Semi-Streaming Algorithms for Submodular Function Maximization under b-Matching Constraint

Chien-Chung Huang[1] François Sellier*[2]

[1]CNRS, DI ENS, École normale supérieure, Université PSL, France

[2]École polytechnique, Institut Polytechnique de Paris, France

August 2021

Some Definitions

A b-matching on a multi-graph $G = (V, E)$ (with no self-loop) is defined as follow:

- for each vertex $v \in V$, a *capacity* $b_v \in \mathbb{Z}_+$ is given
- $M \subseteq E$ is a *b-matching* if M has at most b_v edges incident to v, for all $v \in V$ (through the presentation, $\delta(v)$ will denote the set of edges incident to v).

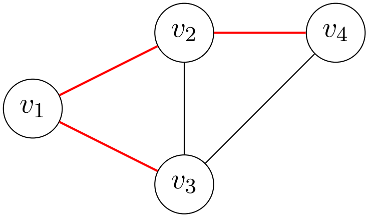


Figure: A b-matching for $b_{v_1} = b_{v_2} = 2$ and $b_{v_3} = b_{v_4} = 1$

Some Definitions

In the maximum weight b-matching problem:

- a weight function $w : E \to \mathbb{R}_+$ on the edges is given
- we have to find the b-matching M having the largest sum the weights $w(M) = \sum_{e \in M} w(e)$

Some Definitions

In the submodular maximum b-matching problem:

- a non-negative *submodular function* $f : 2^E \to \mathbb{R}_+$ is given, *i.e.* a set function satisfying:

$$\forall X \subseteq Y \subsetneq E, \forall e \in E \backslash Y, f(X \cup \{e\}) - f(X) \geq f(Y \cup \{e\}) - f(Y),$$

  additionally we sometimes suppose that f is *monotone*, meaning that $\forall X \subseteq Y \subseteq E, f(X) \leq f(Y)$
- we want to find a b-matching M maximizing $f(M)$

Some Definitions

Other possible constraints:

- k-uniform hypergraph with b-matching constraint
- add a matroid constraint so that a b-matching M also has to be independent in that matroid

Some Definitions

$\mathcal{M} = (E, \mathcal{I})$ is a *matroid* if these conditions hold:

1. $\emptyset \in \mathcal{I}$,
2. if $X \subseteq Y \in \mathcal{I}$, then $X \in \mathcal{I}$,
3. if $X, Y \in \mathcal{I}, |Y| > |X|$, there exists an element $e \in Y \setminus X$ so that $X \cup \{e\} \in \mathcal{I}$,

the sets in $\mathcal{I}$ are the *independent sets* and the *rank* $r_{\mathcal{M}}$ of the matroid $\mathcal{M}$ is defined as $\max_{X \in \mathcal{I}} |X|$.

Some Definitions

In the *semi-streaming* model:

- edges of E arrive over time
- we have access only to a limited memory, ideally proportional to the output size
- as a result, some edges may have to be discarded during the execution of the algorithm

Previous Results and Our Contribution

Maximization problem	State of the Art	Our result
Linear b-matching	$3+\varepsilon$	$2+\varepsilon$
Submod. b-matching	8.899	7.464
Submod. b-matching on k-hypergraph + matroid	$4k+O(1)$	$(8/3)k+O(1)$

Table: Comparison of our results with the State of the Art

Previous Results and Our Contribution

Semi-steaming maximum weight matching approximation:

- 6 [Feigenbaum, Kannan, McGregor, Suri, and Zhang, 2005]
- 5.83 [McGregor, 2005]
- 5.58 [Zelke, 2010]
- $4.91 + \varepsilon$ [Epstein, Levin, Mestre, and Segev, 2010]
- $4 + \varepsilon$ [Crouch and Stubbs, 2014]
- $2 + \varepsilon$ [Paz and Schwartzman, 2018]

Semi-steaming maximum weight b-matching approximation:

- $4 + \varepsilon$ [Crouch and Stubbs, 2014]
- $3 + \varepsilon$ [Levin and Wajc, 2021]
- $2 + \varepsilon$ [our paper]

Previous Results and Our Contribution

	matching	b-matching
linear	$2+\varepsilon$	$3+\varepsilon$
monotone	$3+2\sqrt{2}\approx 5.828$	$3+2\sqrt{2}\approx 5.828$
general	$4+2\sqrt{3}\approx 7.464$	$4+2\sqrt{6}\approx 8.899$

Table: State of the Art for semi-streaming maximum submodular matching and b-matching

Now we are able to achieve for the b-matching constraint the same bounds as for the simple matching constraint for all these types of submodular functions.

Previous Results and Our Contribution

For the case of the submodular maximum b-matching on a k-uniform hypergraph with a matroid constraint:

- current algorithms provide a $4k + O(1)$ approximation [Chekuri, Gupta, and Quanrud, 2015; Feldman, Karbasi, and Kazemi, 2018]
- we get approximation ratios bounded by $(8/3)k + O(1)$

General Presentation of the Algorithm

We use a similar framework to [Paz and Schwartzman, 2018; Levin and Wajc, 2021]:

- a local ratio technique for the streaming phase to build a set S
- followed by a greedy construction phase going back in time on the elements of the set S

General Presentation of the Algorithm

We use a similar framework to [Paz and Schwartzman, 2018; Levin and Wajc, 2021]:

- a local ratio technique for the streaming phase to build a set S *using a new data structure to store the edges, allowing us to perform a more accurate discrimination*
- followed by a greedy construction phase going back in time on the elements of the set S *using the aforementioned data structure to make better choices*

General Presentation of the Algorithm

We use a similar framework to [Paz and Schwartzman, 2018; Levin and Wajc, 2021]:

- a local ratio technique for the streaming phase to build a set S *using a new data structure to store the edges, allowing us to perform a more accurate discrimination*
- followed by a greedy construction phase going back in time on the elements of the set S *using the aforementioned data structure to make better choices*

For this presentation, we will focus on the Maximum Weight b-Matching problem. For ease of description, we explain how to achieve a 2-approximation; the issue of space complexity will be tackled later.

Description of the Streaming Phase

- for each vertex v, we have b_v queues $Q_{v,1}, \cdots, Q_{v,b_v}$
- these queues contain the edges in S incident to v

Description of the Streaming Phase

- for each vertex v, we have b_v queues $Q_{v,1}, \cdots, Q_{v,b_v}$
- these queues contain the edges in S incident to v
- we define the reduced weight $w_v(e)$ of an edge e in a queue $Q_{v,i}$ as the sum of the *gains* of the elements that arrived earlier in the queue

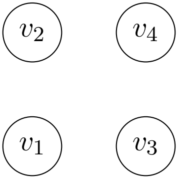
Description of the Streaming Phase

- for each vertex v, we have b_v queues $Q_{v,1}, \cdots, Q_{v,b_v}$
- these queues contain the edges in S incident to v
- we define the reduced weight $w_v(e)$ of an edge e in a queue $Q_{v,i}$ as the sum of the *gains* of the elements that arrived earlier in the queue
- when an edge $e = \{u, v\}$ arrives, we define $w^*_u(e)$ (reps. $w^*_v(e)$) as minimum the sum of the gains among the queues of u (resp. v)

Description of the Streaming Phase

- for each vertex v, we have b_v queues $Q_{v,1}, \cdots, Q_{v,b_v}$
- these queues contain the edges in S incident to v
- we define the reduced weight $w_v(e)$ of an edge e in a queue $Q_{v,i}$ as the sum of the *gains* of the elements that arrived earlier in the queue
- when an edge $e = \{u, v\}$ arrives, we define $w^*_u(e)$ (reps. $w^*_v(e)$) as minimum the sum of the gains among the queues of u (resp. v)
- an edge e is added to S only if $w(e) > w^*_u(e) + w^*_v(e)$, in that case we define the *gain* $g(e) = w(e) - w^*_u(e) - w^*_v(e)$ and e is pushed into the corresponding queues, otherwise we set $g(e) = 0$
- we also set $w_v(e) = g(e) + w^*_v(e)$

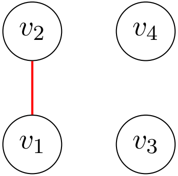
Description of the Streaming Phase



For $V = \{v_1, v_2, v_3, v_4\}$, $b_{v_1} = b_{v_2} = 2$ and $b_{v_3} = b_{v_4} = 1$.

$$Q_{v_1,1} = \emptyset \qquad Q_{v_1,2} = \emptyset \quad w_{v_1}(Q_{v_1,1}) = 0,\ w_{v_1}(Q_{v_1,2}) = 0$$
$$Q_{v_2,1} = \emptyset \qquad Q_{v_2,2} = \emptyset \quad w_{v_2}(Q_{v_2,1}) = 0,\ w_{v_2}(Q_{v_2,2}) = 0$$
$$Q_{v_3,1} = \emptyset \qquad w_{v_3}(Q_{v_3,1}) = 0$$
$$Q_{v_4,1} = \emptyset \qquad w_{v_4}(Q_{v_3,1}) = 0$$

Description of the Streaming Phase

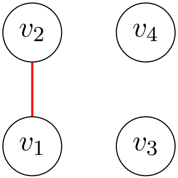


For $V = \{v_1, v_2, v_3, v_4\}$, $b_{v_1} = b_{v_2} = 2$ and $b_{v_3} = b_{v_4} = 1$.

$$
\begin{array}{lll}
*\, Q_{v_1,1} = \emptyset & Q_{v_1,2} = \emptyset & w_{v_1}(Q_{v_1,1}) = 0,\ w_{v_1}(Q_{v_1,2}) = 0 \\
*\, Q_{v_2,1} = \emptyset & Q_{v_2,2} = \emptyset & w_{v_2}(Q_{v_2,1}) = 0,\ w_{v_2}(Q_{v_2,2}) = 0 \\
\, Q_{v_3,1} = \emptyset & & w_{v_3}(Q_{v_3,1}) = 0 \\
\, Q_{v_4,1} = \emptyset & & w_{v_4}(Q_{v_3,1}) = 0
\end{array}
$$

$e_1 = \{v_1, v_2\}$, $w(e_1) = 1$
$w^*_{v_1}(e_1) = 0$ and $w^*_{v_2}(e_1) = 0$ so $g(e_1) = 1$

Description of the Streaming Phase

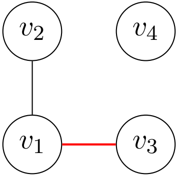


For $V = \{v_1, v_2, v_3, v_4\}$, $b_{v_1} = b_{v_2} = 2$ and $b_{v_3} = b_{v_4} = 1$.

$$
\begin{array}{lll}
* \, Q_{v_1,1} = \{e_1\} & Q_{v_1,2} = \emptyset & w_{v_1}(Q_{v_1,1}) = 1,\ w_{v_1}(Q_{v_1,2}) = 0 \\
* \, Q_{v_2,1} = \{e_1\} & Q_{v_2,2} = \emptyset & w_{v_2}(Q_{v_2,1}) = 1,\ w_{v_2}(Q_{v_2,2}) = 0 \\
\quad Q_{v_3,1} = \emptyset & & w_{v_3}(Q_{v_3,1}) = 0 \\
\quad Q_{v_4,1} = \emptyset & & w_{v_4}(Q_{v_3,1}) = 0
\end{array}
$$

$e_1 = \{v_1, v_2\}$, $w(e_1) = 1$
$w^*_{v_1}(e_1) = 0$ and $w^*_{v_2}(e_1) = 0$ so $g(e_1) = 1$

Description of the Streaming Phase

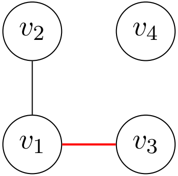


For $V = \{v_1, v_2, v_3, v_4\}$, $b_{v_1} = b_{v_2} = 2$ and $b_{v_3} = b_{v_4} = 1$.

$$\begin{array}{llll} *\, Q_{v_1,1} = \{e_1\} & Q_{v_1,2} = \emptyset & w_{v_1}(Q_{v_1,1}) = 1,\ w_{v_1}(Q_{v_1,2}) = 0 \\ Q_{v_2,1} = \{e_1\} & Q_{v_2,2} = \emptyset & w_{v_2}(Q_{v_2,1}) = 1,\ w_{v_2}(Q_{v_2,2}) = 0 \\ *\, Q_{v_3,1} = \emptyset & & w_{v_3}(Q_{v_3,1}) = 0 \\ Q_{v_4,1} = \emptyset & & w_{v_4}(Q_{v_3,1}) = 0 \end{array}$$

$e_2 = \{v_1, v_3\}$, $w(e_2) = 2$
$w^*_{v_1}(e_2) = 0$ and $w^*_{v_3}(e_2) = 0$ so $g(e_2) = 2$

Description of the Streaming Phase

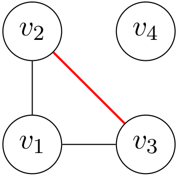


For $V = \{v_1, v_2, v_3, v_4\}$, $b_{v_1} = b_{v_2} = 2$ and $b_{v_3} = b_{v_4} = 1$.

$$\begin{array}{rrl} *\,Q_{v_1,1} = \{e_1\} & Q_{v_1,2} = \{e_2\} & w_{v_1}(Q_{v_1,1}) = 1,\ w_{v_1}(Q_{v_1,2}) = 2 \\ Q_{v_2,1} = \{e_1\} & Q_{v_2,2} = \emptyset & w_{v_2}(Q_{v_2,1}) = 1,\ w_{v_2}(Q_{v_2,2}) = 0 \\ *\,Q_{v_3,1} = \{e_2\} & & w_{v_3}(Q_{v_3,1}) = 2 \\ Q_{v_4,1} = \emptyset & & w_{v_4}(Q_{v_3,1}) = 0 \end{array}$$

$e_2 = \{v_1, v_3\}$, $w(e_2) = 2$
$w^*_{v_1}(e_2) = 0$ and $w^*_{v_3}(e_2) = 0$ so $g(e_2) = 2$

Description of the Streaming Phase

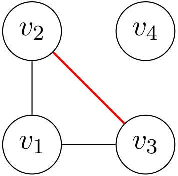


For $V = \{v_1, v_2, v_3, v_4\}$, $b_{v_1} = b_{v_2} = 2$ and $b_{v_3} = b_{v_4} = 1$.

$$\begin{array}{lll} Q_{v_1,1} = \{e_1\} & Q_{v_1,2} = \{e_2\} & w_{v_1}(Q_{v_1,1}) = 1,\ w_{v_1}(Q_{v_1,2}) = 2 \\ *\, Q_{v_2,1} = \{e_1\} & Q_{v_2,2} = \emptyset & w_{v_2}(Q_{v_2,1}) = 1,\ w_{v_2}(Q_{v_2,2}) = 0 \\ *\, Q_{v_3,1} = \{e_2\} & & w_{v_3}(Q_{v_3,1}) = 2 \\ Q_{v_4,1} = \emptyset & & w_{v_4}(Q_{v_3,1}) = 0 \end{array}$$

$e_3 = \{v_2, v_3\}$, $w(e_3) = 4$
$w^*_{v_2}(e_3) = 0$ and $w^*_{v_3}(e_3) = 2$ so $g(e_3) = 4 - 2 = 2$

Description of the Streaming Phase

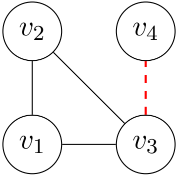


For $V = \{v_1, v_2, v_3, v_4\}$, $b_{v_1} = b_{v_2} = 2$ and $b_{v_3} = b_{v_4} = 1$.

$Q_{v_1,1} = \{e_1\}$	$Q_{v_1,2} = \{e_2\}$	$w_{v_1}(Q_{v_1,1}) = 1,\ w_{v_1}(Q_{v_1,2}) = 2$
$*\,Q_{v_2,1} = \{e_1\}$	$Q_{v_2,2} = \{e_3\}$	$w_{v_2}(Q_{v_2,1}) = 1,\ w_{v_2}(Q_{v_2,2}) = 2$
$*\,Q_{v_3,1} = \{e_3, e_2\}$		$w_{v_3}(Q_{v_3,1}) = 4$
$Q_{v_4,1} = \emptyset$		$w_{v_4}(Q_{v_3,1}) = 0$

$e_3 = \{v_2, v_3\}$, $w(e_3) = 4$
$w^*_{v_2}(e_3) = 0$ and $w^*_{v_3}(e_3) = 2$ so $g(e_3) = 4 - 2 = 2$

Description of the Streaming Phase

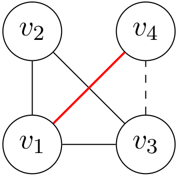


For $V = \{v_1, v_2, v_3, v_4\}$, $b_{v_1} = b_{v_2} = 2$ and $b_{v_3} = b_{v_4} = 1$.

$$
\begin{array}{lll}
\;\; Q_{v_1,1} = \{e_1\} & Q_{v_1,2} = \{e_2\} & w_{v_1}(Q_{v_1,1}) = 1,\ w_{v_1}(Q_{v_1,2}) = 2 \\
\;\; Q_{v_2,1} = \{e_1\} & Q_{v_2,2} = \{e_3\} & w_{v_2}(Q_{v_2,1}) = 1,\ w_{v_2}(Q_{v_2,2}) = 2 \\
* \, Q_{v_3,1} = \{e_3, e_2\} & & w_{v_3}(Q_{v_3,1}) = 4 \\
* \, Q_{v_4,1} = \emptyset & & w_{v_4}(Q_{v_3,1}) = 0
\end{array}
$$

$e_4 = \{v_3, v_4\}$, $w(e_4) = 3$
$w^*_{v_3}(e_4) = 4$ and $w^*_{v_4}(e_4) = 0$ so $g(e_4) = 0$

Description of the Streaming Phase

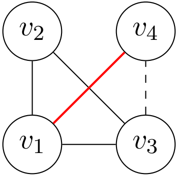


For $V = \{v_1, v_2, v_3, v_4\}$, $b_{v_1} = b_{v_2} = 2$ and $b_{v_3} = b_{v_4} = 1$.

$$
\begin{array}{llll}
* & Q_{v_1,1} = \{e_1\} & Q_{v_1,2} = \{e_2\} & w_{v_1}(Q_{v_1,1}) = 1,\ w_{v_1}(Q_{v_1,2}) = 2 \\
 & Q_{v_2,1} = \{e_1\} & Q_{v_2,2} = \{e_3\} & w_{v_2}(Q_{v_2,1}) = 1,\ w_{v_2}(Q_{v_2,2}) = 2 \\
 & Q_{v_3,1} = \{e_3, e_2\} & & w_{v_3}(Q_{v_3,1}) = 4 \\
* & Q_{v_4,1} = \emptyset & & w_{v_4}(Q_{v_3,1}) = 0
\end{array}
$$

$e_5 = \{v_1, v_4\}$, $w(e_5) = 3$
$w^*_{v_1}(e_5) = 1$ and $w^*_{v_4}(e_5) = 0$ so $g(e_5) = 3 - 1 = 2$

Description of the Streaming Phase

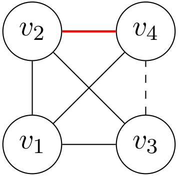


For $V = \{v_1, v_2, v_3, v_4\}$, $b_{v_1} = b_{v_2} = 2$ and $b_{v_3} = b_{v_4} = 1$.

$$
\begin{array}{lll}
* \, Q_{v_1,1} = \{e_5, e_1\} & Q_{v_1,2} = \{e_2\} & w_{v_1}(Q_{v_1,1}) = 3,\ w_{v_1}(Q_{v_1,2}) = 2 \\
\, Q_{v_2,1} = \{e_1\} & Q_{v_2,2} = \{e_3\} & w_{v_2}(Q_{v_2,1}) = 1,\ w_{v_2}(Q_{v_2,2}) = 2 \\
\, Q_{v_3,1} = \{e_3, e_2\} & & w_{v_3}(Q_{v_3,1}) = 4 \\
* \, Q_{v_4,1} = \{e_5\} & & w_{v_4}(Q_{v_3,1}) = 2
\end{array}
$$

$e_5 = \{v_1, v_4\}$, $w(e_5) = 3$
$w^*_{v_1}(e_5) = 1$ and $w^*_{v_4}(e_5) = 0$ so $g(e_5) = 3 - 1 = 2$

Description of the Streaming Phase

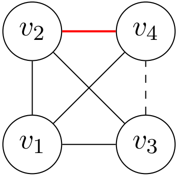


For $V = \{v_1, v_2, v_3, v_4\}$, $b_{v_1} = b_{v_2} = 2$ and $b_{v_3} = b_{v_4} = 1$.

$$
\begin{array}{llll}
& Q_{v_1,1} = \{e_5, e_1\} & Q_{v_1,2} = \{e_2\} & w_{v_1}(Q_{v_1,1}) = 3,\ w_{v_1}(Q_{v_1,2}) = 2 \\
* & Q_{v_2,1} = \{e_1\} & Q_{v_2,2} = \{e_3\} & w_{v_2}(Q_{v_2,1}) = 1,\ w_{v_2}(Q_{v_2,2}) = 2 \\
& Q_{v_3,1} = \{e_3, e_2\} & & w_{v_3}(Q_{v_3,1}) = 4 \\
* & Q_{v_4,1} = \{e_5\} & & w_{v_4}(Q_{v_3,1}) = 2
\end{array}
$$

$e_6 = \{v_2, v_4\}$, $w(e_6) = 5$
$w^*_{v_2}(e_6) = 1$ and $w^*_{v_4}(e_6) = 2$ so $g(e_6) = 5 - 2 - 1 = 2$

Description of the Streaming Phase

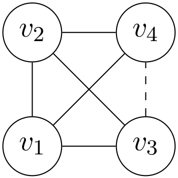


For $V = \{v_1, v_2, v_3, v_4\}$, $b_{v_1} = b_{v_2} = 2$ and $b_{v_3} = b_{v_4} = 1$.

$$\begin{array}{llll}
 & Q_{v_1,1} = \{e_5, e_1\} & Q_{v_1,2} = \{e_2\} & w_{v_1}(Q_{v_1,1}) = 3,\ w_{v_1}(Q_{v_1,2}) = 2 \\
* & Q_{v_2,1} = \{e_6, e_1\} & Q_{v_2,2} = \{e_3\} & w_{v_2}(Q_{v_2,1}) = 3,\ w_{v_2}(Q_{v_2,2}) = 2 \\
 & Q_{v_3,1} = \{e_3, e_2\} & & w_{v_3}(Q_{v_3,1}) = 4 \\
* & Q_{v_4,1} = \{e_6, e_5\} & & w_{v_4}(Q_{v_3,1}) = 4
\end{array}$$

$e_6 = \{v_2, v_4\}$, $w(e_6) = 5$
$w^*_{v_2}(e_6) = 1$ and $w^*_{v_4}(e_6) = 2$ so $g(e_6) = 5 - 2 - 1 = 2$

Description of the Streaming Phase



For $V = \{v_1, v_2, v_3, v_4\}$, $b_{v_1} = b_{v_2} = 2$ and $b_{v_3} = b_{v_4} = 1$.

$$Q_{v_1,1} = \{e_5, e_1\} \quad Q_{v_1,2} = \{e_2\} \quad w_{v_1}(Q_{v_1,1}) = 3,\ w_{v_1}(Q_{v_1,2}) = 2$$

$$Q_{v_2,1} = \{e_6, e_1\} \quad Q_{v_2,2} = \{e_3\} \quad w_{v_2}(Q_{v_2,1}) = 3,\ w_{v_2}(Q_{v_2,2}) = 2$$

$$Q_{v_3,1} = \{e_3, e_2\} \quad w_{v_3}(Q_{v_3,1}) = 4$$

$$Q_{v_4,1} = \{e_6, e_5\} \quad w_{v_4}(Q_{v_3,1}) = 4$$

Description of the Streaming Phase

Algorithm 1 Streaming phase for weighted matching

$S \leftarrow \emptyset$
$\forall v \in V : Q_v \leftarrow (Q_{v,1} = \emptyset, \cdots, Q_{v,b_v} = \emptyset)$ ▷ b_v queues for a vertex v
for $e = e_t$, $1 \leq t \leq |E|$ an edge from the stream **do**
  for $u \in e$ **do**
    $w_u^*(e) \leftarrow \min\{w_u(Q_{u,q}.top()) : 1 \leq q \leq b_u\}$
    $q_u(e) \leftarrow q$ such that $w_u(Q_{u,q}.top()) = w_u^*(e)$
  if $w(e) > \sum_{u \in e} w_u^*(e)$ **then**
    $g(e) \leftarrow w(e) - \sum_{u \in e} w_u^*(e)$
    $S \leftarrow S \cup \{e\}$
    for $u \in e$ **do**
      $w_u(e) \leftarrow w_u^*(e) + g(e)$
      $r_u(e) \leftarrow Q_{u,q_u(e)}.top()$ ▷ $r_u(e)$ is the element below e in the queue
      $Q_{u,q_u(e)}.push(e)$ ▷ add e on the top of the smallest queue

We can show that $g(S) \geq 1/2 \cdot w(M^{opt})$.

Description of the Construction Phase

- start with $M = \emptyset$
- in the reverse arrival order in S, add an edge e to M if no queue it appears in contains any element already in M

Description of the Construction Phase

- start with $M = \emptyset$
- in the reverse arrival order in S, add an edge e to M if no queue it appears in contains any element already in M
- in fact, we want a b-matching of weight at least $g(S)$, and when an edge e is taken, all the gains for elements below it in the queues are counted, as $w(e) = g(e) + \sum_{u \in e} w_u^*(e)$

Description of the Construction Phase

Going back to the previous example:

$$\begin{array}{ll} Q_{v_1,1} = \{e_5, e_1\} & Q_{v_1,2} = \{e_2\} \\ Q_{v_2,1} = \{e_6, e_1\} & Q_{v_2,2} = \{e_3\} \\ Q_{v_3,1} = \{e_3, e_2\} & \\ Q_{v_4,1} = \{e_6, e_5\} & \end{array}$$

$M = \emptyset$
$w(M) = 0$

Description of the Construction Phase

Going back to the previous example:

$$\begin{array}{ll} Q_{v_1,1} = \{e_5, e_1\} & Q_{v_1,2} = \{e_2\} \\ Q_{v_2,1} = \{e_6, e_1\} & Q_{v_2,2} = \{e_3\} \\ Q_{v_3,1} = \{e_3, e_2\} & \\ Q_{v_4,1} = \{e_6, e_5\} & \end{array}$$

$M = \{e_6\}$
$w(M) = w(e_6) = g(e_6) + g(e_5) + g(e_1)$

Description of the Construction Phase

Going back to the previous example:

$$
\begin{array}{ll}
Q_{v_1,1} = \{e_5, e_1\} & Q_{v_1,2} = \{e_2\} \\
Q_{v_2,1} = \{e_6, e_1\} & Q_{v_2,2} = \{e_3\} \\
Q_{v_3,1} = \{e_3, e_2\} & \\
Q_{v_4,1} = \{e_6, e_5\} &
\end{array}
$$

$M = \{e_6, e_3\}$
$w(M) = w(e_6) + w(e_3) = g(e_6) + g(e_5) + g(e_1) + g(e_3) + g(e_2)$

Description of the Construction Phase

Algorithm 2 Greedy construction phase

$M \leftarrow \emptyset$
$\forall e \in S : z_e \leftarrow 1$
for $e \in S$ in reverse order **do**
    if $z_e = 0$ **then continue** ▷ skip edge e if it is marked
    $M \leftarrow M \cup \{e\}$
    for $u \in e$ **do**
        $c \leftarrow e$
        while $c \neq \perp$ **do**
            $z_c \leftarrow 0$ ▷ mark elements below e in each queue
            $c \leftarrow r_u(c)$
return M

Outline of the Analysis

Theorem

Algorithms 1 and 2 provide a 2-approximation for the maximum weight b-matching problem.

The main steps are:

- it holds that $2g(S) \geq w(M^{opt})$
- the construction phase builds a feasible b-matching M such that $w(M) \geq g(S)$

Outline of the Analysis

- elements on top of queues $Q_{v,i}$ are the biggest regarding the reduced weight w_v, and the sum of their reduced weight is $g(\delta(v))$

Outline of the Analysis

- elements on top of queues $Q_{v,i}$ are the biggest regarding the reduced weight w_v, and the sum of their reduced weight is $g(\delta(v))$
- as a result, for any b-matching M', it holds that $w_v(M' \cap \delta(v)) \leq w_v(\{Q_v.top(), 1 \leq i \leq b_v\}) = g(\delta(v))$

Outline of the Analysis

- elements on top of queues $Q_{v,i}$ are the biggest regarding the reduced weight w_v, and the sum of their reduced weight is $g(\delta(v))$
- as a result, for any b-matching M', it holds that $w_v(M' \cap \delta(v)) \leq w_v(\{Q_v.top(), 1 \leq i \leq b_v\}) = g(\delta(v))$
- we can use it to show that $w(M') \leq 2g(S)$ for any b-matching M'.

Outline of the Analysis

- elements on top of queues $Q_{v,i}$ are the biggest regarding the reduced weight w_v, and the sum of their reduced weight is $g(\delta(v))$
- as a result, for any b-matching M', it holds that $w_v(M' \cap \delta(v)) \leq w_v(\{Q_v.top(), 1 \leq i \leq b_v\}) = g(\delta(v))$
- we can use it to show that $w(M') \leq 2g(S)$ for any b-matching M'.
- moreover, the set M built during the construction phase is a b-matching of weight at least $g(S)$.

Outline of the Analysis

Below are listed the steps of the analysis, where M^{opt} denotes the optimal b-matching:

i. $\forall v \in V$, $g(\delta(v)) = g(\delta(v) \cap S) = \sum_{i=1}^{b_v} w(Q_{v,i}.top())$
ii. $\{Q_{v,q}.top() : 1 \leq q \leq b_v\}$ contains the b_v heaviest elements of $S \cap \delta(v)$ in terms of reduced weights w_v
iii. $\forall v \in V$, $w_v(Q_v) := \sum_{i=1}^{b_v} w(Q_{v,i}.top()) \geq w_v(M^{opt} \cap \delta(v))$
iv. $2g(S) \geq w(M^{opt})$
v. Algorithm 2 outputs a feasible b-matching M with weight $w(M) \geq g(S)$

Analysis

Proposition

1. *For all $v \in V$ we have $g(\delta(v)) = g(\delta(v) \cap S) = w_v(Q_v)$, where $w_v(Q_v) = \sum_{i=1}^{b_v} w(Q_{v,i}.top())$.*
2. *The set $\{Q_{v,q}.top() : 1 \leq q \leq b_v\}$ contains the b_v heaviest elements of $S \cap \delta(v)$ in terms of reduced weights w_v.*

Algorithm 1 Streaming phase for weighted matching

$S \leftarrow \emptyset$
$\forall v \in V : Q_v \leftarrow (Q_{v,1} = \emptyset, \cdots, Q_{v,b_v} = \emptyset)$ ▷ b_v queues for a vertex v
for $e = e_t$, $1 \leq t \leq |E|$ an edge from the stream **do**
  for $u \in e$ **do**
    $w_u^*(e) \leftarrow \min\{w_u(Q_{u,q}.top()) : 1 \leq q \leq b_u\}$
    $q_u(e) \leftarrow q$ such that $w_u(Q_{u,q}.top()) = w_u^*(e)$
  if $w(e) > \sum_{u \in e} w_u^*(e)$ **then**
    $g(e) \leftarrow w(e) - \sum_{u \in e} w_u^*(e)$
    $S \leftarrow S \cup \{e\}$
    for $u \in e$ **do**
      $w_u(e) \leftarrow w_u^*(e) + g(e)$
      $r_u(e) \leftarrow Q_{u,q_u(e)}.top()$ ▷ $r_u(e)$ is the element below e in the queue
      $Q_{u,q_u(e)}.push(e)$ ▷ add e on the top of the smallest queue

Analysis

Lemma

$\forall v \in V$, $w_v(Q_v) \geq w_v(M^{opt} \cap \delta(v))$.

- by the previous proposition, $w_v(Q_v)$ is exactly the sum of the reduced weights of the b_v heaviest elements in $S \cap \delta(v)$
- $\forall e = e_t \in M^{opt} \backslash S$, $w_v(e_t) = \min\{w_v(Q_{v,q}^{(t-1)}) : 1 \leq q \leq b_v\} \leq \min\{w_v(Q_{v,q}^{|E|}) : 1 \leq q \leq b_v\}$

Analysis

Lemma

$2g(S) \geq w(M^{opt})$

For any $e = \{u, v\}$, $w_u(e) + w_v(e) \geq w(e)$, so

$$\begin{aligned} w(M^{opt}) &\leq \sum_{e=\{u,v\}\in M^{opt}} w_u(e) + w_v(e) \\ &= \sum_{u\in V} w_u(M^{opt} \cap \delta(u)) \\ &\leq \sum_{u\in V} w_u(Q_u) = \sum_{u\in V} g(S \cap \delta(u)) = 2g(S). \end{aligned}$$

Analysis

Lemma

Algorithm 2 builds a b-matching M with $w(M) \geq g(S)$.

Using $w(e) = g(e) + \sum_{u \in e} w_u^*(e) = g(e) + \sum_{u \in e} \sum_{e' \in Q_{u,q_u(e)}^{(t-1)}} g(e')$.

Algorithm 1 Streaming phase for weighted matching

$S \leftarrow \emptyset$
$\forall v \in V : Q_v \leftarrow (Q_{v,1} = \emptyset, \cdots, Q_{v,b_v} = \emptyset)$ ▷ b_v queues for a vertex v
for $e = e_t$, $1 \leq t \leq |E|$ an edge from the stream **do**
  for $u \in e$ **do**
    $w_u^*(e) \leftarrow \min\{w_u(Q_{u,q}.top()) : 1 \leq q \leq b_u\}$
    $q_u(e) \leftarrow q$ such that $w_u(Q_{u,q}.top()) = w_u^*(e)$
  if $w(e) > \sum_{u \in e} w_u^*(e)$ **then**
    $g(e) \leftarrow w(e) - \sum_{u \in e} w_u^*(e)$
    $S \leftarrow S \cup \{e\}$
    for $u \in e$ **do**
      $w_u(e) \leftarrow w_u^*(e) + g(e)$
      $r_u(e) \leftarrow Q_{u,q_u(e)}.top()$ ▷ $r_u(e)$ is the element below e in the queue
      $Q_{u,q_u(e)}.push(e)$ ▷ add e on the top of the smallest queue

Analysis

Using $w(e) = g(e) + \sum_{u \in e} w_u^*(e) = g(e) + \sum_{u \in e} \sum_{e' \in Q_{u,q_u(e)}^{(t-1)}} g(e')$.

Algorithm 2 Greedy construction phase

$M \leftarrow \emptyset$
$\forall e \in S : z_e \leftarrow 1$
for $e \in S$ in reverse order **do**
  if $z_e = 0$ **then continue** ▷ skip edge e if it is marked
  $M \leftarrow M \cup \{e\}$
  for $u \in e$ **do**
    $c \leftarrow e$
    while $c \neq \perp$ **do**
      $z_c \leftarrow 0$ ▷ mark elements below e in each queue
      $c \leftarrow r_u(c)$
return M

Each gain $g(e)$ for is paid for by at least one element of M.

Analysis

Theorem

Algorithms 1 and 2 provide a 2-approximation for the maximum weight b-matching problem.

Remark

It is straightforward to extend our algorithm to a k-uniform hypergraph, for which we can get an approximation ratio of k.

Making the Algorithm Memory-Efficient

The key ideas, from [Ghaffari and Wajc, 2019], are:

- to choose a constant $\alpha = 1 + \varepsilon > 0$ and replace the test $w(e) > \sum_{u \in e} w_u^*(e)$ by $w(e) > \alpha \sum_{u \in e} w_u^*(e)$, so that in the queues the reduces weights w_v grow exponentially, without hurting too much the approximation ratio
- to delete the elements that are far below in the queues, because these elements have very small gains

Making the Algorithm Memory-Efficient

Here, M_{max} denotes the maximum cardinality b-matching and W the maximum ratio between two non-zero weights.

Theorem

Using the first memory optimization, we can then recover a $2+\varepsilon$ *approximation using* $O\left(\log_{1+\varepsilon}(W/\varepsilon)\cdot|M_{max}|\right)$ *variables.*

Theorem

Using both memory optimizations, we can then recover a $2+\varepsilon$ *approximation using* $O\left(\log_{1+\varepsilon}(1/\varepsilon)\cdot|M_{max}|+\sum_{v\in V} b_v\right)$ *variables.*

Generalization to Submodular Functions

For submodular functions:

- use the memory-efficient version of the algorithm for the streaming phase, and replace $w(e)$ by the marginal gain $f(e \mid S) := f(S \cup \{e\}) - f(S)$

this analysis borrows ideas from [Levin and Wajc, 2021].

Generalization to Submodular Functions

For submodular functions:

- use the memory-efficient version of the algorithm for the streaming phase, and replace $w(e)$ by the marginal gain $f(e \mid S) := f(S \cup \{e\}) - f(S)$
- $g(S)$ will be a significant portion of $f(S \mid \emptyset)$, if ε is big enough

this analysis borrows ideas from [Levin and Wajc, 2021].

Generalization to Submodular Functions

For submodular functions:

- use the memory-efficient version of the algorithm for the streaming phase, and replace $w(e)$ by the marginal gain $f(e \mid S) := f(S \cup \{e\}) - f(S)$
- $g(S)$ will be a significant portion of $f(S \mid \emptyset)$, if ε is big enough
- $g(S)$ will also be a significant part of $f(M^{opt} \mid S)$, if ε is small enough

this analysis borrows ideas from [Levin and Wajc, 2021].

Generalization to Submodular Functions

For submodular functions:

- use the memory-efficient version of the algorithm for the streaming phase, and replace $w(e)$ by the marginal gain $f(e \mid S) := f(S \cup \{e\}) - f(S)$
- $g(S)$ will be a significant portion of $f(S \mid \emptyset)$, if ε is big enough
- $g(S)$ will also be a significant part of $f(M^{opt} \mid S)$, if ε is small enough
- The construction phase will provide a b-matching M such that $f(M) \geq g(S) + f(\emptyset)$

this analysis borrows ideas from [Levin and Wajc, 2021].

Generalization to Submodular Functions

For submodular functions:

- use the memory-efficient version of the algorithm for the streaming phase, and replace $w(e)$ by the marginal gain $f(e \mid S) := f(S \cup \{e\}) - f(S)$
- $g(S)$ will be a significant portion of $f(S \mid \emptyset)$, if ε is big enough
- $g(S)$ will also be a significant part of $f(M^{opt} \mid S)$, if ε is small enough
- The construction phase will provide a b-matching M such that $f(M) \geq g(S) + f(\emptyset)$
- randomization for non-monotone submodular functions

this analysis borrows ideas from [Levin and Wajc, 2021].

Conclusion

Main results:

Maximization problem	State of the Art	Our result
Linear b-matching	$3 + \varepsilon$	$2 + \varepsilon$
Submod. b-matching	8.899	7.464
Submod. b-matching on k-hypergraph + matroid	$4k + O(1)$	$(8/3)k + O(1)$

Table: Comparison of our results with the State of the Art

Main ideas:

- a local ratio technique for the streaming phase to build a set S *using one queue per vertex capacity to store the edges*
- followed by a greedy construction phase going back in time on the elements of the set S *using the aforementioned queues to make better choices*