Dariusz Leniowski

Uniwersytet Warszawski

On edge usage in adwords problem

Praca semestralna nr 2

(semestr zimowy 2012/13)

Opiekun pracy: Bartłomiej Bosek

On edge usage in adwords problem

Dariusz Leniowski

September 2013

Abstract

This paper proposes a new approach to one-sided online maximum cardinality bipartite matching algorithms based on augmenting paths. It analyzes how distribution of workload among different vertices and edges could ensure that none of them will be handled excessively many times. We bound the number of times an edge is used in the augmenting paths. In particular, a greedy approach that at each step minimizes the maximum use of edges achieves a bound of $O(\sqrt{n})$ times. Presented theorems and methods provide theoretical foundations for an algorithm running in amortized $O(m\sqrt{n})$ time that maintains the maximum cardinality matching in the adwords problem. This is a part of joint work with Bartłomiej Bosek, Piotr Sankowski and Anna Zych.

1 Introduction

The adwords problem is a special case of online bipartite maximum cardinality matching problem, sometimes being called its one-sided version. The traditional formulation involves a bipartite graph of which one side (we will call it white) is known beforehand, while the vertices from the other side (black) are presented online together with all their edges.

Although the problem itself is as old as the full (two-sided) online bipartite matching, the main practical motivation comes from mobile and internet ads industry, most prominently, Google Adwords. The scheme considers a known set of advertisers that wish to show their ads to customers and an unknown stream of publishers (basically instances of web pages or mobile devices used by the targeted customer) who could make it possible. As both publishers and advertisers have preferences on who they want to deal with, the challenge is to maximize number of matches made. In a general setting the graph might be weighted which would describe the traffic load a publisher can handle, however, in this work we will focus in the unweighted case where each publisher corresponds to a single ad impression.

Recent research focused on approximation algorithms which tried to obtain the highest possible cardinality (or weight) without changing its decisions. The investigated models fall into three classes:

- *adversarial*, where the input graph and the order of black vertices are given by an adversary; in particular, it can adjust its strategy based on the decisions made by the algorithm;
- *random order* (or unknown stochastic), where the adversary creates the graph, but it has no control over the order in which the black vertices arrive; significantly, the graph has to be fixed before the algorithm starts;

- *IID* (or known stochastic), where the input graph is a random graph with vertices (and their edges) coming from some known distribution, each independent of the others.

The adversarial case was solved by Karp, Vazirani and Vazirani in [7]. They provide a proof that the greedy algorithm has competetive ratio of $\frac{1}{2}$ and provide another algorithm (called *RANKING*) with competetive ratio of $1 - \frac{1}{e} \approx 0.63$. In the random order setting, the greedy becomes the dual of *RANKING* and also achieves bound $1 - \frac{1}{e}$. The breakthrough was made by Feldman, Mehta, Mirrokni and Muthukrishnan in [4], where the authors used an offline hinting technique to beat the $1 - \frac{1}{e}$ bound in the case of IID. They also provide a upper bound of 0.9898. However, with additional assumptions Feldman, Henzinger, Korula, Mirrokni and Stein [3] show an $1 - \varepsilon$ approximation for the random order setting which also contains the IID case. The lower and upper bounds were tightened by Bahmani and Kapralov [1] to 0.699 and 0.902 respectively and have been recently strengthened even more by Manshadi, Gharan and Saberi [8] to 0.702 and 0.823 (or 0.86 with the assumption of integral arrivals). Some additional progress was achieved by Jaillet and Lu [6] who improved the lower bound to 0.706, 0.725 and 0.729 under various assumptions.

All the above works focused at competitive ratio of algorithms that keep their decisions. On the other hand, this text is concerned with maintaining the maximum cardinality matching during the algorithm run and by necessity it changes the previously calculated matchings. Very little research has been done in this area and up to author's best knowledge, no papers exists that directly address this issue.

The rest of this article is as follows: section 2 contains all the necessary definitions, section 3 starts with the formulation of the paper's main theorem, proceeds by number of lemmas and ends with the proof of the theorem. The whole text is concluded with a short discussion on the obtained results and their implications.

2 Definitions

Let $G = \langle U \uplus V, E \rangle$ be a bipartite graph, where vertices $u \in U$ will be called white and $v \in V$ will be called black. For the sake of simplicity we assume that G contains perfect matching saturating black vertices, also we will denote $|U| = |V| = n$ and $|E| = m$, and for convenience add $r = \sqrt{n}$. We consider the following process:

- start with $G_0 = \langle U \uplus \varnothing, \varnothing \rangle = \langle U_0 \uplus V_0, E_0 \rangle$,
- in turn $t+1$ we add a new vertex $v_{t+1} \in V$ along with all its adjacent edges $\check{E}_{t+1} = \{(u, v_{t+1}) \mid u \in U, (u, v_{t+1}) \in E\}$, that is $G_{t+1} = \big\langle U \uplus V_t \cup \{v_{t+1}\}, E_t \cup \check{E}_{t+1} \big\rangle$,
- at all times there is a matching $M_t \subseteq E_t$ saturating black vertices,
- we finish after all vertices $v \in V$ have been added, $G_N = G$.

Moreover, for convenience we orient the edges of graphs G_t so that the matched edges lead from white to black vertices and the unmatched edges go in the opposite direction, that is $u \to v$ for $(u, v) \in M_t$ and $v \to u$ for $(u, v) \notin M_t$. A successor of w is any neighbour w' such that $(w \to w') \in E_t$. The unmatched white vertices will be called *seeds*, $\mathcal{S}_t = \{u \in U \mid (u, v) \notin M_t \text{ for any } v \in V\}$. Furthermore, we will use notions of ranks and tiers. Rank of an object is intuitively the number of times some augmenting path has visited that object, while tier is the minimal rank that is needed to reach some seed.

An augmenting path is a directed path from an unmatched black vertex to an unmatched white vertex (a seed). For the purpose of our approach we allow augmenting paths to be non simple, i.e. an augmenting path may visit vertices and edges multiple times. Whenever an augmenting path visits an edge, it immediately changes its orientation. We allow any single edge to be traversed at most ψ times, this allows the path to turn around using the newly-reversed edges. Furthermore, by definition of G_t, v_t is already matched in M_t, i.e. the augmenting path of turn t is the one that connects v_t to a seed of G_{t-1}. However, ranks and tiers are calculated after the whole augmenting path have been applied, the reason being that in-middle-of-applying states might lead to undesirable configurations. As the augmenting path can traverse an edge multiple times, let it be represented as a multiset of edges $A_t : E \to \{0, \dots, \psi\}$. We define the *rank* of edge $e \in E$ at step t as

$$\operatorname{rank}_t(e) = \sum_{k=1}^{t} A_k(e).$$

This mapping extends to sets or sequences of edges (e.g. paths), that is

$$\operatorname{rank}_t(P) = \max_{e \in P} \operatorname{rank}_t(e).$$

The *tier* of some vertex w is the minimal rank of a (directed) path in G_t leading to a seed

$$\operatorname{tier}_t(w) = \min_{s \in \mathcal{S}_t, w \xrightarrow{P} s} \operatorname{rank}_t(P).$$

An edge will be called *tiered* if it leads from a vertex to any of its successors of the smallest possible tier and its rank is not bigger than the tier of source vertex. A path consisting of such edges will be also called tiered, while the set of all tiered edges in turn t will be denoted by $\vec{E}_t$,

$$\begin{aligned} \vec{E}_t = \Big\{ (uv) \in E_t \;\Big|\; & \operatorname{tier}_t(u) \geq \operatorname{tier}_t(v), \\ & \operatorname{tier}_t(u) \geq \operatorname{rank}_t\big((uv)\big), \\ & \forall (uw) \in E_t.\ \operatorname{tier}_t(w) \geq \operatorname{tier}_t(v) \Big\}. \end{aligned}$$

While applying an augmenting path, we change the directions of edges immediately. Together with the ranks and tiers being calculated after the whole path have been applied, this allows the path to retrace back its steps as long as it stays among vertices of a single tier and no edge have been used more than ψ times. In other words, the requirement for the tiers to be non-increasing and smallest possible, makes the paths to seek the next smaller tier and once it is found, the path can't go back. The value of tier guarantees that some seed can be reached.

A *seeded* tree T is any directed tree rooted at some seed $s \in \mathcal{S}_t$. Observe that vertices for which no seed is reachable, cannot belong to a seeded tree. This state is terminal, i.e. once there is no seeded tree a vertex belongs to, no seed will be reachable ever again. The reason for this is that any new edges that might connect to such vertex, are pointed *to* it and nothing can switch their direction since every augmenting path requires a reachable seed in the first place. Hence, vertices for which no seed is reachable can be, without loss of generality, removed from the graph. Intuitively, we set their tier to be infinite and from this point we assume that no such vertex belongs to the graph.

Any forest of $|\mathcal{S}_t|$ vertex-disjoint seeded trees that spans G (vertices of finite tier, the rest is not considered anymore) will be also called seeded. The set of all seeded forests at step t will be denoted $\mathcal{F}_t$.

For a tree T of a seeded forest F, let the closure $\overline{T}$ of T be the collection of all edges that end in T. In particular, for two different trees T_1, T_2 of F, we have $\overline{T_1} \cap \overline{T_2} = \varnothing$. We define $\mathrm{rank}_t(T) = \max_{e \in \overline{T}} \mathrm{rank}_t(e)$. Given a seeded forest F as a sequence of trees $T_0, T_1, \ldots, T_{|\mathcal{S}_t|-1}$ sorted by its ranks $\mathrm{rank}_t(T_k) \geq \mathrm{rank}_t(T_{k+1})$, we set

$$\mathrm{rank}_t(F) = \max_{k \in \{0, \ldots, |\mathcal{S}_t|-1\}} \mathrm{rank}_t(T_k) + \psi k.$$

This allows us to define the function Φ as:

$$\Phi_t = \max_{F \in \mathcal{F}_t} \mathrm{rank}_t(F). \tag{1}$$

3 Proof

This is the main theorem of this section, its proof is at the very end.

Theorem 3.1. *The algorithm that repeatedly applies tiered augmenting paths satisfies* $\mathrm{rank}_t(e) = O(r)$ *for any* $e \in E_t$ *for* $t = 0, \ldots, n$.

Corollary 3.2. The total length of all tiered augmenting paths applied by the algorithm is $O(mr)$, that is

$$\sum_{e \in E} \mathrm{rank}_n(e) = O(mr).$$

Before we move to the proof of Theorem 3.1 we introduce a few helpful lemmas and observations.

Lemma 3.3.

- *For any seed* $s \in \mathcal{S}_t$, *every path* $w \xrightarrow{P} s$ *contains an edge* e *such that* $\mathrm{rank}_t(e) \geq \mathrm{tier}_t(w)$.
- *For any* w *there exists a tiered path* $w \xrightarrow{P} s$ *such that* $\mathrm{rank}_t(P) = \mathrm{tier}_t(w)$.
- *For any* w *and any tiered path* $w \xrightarrow{P} s$ *we have* $\mathrm{rank}_t(P) = \mathrm{tier}_t(w)$.
- *Any path* P *is tiered in turn* t *if and only if there exists a seeded tree* T *of a seeded forest* $F \in \mathcal{F}_t$ *such that* $F \subseteq_{E_t} \vec{E}_t$ *and* $P \subseteq T$.

Proof. Immediate from the definitions of $\mathrm{tier}_t(w)$ and tiered path. □

Lemma 3.4. *The* $\mathrm{tier}_t(w)$ *is non-decreasing in* t *for any vertex* w.

Proof. Fix some seeded forest F in turn t such that $F \subseteq_E \vec{E}_t$, its existence follows from Lemma 3.3. Let $\preceq$ be the relation of reachability in F, that is,

$$v \preceq u \iff \exists T \in F.\, \exists P \subseteq T.\, u \xrightarrow{P} v.$$

We note, that $\prec$ is well-founded strict partial order with seeds being precisely the minimal elements and proceed further by induction on the aforementioned partial order.

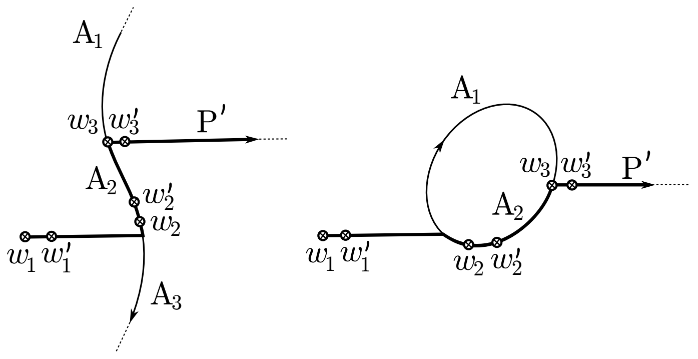
Surely, for any seed $s \in \mathcal{S}_t$ the tier is zero, so it could not have decreased.

Similarly, if $w = v_t$, that is, the newly added vertex, then by definition its tier is non-negative.

Let w be arbitrary vertex of G_t. Denote by A be the set of edges that changed direction between steps t and $(t-1)$. Let P be the best (with regard to rank) path leading from w to any seed $s \in \mathcal{S}_t$. By Lemma 3.3 we have that there exists a tiered path P' of the same rank that belongs to some tree of F. If $A \cap P' = \varnothing$, then P' is a valid path at step $(t-1)$, and because $\mathrm{rank}_t(e)$ is non-decreasing in t for any $e \in E$, we have $\mathrm{rank}_t(P) = \mathrm{rank}_t(P') \geq \mathrm{rank}_{t-1}(P')$, thus $\mathrm{tier}_t(w) \geq \mathrm{tier}_{t-1}(w)$.

Suppose $A \cap P'$ is non-empty. Consider the following diagrams where A_2 is the first connected component of $A \cap P'$. It might be a fragment of a path or a cycle. Let w' be the successor of w in path P' and set $e_w = (ww')$, we consider two cases:

- pairs w_1, w_1' and w_3, w_3' denote the case where the edge $e_w \notin A$,
- pair w_2, w_2' denotes the case where the edge $e_w \in A$,



By induction hypothesis we know that $\mathrm{tier}_t(w') \geq \mathrm{tier}_{t-1}(w')$. In the first case (vertices w_1, w_1' and w_3, w_3') we have

$$\mathrm{tier}_t(w) \geq \max\left\{\, \mathrm{rank}_t(e_w), \mathrm{tier}_t(w')\right\} \geq \max\left\{\, \mathrm{rank}_{t-1}(e_w), \mathrm{tier}_{t-1}(w')\right\} \geq \mathrm{tier}_{t-1}(w)$$

For the second case we only need to observe, that by the direction in which A goes we have $\mathrm{tier}_{t-1}(w') \geq \mathrm{tier}_{t-1}(w)$, and so

$$\mathrm{tier}_t(w) \geq \mathrm{tier}_t(w') \geq \mathrm{tier}_{t-1}(w') \geq \mathrm{tier}_{t-1}(w).$$

□

Corollary 3.5. For any edge $e = (uv) \in E_t$, we have $\mathrm{tier}_t(v) + \psi \geq \mathrm{rank}_t(e)$.

Proof. Let $\tilde{t}$ be the last time $\mathrm{rank}_\bullet(e)$ changed. We consider two cases, at step $\tilde{t}$ the edge e was used exactly once (1) or strictly more than once (2). It is worth noting, that this lemma works for newly added edges too, since their ranks in previous turns are by definition zero.

1. For the edge to be used, it had to be tiered and by the definition of tiered edge, $\mathrm{tier}_{\tilde{t}-1}(v) \geq \mathrm{rank}_{\tilde{t}-1}(e)$. However, by Lemma 3.4, the tier does not decrease, so
$$\mathrm{tier}_t(v) + \psi \geq \mathrm{tier}_t(v) + 1 \geq \mathrm{tier}_{\tilde{t}-1}(v) + 1 \geq \mathrm{rank}_{\tilde{t}-1}(e) + 1 \geq \mathrm{rank}_t(e).$$
2. In this case e had to be part of the cycle, so $\mathrm{tier}_{\tilde{t}-1}(u) = \mathrm{tier}_{\tilde{t}-1}(v)$. By the definition of tiered edge $\mathrm{tier}_{\tilde{t}-1}(u) \geq \mathrm{rank}_{\tilde{t}-1}(e)$, thus
$$\mathrm{tier}_t(v) + \psi \geq \mathrm{tier}_{\tilde{t}-1}(v) + \psi = \mathrm{tier}_{\tilde{t}-1}(u) + \psi \geq \mathrm{rank}_{\tilde{t}-1}(e) + \psi \geq \mathrm{rank}_t(e).$$

□

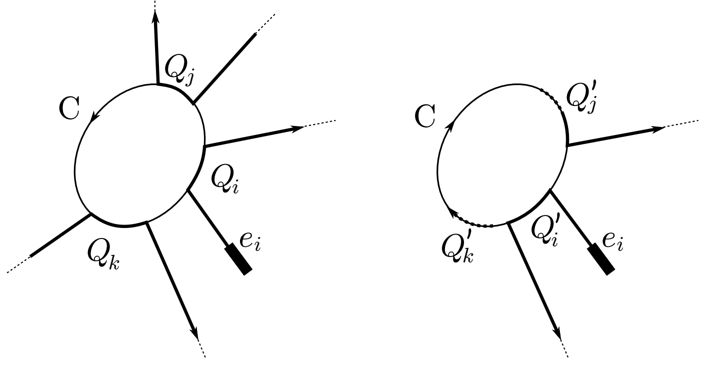
Corollary 3.6. Let P be any path that ends at a seed $w \xrightarrow{P} s$. Every edge $e \in P$ of rank $\mathrm{rank}_t(e) \geq \psi$ is followed by some edge e' of rank $\mathrm{rank}_t(e') \geq \mathrm{rank}_t(e) - \psi$.

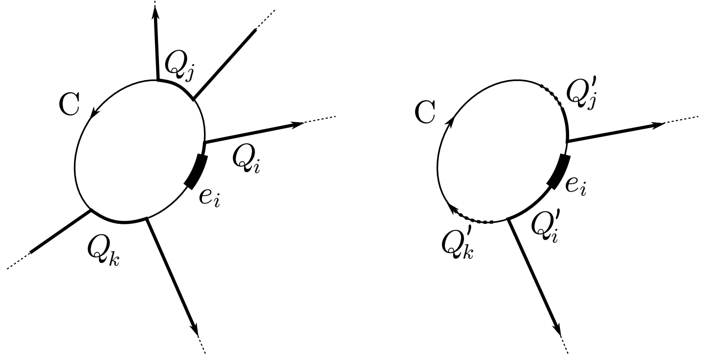
Proof. Let $e = (uv)$, by Corollary 3.5 $\mathrm{tier}_t(v) \geq \mathrm{rank}_t(e) - \psi$. By Lemma 3.3 any path from v to any sink contains an edge of rank at least $\mathrm{tier}_t(v)$. □

Corollary 3.7. The length of the shortest path from w to some seed s is at least $\frac{1}{\psi}\,\mathrm{tier}_t(w)$.

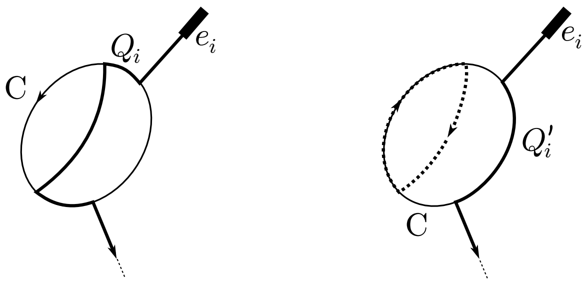
Lemma 3.8. *Let $\pi_2\big((u,v)\big) = v$ denote the end of a directed edge (uv). For a directed cycle $C \subseteq G_t$ we represent by $G_t \oplus C$ the graph that is the same as G_t, but the direction of edges of C is changed. Let $\{e_i\}_{i=1}^{l}$ be l distinct edges and $\pi_2(e_i) \xrightarrow{Q_i} s_i$ be vertex-disjoint paths in G_t. If $C \cap \{s_i\}_i = \varnothing$, then there exist l vertex-disjoint paths $\pi_2(e_i) \xrightarrow{Q_i'} s_{\sigma(i)}$ in $G_t \oplus C$, where σ is some permutation of seeds.*

Proof. For those paths Q_i such that $Q_i \cap C = \varnothing$ nothing happens, so $Q_i' = Q_i$. Otherwise, there two cases: $e_i \notin C$ or $e_i \in C$, the solutions for Q_i are presented below, where Q_k and Q_j are the closest left and right paths.





Such reconnections create valid paths, because the number of entry-points is the same as the number of exit-points. Moreover, the number of exit-points that lead back to cycle is the same as the number of entry-points that come from a cycle, so parts of old Q_i paths may create loops (dotted in picture below), but Q'_i will be still connected to seeds.



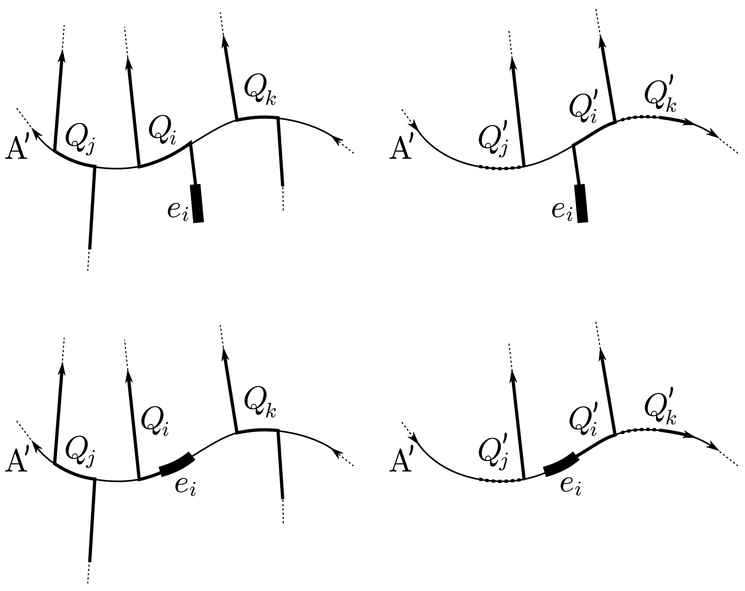
□

Lemma 3.9. *Let $Q_1, Q_2, \ldots, Q_l$ be a sequence of edge-disjoint seeded simple paths to disjoint seeds at step t. Also, let $w \xrightarrow{P} s$ be the augmenting path from turn t. Then, there exists edge-disjoint seeded paths $Q'_1, \ldots, Q'_l$ and P' in G_{t-1} such that*

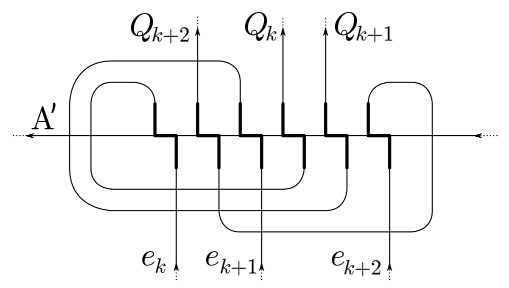
$$\begin{aligned}
\mathrm{rank}_{t-1}(P) &\leq \mathrm{rank}_{t-1}(P'), \\
\mathrm{rank}_t(Q_i) &\leq \mathrm{rank}_{t-1}(Q'_i) && \text{for } \mathrm{rank}_t(Q_i) > \mathrm{rank}_{t-1}(P) + \psi, \\
\mathrm{rank}_t(Q_i) &\leq \mathrm{rank}_{t-1}(Q'_i) + \psi && \text{otherwise.}
\end{aligned}$$

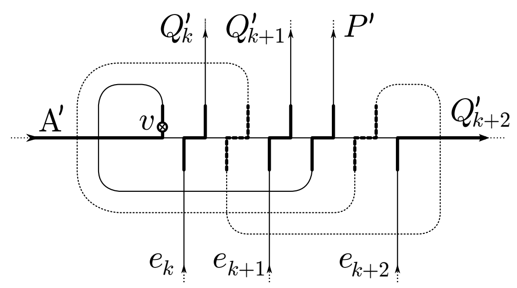
Proof. Let A be the set of edges that changed direction between turns $(t-1)$ and t, that is, A is a single simple path and a collection of cycles such that $A \subseteq P$. Also, we know that in any cycle C of augmenting path all the tiers are the same, moreover, any neighbour of C has tier at least as high. Therefore, if $e \in P$ is the most high-ranked edge of P at step $(t-1)$, there exist an edge e' such that $\mathrm{rank}_{t-1}(e') \geq \mathrm{rank}_{t-1}(e)$ and e' does not belong to any cycle of P, i.e. it belongs to the simple path part of P (naturally, it might happen that $e' = e$). Please observe, that cycles of A constitute a (possibly proper) subset of cycles of P, namely there might be a cycle of P consisting of edges that finally didn't change their orientation.

Set A' to be the path component of A. As the number of cycles in A is finite, using Lemma 3.8, we know there exist paths $Q''_1, \ldots, Q''_l$ in $G_t \oplus A'$ such that each Q''_i contains the highest-ranked edge e_i of Q_i. Note, that without loss of generality we can assume that both Q_i and Q''_i start with e_i. The only thing left to take care of is the acyclic A' and the existence of path P'. If $Q_i \cap A' = \varnothing$ edge-wise, then $Q'_i = Q''_i$ is a valid solution. It is worth emphasizing that G_t being a matching-directed bipartite graph implies that no edge-disjoint paths cross, i.e. they are vertex-disjoint except endpoints. Otherwise, similarly to Lemma 3.8 there are two cases: $e_i \notin A'$ and $e_i \in A'$. Both are presented below.

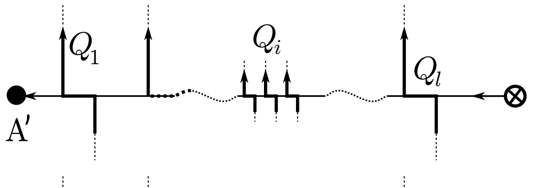


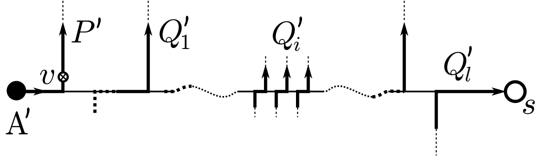
When some path Q_i intersects with A' multiple times, we regard each such part separately, and relink it as any other to its next neighbor. If we were to regard each seed as 1-sink, each edge e_i as 1-source, and each fragment as described in the previous sentence as a pair sink-source, then matching sources with sinks might create cycles, but any each edge e_i will find a way to some seed. The idea is shown in diagram below.





Special cases include the last Q_l (the closest to the seed) and Q_1 (the farthest one from the seed). The former just connects to the seed (the only available option), while the latter constitutes a new path. Let path P'' be the part of Q_1 after the A' that is not used by any of the other Q_is. Observe, that because A' is a tiered path, then $\max\{\mathrm{rank}_{t-1}(P_0), \mathrm{rank}_{t-1}(e), \mathrm{tier}_{t-1}(v)\} \geq \mathrm{rank}_{t-1}(P)$ where v is the first vertex of P'', e is the edge between A' and P'' and P_0 is the part of A' before any Q_is. Finally we set $P' = P_0 \cup P''$.



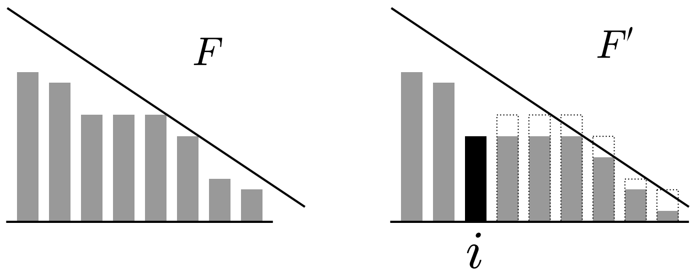


□

Lemma 3.10. *If F is a seeded forest at step t, then there exists at least as bad seeded forest F' at step $(t-1)$, that is* $\mathrm{rank}_t(F) \leq \mathrm{rank}_{t-1}(F')$.

Proof. We will prove that there exists forest F' such that all ranks of trees that changed were not bigger than the rank of tree of which the seed was matched at step $(t-1)$. The idea is pictured below, the bars represent the ranks of trees, the top line is angled at $\frac{-\psi \text{ rank}}{1 \text{ bar}}$, that is, ψ units of rank per each bar of the chart. With that in mind one can word the lemma informally as "during the algorithm run the ridge does not rise".

More verbosely, given that the trees are sorted descending by the ranks and the angled top line is put at the most protruding one, in spite of rising ranks between F' and F, the deleted tree causes a shift such that all the bars are, again, under the same line.



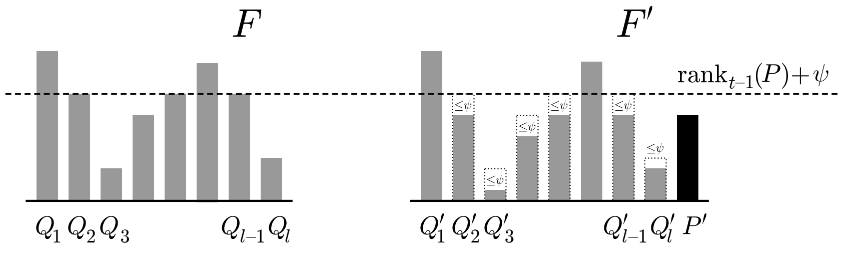
Let F be any seeded forest in G_t and $T_0, \ldots, T_{l-1}$ be the trees of F sorted descending by their rank. Recall that the $\operatorname{rank}_t(T) = \max_{e \in \overline{T}} \operatorname{rank}_t(e)$ is defined over the closure $\overline{T}$ of T rather than plain tree. We will show a seeded forest F' in G_{t-1} with $(l+1)$ trees $T'_0, \ldots, T'_l$ (mind that the cardinality of seeds changes by one each turn) such that

$$\begin{aligned}
&\operatorname{rank}_t(T_k) = \operatorname{rank}_{t-1}(T'_k) && \text{for } 0 \le k < i, \\
&\operatorname{rank}_t(T_k) \le \operatorname{rank}_{t-1}(T'_k) + \psi && \text{for } k = i, \text{ the new tree} \\
&\operatorname{rank}_t(T_k) \le \operatorname{rank}_{t-1}(T'_{k+1}) + \psi && \text{for } i \le k < l,
\end{aligned}$$

where

$$i = \min \left\{ k \in \{0, \ldots, l-1\} \;\middle|\; \operatorname{rank}_t(T_k) \ne \operatorname{rank}_{t-1}(T'_k) \right\}.$$

As definition requires, for any tree $T \in F$ of rank k there exists an edge $e = (u, v)$ of rank k such that $v \in T$. In other words, in forest F there are l vertex-disjoint paths that start with high-ranked edges and end at seeds. By Lemma 3.9 we know that there are analogous edge-disjoint paths in turn $(t-1)$ of ranks not higher by ψ if changed. Moreover, there exists additional path P' of rank of the augmenting path, also disjoint from all the others. Without the sorting, the above diagrams would look like below.



The augmenting path is a tiered path, so when it follows some edge for the first time, it could not have bigger rank and hence, $\operatorname{rank}_{t-1}(P') = \operatorname{rank}_{t-1}(T_i)$. This provides us with the skeleton on which we can build our trees T'. The rest of the edges and vertices of the trees is unimportant and can be filled arbitrarily. □

Corollary 3.11. Function Φ defined in (1) is a potential function.

Lemma 3.12. *If $R > 2\psi$ is the maximum rank of some tree T of some forest $F \in \mathcal{F}_t$ in turn t, then there exists step $\tilde{t} \le t$ such that the sum of ranks of all the trees of some forest $F \in \mathcal{F}_{\tilde{t}}$ is $\Omega(R^2 \frac{1}{\psi})$.*

Proof. Let F be the F_t, and $F_{t-1}, F_{t-2}, \ldots$ be forests in turns $(t-1), (t-2), \ldots$, constructed sequentially as specified in the proof of Lemma 3.10. Set $\sigma_k : F_k \to F_{k-1}$ be a correspondence between the trees of forest F_k and trees of forest F_{k-1} as in the aforementioned proof.

Moreover, let $T_k^{\circledast}$ be the new trees (indexed before by i) from forests F_k. In particular, $T_{k-1}^{\circledast}$ is the only tree that does not belong to the image of σ_k.

Define B_t as the singleton of T, that is, $B_t = \{T\}$ and the sequence $B_{t-1}, B_{t-2}, \ldots$ as follows

$$B_{k-1} = \begin{cases} \sigma_k(B_k) & \text{if } \forall \tau \in B_k.\, \operatorname{rank}_k(\tau) = \operatorname{rank}_{k-1}(\sigma_k(\tau)), \\ \sigma_k(B_k) \cup \{T_k^{\circledast}\} & \text{otherwise, i.e. if some tree of } B_k \text{ did change rank.} \end{cases}$$

Intuitively it denotes the trees that directly caused the rank of T to be R. By the nature of trees $T^{\circledast}$ we know that if a new tree is added to B_k, then its rank is at least as high as the minimum of the ranks of trees of B_k. Moreover, the size of B_k doesn't increase if the ranks of the trees do not change, therefore

$$\forall \tau \in B_k.\ \operatorname{rank}_k(\tau) \geq \operatorname{rank}_t(T) - \psi |B_k|.$$

Let $\tilde{t}$ be a step for which $|B_{\tilde{t}}| = \frac{\operatorname{rank}_t(T)}{2\psi}$. Then, each tree of $B_{\tilde{t}}$ has rank at least $\frac{1}{2}\operatorname{rank}_t(T)$, thus $\tilde{t} \geq \frac{1}{2\psi}\operatorname{rank}_t(T) > 0$ and the sum of ranks of all the trees of $B_{\tilde{t}}$ is

$$\sum_{\tau \in B_{\tilde{t}}} \operatorname{rank}_{\tilde{t}}(\tau) \geq \frac{1}{4\psi}\operatorname{rank}_t^2(T).$$

□

Proof of Theorem 3.1. Let R be the largest edge rank achieved during the algorithm run. Then, there was a tree of some forest that had rank R. By Lemma 3.12 there exists a turn t in which the sum of tree ranks is $\Omega(\frac{R^2}{\psi})$, and given the graph is bipartite, Corollary 3.7 implies there are vertex-disjoint paths of total length $\Omega(\frac{R^2}{\psi^2})$. However, there are n vertices, so $R = O(\frac{1}{\psi}\sqrt{n})$, and with $\psi = \Theta(1)$ we have $R = O(r)$. ■

4 Discussion and final remarks

Theorem 3.1 has some nice consequences. However, before we dig into them, it is worth noting that the whole proof could be rewritten using ranks of vertices instead, i.e. how many times each augmenting path has touched the node in question. The problem is, that we are unable to require for analogous ψ to be constant, on the other hand, it is true if we were to assume the augmenting paths to be simple. With such restriction the theorem yields even stronger bound of $O(nr)$ on the total length of all the augmenting paths. Unfortunately, finding such paths is costly and edge-ranking approach was more convenient.

This brings us to seed-searching algorithm. Observe, that the graph of tiered edges may be cyclic, yet no strongly connected components can cross the boundaries of tiers. This implies, that each time we arrive at previously visited vertex, we can back out the way we came, simulating a plain depth-first search on each tier. As long as we don't find a vertex of smaller tier, we can "run rampant" across the edges and we would construct a valid tiered augmenting path. However, should we find a vertex of a smaller tier, we can't go back, but this is also precisely what we were looking for. Repeating this process along with some simple algorithmic tricks to ensure constant time successor and edge processing gives an algorithm running in amortized $O(mr)$ time that maintains the maximum cardinality matching in one-sided on-line bipartite matching problem. In particular, it also solves the offline maximum cardinality matching problem in the same time as the well-known Hopcroft-Karp algorithm [5]. The described algorithm has been implemented and will be a subject of another article.

Another consequence of Theorem 3.1 are the constrains on the structure of graph inside the tiers. As the handling of edges amortizes itself during the algorithm run, the average depth of tiers has to be rather low. This is a property that urges a further investigation.

Finally, surely the methods used in this paper might be generalized. The first example that could befit from presented approach is the maximum unit-flow problem and, naturally, the weighted cases of bipartite matching problem.

References

[1] B. Bahmani and M. Kapralov, *Improved bounds for online stochastic matching*, in de Berg and Meyer [2], pp. 170–181.

[2] M. de Berg and U. Meyer, eds., *Algorithms - ESA 2010, 18th Annual European Symposium, Liverpool, UK, September 6-8, 2010. Proceedings, Part I*, vol. 6346 of Lecture Notes in Computer Science, Springer, 2010.

[3] J. Feldman, M. Henzinger, N. Korula, V. S. Mirrokni, and C. Stein, *Online stochastic packing applied to display ad allocation*, in de Berg and Meyer [2], pp. 182–194.

[4] J. Feldman, A. Mehta, V. S. Mirrokni, and S. Muthukrishnan, *Online stochastic matching: Beating 1-1/e*, in FOCS, IEEE Computer Society, 2009, pp. 117–126.

[5] J. E. Hopcroft and R. M. Karp, *An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs*, SIAM Journal on Computing, 2 (1973), pp. 225–231.

[6] P. Jaillet and X. Lu, *Online stochastic matching: New algorithms with better bounds*, (2013). To be published in *Mathematics of Operations Research*.

[7] R. M. Karp, U. V. Vazirani, and V. V. Vazirani, *An optimal algorithm for on-line bipartite matching*, in STOC, H. Ortiz, ed., ACM, 1990, pp. 352–358.

[8] V. H. Manshadi, S. O. Gharan, and A. Saberi, *Online stochastic matching: Online actions based on offline statistics*, in SODA, D. Randall, ed., SIAM, 2011, pp. 1285–1294.