslant w(M_{\text{mon}}(B)) \leqslant w(M_{\text{opt}}(B))$.
- ***Monotonicity Property 2:*** $w(M_{\text{mon}}(B)) \leqslant w(M_{\text{mon}}(BC))$.

3 Maximum-weight matching in the sliding-window model

In this section, we develop our 3.5-approximation algorithm for the maximum-weight matching problem in the sliding-window model. As the first step, we next provide a generic framework for computing monotone bounded functions in the sliding-window model. Our framework is based on the *smooth-histogram techniques* developed by Braverman and Ostrovsky [2] for the sliding-window model. The smooth histogram technique was later used by Crouch, McGregor, and Stubbs [4] to develop algorithms for graph problems in the sliding-window model.

The idea behind this technique is that at any time t, we have a logarithmic number of insertion-only algorithms running in parallel, that each started at different times. As our answer at time t we then report, roughly speaking, the answer given by the "oldest"

still running algorithm; the technique ensures that this algorithm started shortly after time $t - L$, so that the answer is a good approximation of the actual answer. The next definition formalizes the properties we need to apply the technique. Suppose we want to approximate a function f in a sliding-window setting, and we have an algorithm $\mathcal{ALG}$ for the insertion-only setting. The idea is that if A, B are two substreams such that $f(ABC) \leqslant \alpha \cdot \mathcal{ALG}(BC)$ for any substream C, then we can α-approximate $f(ABC)$ by $\mathcal{ALG}(BC)$.

▶ **Definition 9** ((f, α, β)-lookahead algorithm). *Let $\beta \in (0, 1)$ and $\alpha \in \mathbb{R}^+$ be two parameters. Let f be a real-valued monotone function defined on subsets of a ground set $\mathcal{X}$. Let S be a stream of items of the set $\mathcal{X}$. Let $\mathcal{ALG}$ be a streaming algorithm. We say the streaming algorithm $\mathcal{ALG}$ is an* (f, α, β)-lookahead algorithm *if for any partitioning of S into three disjoint sub-streams A, B, and C with $\mathcal{ALG}(B) \geqslant (1 - \beta) \cdot \mathcal{ALG}(AB)$, we have $f(ABC) \leqslant \alpha \cdot \mathcal{ALG}(BC)$.*

The following theorem shows that if we have such a lookahead algorithm, we can develop a sliding-window algorithm for the function f. The proof of this theorem is based on the techniques developed in [4] and we give the full proof for the sake of completeness.

▶ **Theorem 10.** *Let $0 < \beta < 1$ and $\alpha \geqslant 1$ be two parameters. Let S be a stream of items from an underlying ground set $\mathcal{X}$. Let f be a monotone function defined on subsets of a ground set $\mathcal{X}$. Let $\sigma := f_{\max}/f_{\min^+}$, where $f_{\min^+} := \min\{f(X) : X$ is a substream of S and $f(X) > 0\}$ and $f_{\max} := \max\{f(X) : X$ is a substream of the input$\}$. Suppose there exists an (f, α, β)-lookahead algorithm that uses space s. Then there exists a sliding-window algorithm that maintains an α-approximation of f using $O(\beta^{-1} \cdot s \log \sigma)$ space.*

Proof. We first present our algorithm SlidingLookahead, which is based on the smooth histograms developed by Braverman and Ostrovsky [2] for the sliding-window model.

Algorithm 2 SlidingLookahead (Based on Smooth Histogram [2, 14]).

Initialization:

Let $k \leftarrow 0$ be the number of buckets.

Streaming:

while a new item e of the stream S is revealed at time t **do**
  Create an empty bucket B_{k+1}.
  Let $\mathcal{ALG}_{k+1}$ be an instance of $\mathcal{ALG}$ for the bucket B_{k+1}.
  for $i = 1, \ldots, k + 1$ **do**
    Feed e to the bucket B_i, and update its associated instance $\mathcal{ALG}_i$.
  for $i = 1, \ldots, k - 1$ **do**
    Let $j > i$ be the largest index for which $\mathcal{ALG}(B_j) \geqslant (1 - \beta) \cdot \mathcal{ALG}(B_i)$.
    Delete buckets B_r for $i < r < j$ and their associated instances $\mathcal{ALG}_r$.
  if $W \subseteq B_2$ **then** Delete bucket B_1 and its associated instance $\mathcal{ALG}_1$.
  Let k be number of remaining buckets.
  Renumber buckets and their associated instances.

Output:

if $W = B_1$ **then return** the solution of the $\mathcal{ALG}_1$, which equals $\mathcal{ALG}(B_1)$.
else return the solution of the $\mathcal{ALG}_2$, which equals $\mathcal{ALG}(B_2)$.

Let $W = [\max(t - L, 1), t]$ be the active window. With a slight abuse of notation, we will use W both for the time interval of the active window as well as for the set of items that arrive during this time interval. We consider k buckets $B_1, \cdots, B_k$ such that $B_1 \supseteq W \supsetneq B_2 \supsetneq \cdots \supsetneq B_k$. We later show that $k = O(\varepsilon^{-1} \log \sigma)$. For each bucket, we instantiate an instance of (f, α, β)-lookahead algorithm $\mathcal{ALG}$. Upon arrival of a new item e at a time t, we create a new bucket B_{k+1} and add e to all buckets $B_1, \cdots, B_{k+1}$. Next, if there exists a sequence of buckets whose values of $\mathcal{ALG}$ are too close, we keep the bucket in this sequence that has the lowest index and delete the rest of the buckets in this sequence.

Observe that the active window W is always sandwiched between the buckets B_1 and B_2. If at any time t, the active window W is covered by the bucket B_2 (i.e., $W \subseteq B_2$), we then delete B_1 and renumber the buckets accordingly. The output of smooth histogram is reported by the instance $\mathcal{ALG}$ of the bucket B_2.

Next, we prove the theorem which is based on the combination of the proofs of Lemma 3 and Theorem 4 in [4]. (We should mention that the proof in [4] is for $\alpha = 3 + \varepsilon$, but we prove it for any α.)

Let $\mathcal{ALG}$ be a (f, α, β)-lookahead algorithm using space s. We show that SlidingLookahead maintains an α-approximation of f, using $O(\beta^{-1} \cdot s \log \sigma)$ space. First, we prove the approximation ratio. Later, we show that $k = O(\beta^{-1} \log \sigma)$ at any time t.

Let $W = [\max(t-L, 1), t]$ be the active window and let $0 < \beta < 1$. Recall that the buckets $B_1, \cdots, B_k$ satisfy $B_1 \supseteq W \supsetneq B_2 \supsetneq \cdots \supsetneq B_k$. If $W = B_1$, we return $\mathcal{ALG}(B_1) = \mathcal{ALG}(W)$, which satisfies $f(W) \leqslant \alpha \cdot \mathcal{ALG}(B_1)$. Now, suppose that $W \neq B_1$. Since we report $\mathcal{ALG}(B_2)$, we must show that $f(W) \leqslant \alpha \cdot \mathcal{ALG}(B_2)$. To this end, let t^* be the first time that buckets B_1 and B_2 became adjacent; it is the time when buckets in between B_1 and B_2 were removed in step 8 of the streaming phase. We denote these buckets at time t^* by B_1^* and B_2^*. So, $\mathcal{ALG}(B_2^*) > (1 - \beta)\mathcal{ALG}(B_1^*)$. Moreover, $B_1 = B_1^* C$, $B_2 = B_2^* C$ for the stream C consisting of the items that arrived since time t^*. Now, according to the (f, α, β)-lookahead property of $\mathcal{ALG}$, we can conclude $f(B_1) \leqslant \alpha \cdot \mathcal{ALG}(B_2)$. As f is monotone, we have:

$$f(W) \leqslant f(B_1) \leqslant \alpha \cdot \mathcal{ALG}(B_2) \ .$$

Recall that k is the number of buckets. Next, we prove that at any time t, we have $k = O(\beta^{-1} \log \sigma)$. Recall that for each index $i \in [1..k-2]$, we have

$$\mathcal{ALG}(B_{i+2}) < (1 - \beta) \cdot \mathcal{ALG}(B_i).$$

Since $1/(1 - \beta) > 1 + \beta$, we have

$$\mathcal{ALG}(B_i) > \frac{1}{1 - \beta} \cdot \mathcal{ALG}(B_{i+2}) > (1 + \beta) \cdot \mathcal{ALG}(B_{i+2}).$$

Thus,

$$\mathcal{ALG}(B_1) \geqslant (1 + \beta)^{\lfloor k/2 \rfloor} \cdot \mathcal{ALG}(B_k) \ .$$

Recall that $\frac{f(B_1)}{f(B_k)} \leqslant \sigma$. This essentially means that $\mathcal{ALG}(B_1) \leqslant \sigma\alpha \cdot \mathcal{ALG}(B_k)$. Therefore, $k \leqslant O(\log_{1+\beta}(\sigma\alpha)) = O(\beta^{-1} \cdot \log \sigma)$. ◀

Our main result in this paper is that there exists a lookahead algorithm for the maximum weight matching problem. We state this result formally next.

▶ **Lemma 11.** *Let $G(V, E)$ be a graph with $n = |V|$ vertices and let $w : E \to \mathbb{R}^+$ be a function that assigns a non-negative weight $w(e)$ to each edge $e \in E$. Let $0 < \varepsilon \leqslant \frac{1}{10}$ and $0 < \beta \leqslant \frac{\varepsilon}{9}$ be two parameters. Let f be defined as the weight of a maximum weight matching of G. Then, Algorithm MWM-Monotone is a $(f, (3.5 + \varepsilon), \beta)$-lookahead algorithm. That is, for any partitioning of a stream S of edges of G into three disjoint sub-streams A, B, and C with $w(M_{\text{mon}}(B)) \geqslant (1 - \beta) \cdot w(M_{\text{mon}}(AB))$, we have $w(M_{\text{opt}}(ABC)) \leqslant (3.5 + \varepsilon) \cdot w(M_{\text{mon}}(BC))$.*

Thus, we can use the machinery developed in Theorem 10 to obtain a $(3.5+\varepsilon)$-approximate sliding-window algorithm for the maximum-weight matching.

We first define some notation and prove auxiliary tools to later prove Lemma 11.

Notation. Suppose we have a simple weighted graph $G(V, E)$, whose edges are revealed in a streaming fashion and let that stream be S. Suppose we partition the stream S into three disjoint consecutive substreams A, B, and C. Let $M_{\text{opt}}(ABC)$ be a fixed maximum-weight matching of G. Let $X \in \{A, B, C, AB, BC\}$.

- We denote by $M_{\text{opt}}(ABC) \cap X$ those edges of the maximum-weight matching $M_{\text{opt}}(ABC)$ that belong to the substream X.
- We denote by $M_{\text{alg}}(X)$ the reported matching of Algorithm MWM-Streaming if we invoke it on the input substream X.
- We let $\phi_X(v)$ be the potential that is assigned to a vertex $v \in V$ if we execute Algorithm MWM-Streaming on the input substream X.
- We let $w'_X(e)$ be the reduced weight that we assign to an edge $e \in E$ if we execute Algorithm MWM-Streaming on the input substream X.

A key concept in our analysis is the so-called *critical subgraph* of a graph G. We take advantage of this concept to show upper bounds for $w(M_{\text{opt}}(ABC) \cap AB)$ and $w(M_{\text{opt}}(ABC) \cap C)$ in Lemmas 13–16.

▶ **Definition 12** (Critical Subgraph). *Consider a graph G specified by a stream S of edges. Let A, B, C be disjoint substreams of S such that $S = ABC$. Then, the* critical subgraph *of G with respect to the matching $M_{\text{opt}}(ABC)$ and the substreams A, B, C is the subgraph $H = (V_H, E_H)$ such that*

- *$E_H := \{e \in B | e$ has two neighbors in $M_{\text{opt}}(ABC) \cap C\}$.*
- *$V_H := \{v \in V | \exists u \in V : (u, v) \in E_H\}$, i.e., V_H is the set of endpoints of the edges in E_H.*

Outline of the proof of Lemma 11. We show an upper-bound for $w(M_{\text{opt}}(ABC))$ based on $w(M_{\text{alg}}(AB))$ and $w(M_{\text{alg}}(BC))$. In particular, we show that

$$w(M_{\text{opt}}(ABC)) \leqslant 2(1 + \varepsilon)w(M_{\text{alg}}(AB)) + 2(1 + \varepsilon)w(M_{\text{alg}}(BC)) - \mathcal{Z} \ . \tag{1}$$

The slack term $\mathcal{Z}$ shows that there might be a double counting in the above inequality. Intuitively, this makes sense as the substream B is repeated in the first and the second terms.

However, for the moment, suppose this is not the case and $\mathcal{Z} = 0$. Assume that $w(M_{\text{alg}}(B)) \geqslant (1 - \beta) \cdot w(M_{\text{alg}}(AB))$, that is, the sets A and B (and our algorithm) are such that $M_{\text{alg}}(B)$ gives a good approximation of $M_{\text{alg}}(AB)$. Then, we have

$$\begin{aligned} w(M_{\text{opt}}(ABC)) &\leqslant 2 \cdot \frac{(1 + \varepsilon)}{(1 - \beta)} \cdot w(M_{\text{alg}}(B)) + 2(1 + \varepsilon) \cdot w(M_{\text{alg}}(BC)) \\ &\leqslant 2 \cdot \frac{(1 + \varepsilon)}{(1 - \beta)} \cdot w(M_{\text{mon}}(B)) + 2(1 + \varepsilon) \cdot w(M_{\text{mon}}(BC)) \\ &\leqslant 4 \cdot (1 + O(\varepsilon)) \cdot w(M_{\text{mon}}(BC)) \ , \end{aligned}$$

where for the second inequality we switch from Algorithm MWM-Streaming to its monotone version which is Algorithm MWM-Streaming-Monotone and the third inequality uses Monotonicity Property 2 of Lemma 8.

Observe that this result already improves upon the $(6+\varepsilon)$-approximation algorithm in the sliding-window model due to Crouch and Stubbs [5]. To obtain an even better bound, we show that $\mathcal{Z}$ is lower-bounded by a constant factor of the optimal matching of the substream B. In particular, we show that Inequality (1) holds for $\mathcal{Z} = \frac{1}{2(1+\varepsilon)} \cdot w(M_{\text{opt}}(B))$. Since $w(M_{\text{opt}}(B)) \geqslant w(M_{\text{mon}}(B))$, we then have

$$\begin{aligned} w(M_{\text{opt}}(ABC)) &\leqslant 2\frac{(1+\varepsilon)}{(1-\beta)} w(M_{\text{mon}}(B)) + 2(1+\varepsilon) w(M_{\text{mon}}(BC)) - \frac{1}{2(1+\varepsilon)} w(M_{\text{mon}}(B)) \\ &\leqslant (3.5 + O(\varepsilon)) w(M_{\text{mon}}(BC)) \ . \end{aligned}$$

The complete proof of Lemma 11. We find the bound on $w(M_{\text{opt}}(ABC))$ that we claimed in Equation 1 in three steps. First in Lemma 13, we find an upper-bound on the contribution of the substream AB towards $M_{\text{opt}}(ABC)$. Next, we upper-bound the weight of edges of the optimal matching $M_{\text{opt}}(ABC)$ that are in the substream C. This is done in Lemma 14. Finally, in Lemma 15 we obtain a lower-bound for the slack term $\mathcal{Z}$ of Equation 1.

▶ **Lemma 13.** $w(M_{\text{opt}}(ABC) \cap AB) \leqslant 2(1+\varepsilon) w(M_{\text{alg}}(AB)) - (1+\varepsilon) \cdot \sum_{v \in V_H} \phi_{AB}(v)$.

Proof. By definition, $w(M_{\text{opt}}(ABC) \cap AB) = \sum_{e^* \in M_{\text{opt}}(ABC) \cap AB} w(e^*)$. Let us consider an arbitrary edge $e^* = (u,v) \in M_{\text{opt}}(ABC) \cap AB$. Using Lemma 3, we have $w(e^*) \leqslant (1+\varepsilon) \cdot (\phi_{AB}(u) + \phi_{AB}(v))$. Observe that the vertices u and v cannot be in V_H. Thus, we obtain

$$w(M_{\text{opt}}(ABC) \cap AB) \leqslant (1+\varepsilon) \cdot \sum_{v \in V \setminus V_H} \phi_{AB}(v) = (1+\varepsilon) \cdot \sum_{v \in V} \phi_{AB}(v) - (1+\varepsilon) \cdot \sum_{v \in V_H} \phi_{AB}(v) \ .$$

Now we use Lemma 6 that shows $\sum_{v \in V} \phi_{AB}(v) \leqslant 2w(M_{\text{alg}}(AB))$. Hence,

$$w(M_{\text{opt}}(ABC) \cap AB) \leqslant 2(1+\varepsilon) \cdot w(M_{\text{alg}}(AB)) - (1+\varepsilon) \cdot \sum_{v \in V_H} \phi_{AB}(v) \ . \qquad \blacktriangleleft$$

The next lemma shows that the contribution of optimal edges that are in the substream C is upper-bounded by twice the weight of the reported matching of the substream BC minus the reduced weight of edges of B that are not in the critical subgraph H.

▶ **Lemma 14.** $w(M_{\text{opt}}(ABC) \cap C) \leqslant 2(1+\varepsilon) \cdot w(M_{\text{alg}}(BC)) - (1+\varepsilon) \cdot \sum_{e \in B \setminus E_H} w'_B(e)$.

Proof. Recall that for every edge $e \in E$, $w'_{BC}(e)$ is the reduced weight of e if we run Algorithm MWM-Streaming on the input stream BC, where we define $w'_{BC}(e) = 0$ if $e \notin BC$. To prove the lemma, we will show that

$$w(M_{\text{opt}}(ABC) \cap C) \leqslant 2(1+\varepsilon) \cdot \sum_{e \in BC} w'_{BC}(e) - (1+\varepsilon) \cdot \sum_{e \in B \setminus E_H} w'_{BC}(e) \ . \tag{2}$$

Suppose for now that Inequality (2) is correct. According to Lemma 7, for every $e \in B$, we have $w'_{BC}(e) = w'_B(e)$. Thus, $\sum_{e \in B \setminus E_H} w'_{BC}(e) = \sum_{e \in B \setminus E_H} w'_B(v)$.

Next, we use Lemma 6, where we replace the graph G in that lemma with the subgraph $G'(V, BC)$ to show that $\sum_{e \in BC} w'_{BC}(e) \leqslant w(M_{\text{alg}}(BC))$.

Putting everything together, we prove the statement of this lemma as follows:

$$\begin{aligned} w(M_{\text{opt}}(ABC) \cap C) &\leqslant 2(1+\varepsilon) \cdot \sum_{e \in BC} w'_{BC}(e) - (1+\varepsilon) \cdot \sum_{e \in B \setminus E_H} w'_{BC}(e) \\ &\leqslant 2(1+\varepsilon) \cdot w(M_{\text{alg}}(BC)) - (1+\varepsilon) \cdot \sum_{e \in B \setminus E_H} w'_{BC}(e) \\ &= 2(1+\varepsilon) \cdot w(M_{\text{alg}}(BC)) - (1+\varepsilon) \cdot \sum_{e \in B \setminus E_H} w'_{B}(e) \ . \end{aligned}$$

It remains to prove Inequality (2). Let $e^* \in M_{\text{opt}}(ABC) \cap C$. We use Lemma 5, where we replace the graph G in that lemma with the subgraph $G'(V, BC)$. Then, Lemma 5 shows that $w(e^*) \leqslant (1+\varepsilon) \cdot \sum_{e' \in P_{BC}(e^*)} w'_{BC}(e')$, where $P_{BC}(e) := \{e' \in BC : e' \in N_{G'}(e) \cup \{e\}$ and $t_{e'} \leqslant t_e\}$. (Here $t_{e'}$ and t_e are the arrival time of edges e and e' in the substream BC.)

Hence,

$$w(M_{\text{opt}}(ABC) \cap C) = \sum_{e^* \in M_{\text{opt}}(ABC) \cap C} w(e^*) \leqslant (1+\varepsilon) \sum_{e^* \in M_{\text{opt}}(ABC) \cap C} \sum_{e' \in P_{BC}(e^*)} w'_{BC}(e')$$

Consider the double summation $\sum_{e^* \in M_{\text{opt}}(ABC) \cap C} \sum_{e' \in P_{BC}(e^*)} w'_{BC}(e')$. Every edge $e' \in BC$ appears in $P_{BC}(e^*)$ for at most two edges e^*, because each endpoint of e' is incident to at most one edge from $M_{\text{opt}}(ABC)$. Hence, the double summation can be bounded by $2 \cdot \sum_{e \in BC} w'_{BC}(e)$. If, however, e' appears twice then $e' \in E_H$, that is, e' is part of the critical subgraph H. This means that the edges in $B \setminus E_H$ appear at most once in the sum. Therefore,

$$w(M_{\text{opt}}(ABC) \cap C) \ \leqslant \ 2(1+\varepsilon) \cdot \sum_{e \in BC} w'_{BC}(e) - (1+\varepsilon) \cdot \sum_{e \in B \setminus E_H} w'_{BC}(e),$$

which proves Inequality (2) and finishes the proof of the lemma. ◀

▶ **Lemma 15.** *Suppose we execute Algorithm MWM-Streaming on the substream AB. Let $\phi_{AB}(v)$ be the potential assigned to a vertex $v \in V_H$ at the end of this algorithm. Then, $(1+\varepsilon) \cdot \sum_{v \in V_H} \phi_{AB}(v) \geqslant \sum_{e \in E_H} w'_B(e)$.*

Proof. Observe that $E_H \subseteq B \subseteq AB$. Let us consider an arbitrary edge $e = (u, v)$ in E_H. By applying Lemma 3 to the graph $G'(V, AB)$ we obtain $w(e) \leqslant (1+\varepsilon)(\phi_{AB}(u) + \phi_{AB}(v))$. Fix a maximum-weight matching $M_{\text{opt}}(H)$ of the critical subgraph $H(V_H, E_H)$. Then,

$$\begin{aligned} w(M_{\text{opt}}(H)) &= \sum_{e=(u,v) \in M_{\text{opt}}(H)} w(e) \leqslant \sum_{(u,v) \in M_{\text{opt}}(H)} (1+\varepsilon)(\phi_{AB}(u) + \phi_{AB}(v)) \\ &\leqslant (1+\varepsilon) \sum_{u \in V_H} \phi_{AB}(u) \ , \end{aligned}$$

where the last inequality is due to the fact that the degree of any vertex $u \in V_H$ in the matching $M_{\text{opt}}(H)$ is at most one. We next find a lower bound for $w(M_{\text{opt}}(H))$, which in turns yields a lower bound for $(1+\varepsilon) \sum_{u \in V_H} \phi_{AB}(v)$.

Now, suppose we execute Algorithm MWM-Streaming on the stream B. However, during the postprocessing phase, once we pop an edge e from the stack, we add it to the matching M_{alg} if both end-points of e are free and $e \in E_H$. Thus, the reported matching M_{alg} of the substream B is influenced by the edge set E_H. We denote by $M_{\text{alg}}(B|E_H)$ the

reported matching of B that is influenced by E_H. Observe that $M_{\text{alg}}(B|E_H)$ is a matching of the critical subgraph $H(V_H, E_H)$. Therefore, $w(M_{\text{alg}}(B|E_H)) \leqslant w(M_{\text{opt}}(H))$. We next find a lower bound for $w(M_{\text{alg}}(B|E_H))$.

Consider the graph $G'(V, B)$. Let us fix an arbitrary edge $e \in B$. Recall that $P_B(e) = \{e' \in B : e' \in N_{G'}(e) \cup \{e\}$ and $t_{e'} \leqslant t_e\}$. (Here $t_{e'}$ and t_e are the arrival time of edges e and e' in the substream B.) Observe that if $e \in M_{\text{alg}}(B|E_H)$, then e was added to the stack at some point. Using Lemma 4, we then have $w(e) = \sum_{e' \in P_B(e)} w'_B(e')$. Therefore,

$$\sum_{e \in M_{\text{alg}}(B|E_H)} w(e) = \sum_{e \in M_{\text{alg}}(B|E_H)} \sum_{e' \in P_B(e)} w'_B(e').$$

Now, let us consider an arbitrary edge $e' \in E_H$. For e' we have three cases. The first case is that e' is not in the stack, which implies $w'_B(e') = 0$. The second case is that $e' \in M_{\text{alg}}(B|E_H)$, so e' contributes to the sum $\sum_{e \in M_{\text{alg}}(B|E_H)} w(e)$. The third case is that e' has a neighbor $e \in M_{\text{alg}}(B|E_H)$, so that $e' \in P_B(e)$. Considering all the three cases we conclude

$$\sum_{e \in M_{\text{alg}}(B|E_H)} w(e) = \sum_{e \in M_{\text{alg}}(B|E_H)} \sum_{e' \in P_B(e)} w'_B(e') \geqslant \sum_{e \in E_H} w'_B(e).$$

Putting everything together, we obtain the statement of this lemma as follows:

$$(1+\varepsilon) \cdot \sum_{v \in V_H} \phi_{AB}(v) \geqslant w(M_{\text{opt}}(H)) \geqslant w(M_{\text{alg}}(B|E_H)) \geqslant \sum_{e \in E_H} w'_B(e) \ . \qquad \blacktriangleleft$$

We are now finally ready to prove Inequality (1) on page 9, with $\mathcal{Z} = \frac{1}{2(1+\varepsilon)} w(M_{\text{opt}}(B))$.

▶ **Lemma 16.** *Let $S = ABC$ be a stream of edges of an underlying weighted graph $G(V, E)$. Then, there exists the following upper bound for the weight of $M_{\text{opt}}(ABC)$:*

$$w(M_{\text{opt}}(ABC)) \leqslant 2(1+\varepsilon) w(M_{\text{alg}}(AB)) + 2(1+\varepsilon) w(M_{\text{alg}}(BC)) - \frac{1}{2(1+\varepsilon)} w(M_{\text{opt}}(B)) \ .$$

Proof. First of all, observe that we can decompose the edges of the optimal matching $M_{\text{opt}}(ABC)$ into the subset of edges that are in the substreams AB and C. Thus, we have

$$w(M_{\text{opt}}(ABC)) = w(M_{\text{opt}}(ABC) \cap AB) + w(M_{\text{opt}}(ABC) \cap C) \ .$$

Using Lemma 14 and Lemma 13, we then obtain

$$\begin{aligned} w(M_{\text{opt}}(ABC)) \leqslant\ & 2(1+\varepsilon) \cdot w(M_{\text{alg}}(AB)) - (1+\varepsilon) \cdot \sum_{v \in V_H} \phi_{AB}(v) \\ & + 2(1+\varepsilon) \cdot w(M_{\text{alg}}(BC)) - (1+\varepsilon) \cdot \sum_{e \in B \setminus E_H} w'_B(e) \ . \end{aligned}$$

Next, we replace the negative term on the sum of the potentials for vertices in V_H with its lower-bound as in Lemma 15 .

$$\begin{aligned} & w(M_{\text{opt}}(ABC)) \\ & \leqslant 2(1+\varepsilon) \cdot w(M_{\text{alg}}(AB)) - \sum_{e \in E_H} w'_B(e) + 2(1+\varepsilon) \cdot w(M_{\text{alg}}(BC)) - (1+\varepsilon) \sum_{e \in B \setminus E_H} w'_B(e) \\ & = 2(1+\varepsilon) \cdot w(M_{\text{alg}}(AB)) + 2(1+\varepsilon) w(M_{\text{alg}}(BC)) - \sum_{e \in B} w'_B(e) \ . \end{aligned}$$

Applying Lemma 6 to the subgraph $G'(V,B)$ gives $\sum_{e\in B} w'_B(e) \geqslant \frac{1}{2(1+\varepsilon)} \cdot w(M_{\text{opt}}(G'(V,B)))$. Hence,

$$w(M_{\text{opt}}(ABC)) \leqslant 2(1+\varepsilon)\cdot w(M_{\text{alg}}(AB))+2(1+\varepsilon)\cdot w(M_{\text{alg}}(BC))-\frac{1}{2(1+\varepsilon)}\cdot w(M_{\text{opt}}(B)) \ .\blacktriangleleft$$

With the help of Lemma 16 we can now prove Lemma 11.

Proof of Lemma 11. First of all using Lemma 16, we have the following upper-bound for the weight of $M_{\text{opt}}(ABC)$:

$$w(M_{\text{opt}}(ABC)) \leqslant 2(1+\varepsilon)w(M_{\text{alg}}(AB)) + 2(1+\varepsilon)w(M_{\text{alg}}(BC)) - \frac{1}{2(1+\varepsilon)}w(M_{\text{opt}}(B)) \ .$$

Now, we switch from Algorithm MWM-Streaming to its monotone version, which is Algorithm MWM-Monotone. Monotonicity Property 1 of Lemma 8 states that for the substream B we have $w(M_{\text{alg}}(B)) \leqslant w(M_{\text{mon}}(B)) \leqslant w(M_{\text{opt}}(B))$. The same bounds also hold for the substream AB and BC. Therefore,

$$\begin{aligned} w(M_{\text{opt}}(ABC)) &\leqslant 2(1+\varepsilon)w(M_{\text{mon}}(AB)) + 2(1+\varepsilon)w(M_{\text{mon}}(BC)) - \frac{1}{2(1+\varepsilon)}w(M_{\text{mon}}(B)) \\ &\leqslant \frac{2(1+\varepsilon)}{1-\beta}\cdot w(M_{\text{mon}}(B)) + 2(1+\varepsilon)w(M_{\text{mon}}(BC)) - \frac{1}{2(1+\varepsilon)}w(M_{\text{mon}}(B)) \\ &\leqslant 1.5(1+3\varepsilon)\cdot w(M_{\text{mon}}(B)) + 2(1+\varepsilon)w(M_{\text{mon}}(BC)) \ , \end{aligned}$$

where for the second inequality we use the assumption of this lemma, which is

$$w(M_{\text{mon}}(B)) \geqslant (1-\beta)\cdot w(M_{\text{mon}}(AB)) \ .$$

From Monotonicity Property 2 of Lemma 8, which is $w(M_{\text{mon}}(B)) \leqslant w(M_{\text{mon}}(BC))$, we now for $\varepsilon \leqslant \frac{1}{10}$ conclude

$$\begin{aligned} w(M_{\text{opt}}(ABC)) &\leqslant 1.5(1+3\varepsilon)w(M_{\text{mon}}(B)) + 2(1+\varepsilon)w(M_{\text{mon}}(BC)) \\ &\leqslant (3.5+\varepsilon)w(M_{\text{mon}}(BC)) \ . \end{aligned}$$ ◀

4 Subadditive functions in the sliding-window model

In this section, we prove Theorem 2 and then we show that using this theorem, we can improve the approximation factor of quite a few submodular matching problems in the sliding-window model. We first explain the notations that we use in this section.

Let f be a function defined on streams, which is subadditive, non-negative, and monotone. Let $\sigma := f_{\max}/f_{\min^+}$, where

$$f_{\min^+} := \min\{f(X) : X \text{ is a substream of the input and } f(X) > 0\} \ ,$$

and

$$f_{\max} := \max\{f(X) : X \text{ is a substream of the input}\}$$

Suppose we are given a streaming algorithm $\mathcal{ALG}$ that α-approximates f using space s. Very recently Krauthgamer and Reitblat [14] showed that we can use the streaming algorithm $\mathcal{ALG}$ to develop a sliding-window algorithm that $(2\alpha^2+\varepsilon)$-approximates f using space $O(\varepsilon^{-1}s\cdot\log\sigma)$. They also showed that if the streaming algorithm $\mathcal{ALG}$ is monotone and subadditive, we can reduce the approximation factor of the sliding-window algorithm down to $(2\alpha+\varepsilon)$-factor. Unfortunately, in some cases, although f is subadditive, $\mathcal{ALG}$ is not

subadditive, or it is not easy to show that $\mathcal{ALG}$ is also subadditive. So, the $(2\alpha+\varepsilon)$-factor cannot always be a result of [14]. Nevertheless, Theorem 2 shows that we can obtain a $(2\alpha+\varepsilon)$-approximation algorithm in the sliding-window model, independent of the monotonicity or subadditivity of the streaming algorithm $\mathcal{ALG}$. Our main result in this paper is that there exists a lookahead algorithm for the maximum weight matching problem. We state this result formally next.

Next, we prove Theorem 2. To this end, in Lemma 17, we show how to transform a non-monotone algorithm $\mathcal{ALG}$ into a monotone algorithm $\mathcal{ALG}_{\text{mon}}$. Later, in Lemma 18 we show that a monotone algorithm $\mathcal{ALG}_{\text{mon}}$ that α-approximates a subadditive monotone function f is in fact, a $(f, 2\alpha+\varepsilon(\beta), \beta)$-lookahead algorithm. Thus, we can use the machinery of Theorem 10 to develop $(2\alpha+\varepsilon)$-approximation sliding-window algorithms for f using the streaming algorithm $\mathcal{ALG}_{\text{mon}}$.

▶ **Lemma 17.** *Let f be a non-negative and monotone function defined on streams. Let $\mathcal{ALG}$ be a streaming algorithm that α-approximates f using space s, where $\alpha \geqslant 1$. Then, there is a monotone algorithm $\mathcal{ALG}_{\text{mon}}$ that α-approximates f using $O(s)$ space.*

Proof. Let S be a stream of items of the domain $\mathcal{X}$ of the subadditive f. In order to make the algorithm $\mathcal{ALG}$ monotone, we store the maximum f of all prefixes of the stream S. Formally, we define $\mathcal{ALG}_{\text{mon}}$ as follow: $\mathcal{ALG}_{\text{mon}}(S) := max\{\mathcal{ALG}(T) : T \text{ is a prefix of } S\}$.

To prove the approximation factor, we show that $\frac{1}{\alpha} \cdot f(S) \leqslant \mathcal{ALG}_{\text{mon}}(S) \leqslant f(S)$. Observe that $\mathcal{ALG}$ returns a α-approximation of $f(S)$. Thus, $\frac{1}{\alpha} \cdot f(S) \leqslant \mathcal{ALG}(S) \leqslant \mathcal{ALG}_{\text{mon}}(S)$.

Now, it is enough to show that $\mathcal{ALG}_{\text{mon}}(S) \leqslant f(S)$. Let T^* be the prefix of S for which $\mathcal{ALG}$ returns the maximum value among all prefixes of S. That is, $\mathcal{ALG}_{\text{mon}}(S) = \mathcal{ALG}(T^*)$. Since f is monotone $f(T^*) \leqslant f(S)$, we then have

$$\mathcal{ALG}_{\text{mon}}(S) = \mathcal{ALG}(T^*) \leqslant f(T^*) \leqslant f(S) \ . \qquad \blacktriangleleft$$

▶ **Lemma 18.** *Let $0 < \varepsilon \leqslant 1/2$, $\alpha \geqslant 1$ and $0 < \beta \leqslant \varepsilon/2\alpha$ be three parameters. Let f be a subadditive, non-negative, and monotone function defined on streams. Suppose we have a monotone streaming algorithm $\mathcal{ALG}_{\text{mon}}$ that α-approximates f. Then, Algorithm $\mathcal{ALG}_{\text{mon}}$ is a $(f, (2\alpha+\varepsilon), \beta)$-lookahead algorithm. That is, for any partitioning of a stream S of items of the domain $\mathcal{X}$ of f into three disjoint sub-streams A, B, and C with $\mathcal{ALG}_{\text{mon}}(B) \geqslant (1-\beta) \cdot \mathcal{ALG}_{\text{mon}}(AB)$, we have*

$$f(ABC) \leqslant (2\alpha+\varepsilon) \cdot \mathcal{ALG}_{\text{mon}}(BC) \ .$$

Proof. First of all since f is subadditive, we have $f(ABC) \leqslant f(A)+f(BC)$. Also, the function f is monotone, which means that $f(A) \leqslant f(AB)$. Therefore, $f(ABC) \leqslant f(AB) + f(BC)$.

Since $\mathcal{ALG}_{\text{mon}}$ is an α-approximate algorithm for f, we then have

$$f(ABC) \leqslant \alpha \cdot \mathcal{ALG}_{\text{mon}}(AB) + \alpha \cdot \mathcal{ALG}_{\text{mon}}(BC) \ .$$

We assume that $\mathcal{ALG}_{\text{mon}}(B) \geqslant (1-\beta) \cdot \mathcal{ALG}_{\text{mon}}(AB)$. Using which, we have

$$f(ABC) \leqslant \frac{\alpha}{(1-\beta)} \cdot \mathcal{ALG}_{\text{mon}}(B) + \alpha \cdot \mathcal{ALG}_{\text{mon}}(BC) \ .$$

Since $\mathcal{ALG}_{\text{mon}}$ is a monotone algorithm, we have $\mathcal{ALG}_{\text{mon}}(B) \leqslant \mathcal{ALG}_{\text{mon}}(BC)$. Thus,

$$\begin{aligned} f(ABC) &\leqslant \frac{\alpha}{(1-\beta)} \cdot \mathcal{ALG}_{\text{mon}}(BC) + \alpha \cdot \mathcal{ALG}_{\text{mon}}(BC) \\ &\leqslant \alpha(1+2\beta) \cdot \mathcal{ALG}_{\text{mon}}(BC) + \alpha \cdot \mathcal{ALG}_{\text{mon}}(BC) \\ &\leqslant (2\alpha+\varepsilon) \cdot \mathcal{ALG}_{\text{mon}}(BC) \ , \end{aligned}$$

by setting $0 < \beta \leqslant \frac{\varepsilon}{2\alpha}$ and using the Maclaurin series $\frac{1}{1-\varepsilon} = 1 + \varepsilon + \varepsilon^2 + \varepsilon^3 + \cdots \leq 1 + 2\varepsilon$ for $\varepsilon \leq 1/2$. $\blacktriangleleft$

Having Theorem 2 in hand, we can improve quite a few sliding-window algorithms that have $(2\alpha^2+\varepsilon)$-approximation factor. As an example, if we have α-approximation streaming algorithms for the maximum submodular matching, maximum submodular b-matching, and in general, maximum submodular rank p hypergraph b-matching, we can then use Theorem 2 to develop sliding-window algorithms for all these problems with the approximation factor of $(2\alpha+\varepsilon)$ which significantly improves upon the best known approximation factor of $(2\alpha^2+\varepsilon)$ of these problems due to Krauthgamer and Reitblat [14].

As an example, we consider next the maximum submodular matching, which is defined as follows. We are given a simple graph $G(V,E)$. A non-negative submodular function $w : 2^E \to \mathbb{R}^{\geqslant 0}$ is defined on subsets of edges. The goal is to find a matching M_{opt}, whose submodular function $w(M_{\text{opt}})$ is maximum. We next show that the maximum submodular matching is indeed a subadditive function.

▶ **Lemma 19.** *Let w be a non-negative submodular function defined on subsets of edges of a simple graph $G(V,E)$. Then, the maximum submodular matching of $G(V,E)$ is subadditive.*

Proof. Let us consider an arbitrary stream S of edges of the underlying graph $G(V,E)$. Suppose we partition S into two disjoint consecutive substreams A and B. Since w is non-negative, we then have $w(M_{\text{opt}}(AB)) \leqslant w(M_{\text{opt}}(AB)) + w(\emptyset)$.

Next, we use the property of submodular functions that for every two sets X, Y we have $w(X \cup Y) + w(X \cap Y) \leqslant w(X) + w(Y)$. Let $X = M_{\text{opt}}(AB) \cap A$ and $Y = M_{\text{opt}}(AB) \cap B$. Since A and B are disjoint substreams, we then have $X \cup Y = M_{\text{opt}}(AB)$ and $w(X \cap Y) = \emptyset$. Therefore, we have

$$w(M_{\text{opt}}(AB)) \leqslant w(M_{\text{opt}}(AB) \cap A) + w(M_{\text{opt}}(AB) \cap B) \ .$$

Finally, since $M_{\text{opt}}(AB) \cap A$ and $M_{\text{opt}}(AB) \cap B$ are valid matchings for A and B, respectively, then $w(M_{\text{opt}}(AB) \cap A) \leqslant w(M_{\text{opt}}(A))$ and $w(M_{\text{opt}}(AB) \cap B) \leqslant w(M_{\text{opt}}(B))$. Thus, we have

$$w(M_{\text{opt}}(AB)) \leqslant w(M_{\text{opt}}(A)) + w(M_{\text{opt}}(B)) \ ,$$

as we need. ◀

For the maximum monotone submodular matching problem, Levin and Wajc [15] develop a 5.828-approximation streaming algorithm. If we plug this streaming algorithm into the generic machinery of Theorem 2, we obtain 11.656-approximation sliding-window algorithm for this problem. This significantly improves upon the 67.93-approximation factor that the result of [14] achieves.

On the other hand, for the maximum non-monotone submodular matching problem, there exists a 7.464-approximation streaming algorithm due to Levin and Wajc [15]. We use this streaming algorithm for the the generic machinery of Theorem 2 and obtain 14.9-approximation sliding-window algorithm for this problem improving the 111.4-approximation factor that the result of [14] achieves.

References

1 Aaron Bernstein. Improved bounds for matching in random-order streams. In *Proceedings 47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*, volume 168 of *LIPIcs*, pages 12:1–12:13, 2020. doi:10.4230/LIPIcs.ICALP.2020.12.

2 Vladimir Braverman and Rafail Ostrovsky. Smooth histograms for sliding windows. In *Proceedings 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007)*, pages 283–293, 2007. doi:10.1109/FOCS.2007.55.

3 Jiecao Chen, Huy L. Nguyen, and Qin Zhang. Submodular maximization over sliding windows. *CoRR*, abs/1611.00129, 2016. arXiv:1611.00129.

4 Michael S. Crouch, Andrew McGregor, and Daniel M. Stubbs. Dynamic graphs in the sliding-window model. In *Proceedings 21st Annual European Symposium on Algorithms (ESA 2013)*, volume 8125 of *Lecture Notes in Computer Science*, pages 337–348. Springer, 2013. doi:10.1007/978-3-642-40450-4_29.

5 Michael S. Crouch and Daniel M. Stubbs. Improved streaming algorithms for weighted matching, via unweighted matching. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2014*, volume 28 of *LIPIcs*, pages 96–104, 2014. doi:10.4230/LIPIcs.APPROX-RANDOM.2014.96.

6 Mayur Datar, Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Maintaining stream statistics over sliding windows. *SIAM J. Comput.*, 31(6):1794–1813, 2002. doi:10.1137/S0097539701398363.

7 Hossein Esfandiari, MohammadTaghi Hajiaghayi, and Morteza Monemizadeh. Finding large matchings in semi-streaming. In *IEEE International Conference on Data Mining Workshops, ICDM Workshops 2016*, pages 608–614, 2016. doi:10.1109/ICDMW.2016.0092.

8 Alireza Farhadi, Mohammad Taghi Hajiaghayi, Tung Mai, Anup Rao, and Ryan A. Rossi. Approximate maximum matching in random streams. In *Proceedings 31st ACM-SIAM Symposium on Discrete Algorithms (SODA 2020)*, pages 1773–1785, 2020. doi:10.1137/1.9781611975994.108.

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

10 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 2019*, pages 491–500. ACM, 2019. doi:10.1145/3293611.3331603.

11 Mohsen Ghaffari and David Wajc. Simplified and space-optimal semi-streaming (2+epsilon)-approximate matching. In *Proceedings 2nd Symposium on Simplicity in Algorithms (SOSA 2019)*, volume 69 of *OASICS*, pages 13:1–13:8, 2019. doi:10.4230/OASIcs.SOSA.2019.13.

12 Michael Kapralov. Better bounds for matchings in the streaming model. In Sanjeev Khanna, editor, *Proceedings 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2013)*, pages 1679–1697, 2013. doi:10.1137/1.9781611973105.121.

13 Christian Konrad, Frédéric Magniez, and Claire Mathieu. Maximum matching in semi-streaming with few passes. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 15th International Workshop, APPROX 2012, and 16th International Workshop, RANDOM 2012*, volume 7408 of *Lecture Notes in Computer Science*, pages 231–242, 2012. doi:10.1007/978-3-642-32512-0_20.

14 Robert Krauthgamer and David Reitblat. Almost-smooth histograms and sliding-window graph algorithms. *CoRR*, abs/1904.07957, 2019. arXiv:1904.07957.

15 Roie Levin and David Wajc. Streaming submodular matching meets the primal-dual method. *CoRR*, abs/2008.10062, 2020. arXiv:2008.10062.

16 L. Lovasz and M.D. Plummer. *Matching Theory*. North-Holland, 1986.

17 Silvio Micali and Vijay V. Vazirani. An $O(\sqrt{|V||E|})$ algorithm for finding maximum matching in general graphs. In *21st Annual Symposium on Foundations of Computer Science (FOCS 1980)*, pages 17–27, 1980. doi:10.1109/SFCS.1980.12.

18 Ami Paz and Gregory Schwartzman. A $(2+\epsilon)$-approximation for maximum weight matching in the semi-streaming model. In Philip N. Klein, editor, *Proceedings 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2017)*, pages 2153–2161, 2017. doi:10.1137/1.9781611974782.140.