Available online at www.sciencedirect.com

SCIENCE DIRECT®

Information Processing Letters 91 (2004) 271–276

Information Processing Letters

www.elsevier.com/locate/ipl

A simpler linear time $2/3 - \varepsilon$ approximation for maximum weight matching

Seth Pettie *, Peter Sanders

Max Planck Institut für Informatik, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany

Received 17 November 2003; received in revised form 10 May 2004

Available online 19 June 2004

Communicated by K. Iwama

Abstract

We present two $\frac{2}{3} - \varepsilon$ approximation algorithms for the maximum weight matching problem that run in time $O(m \log \frac{1}{\varepsilon})$. We give a simple and practical randomized algorithm and a somewhat more complicated deterministic algorithm. Both algorithms are exponentially faster in terms of ε than a recent algorithm by Drake and Hougardy. We also show that our algorithms can be generalized to find a $1 - \varepsilon$ approximation to the maximum weight matching, for any $\varepsilon > 0$.

Keywords: Matching; Maximum weight matching; Approximation; Analysis of algorithms

1. Introduction

Consider an undirected weighted graph $G = (V, E, w)$, where m and n are the number of edges and vertices, respectively, and $w(e)$ denotes the weight of edge $e \in E$. A *matching* is a set of edges $M \subseteq E$ that are endpoint disjoint from one another. The *maximum weight matching* problem (or MWM) is to find a matching M^* of maximum weight, where $w(M^*) \stackrel{\text{def}}{=} \sum_{e \in M^*} w(e)$. The fastest algorithms for solving this problem run in polynomial time: $O(mn + n^2 \log n)$ for real-weighted graphs [4] and $O(m\sqrt{n} \cdot \text{polylog}(nC))$ time [6] when the weights are integers less than C. Despite these nice polynomial-time solutions there is considerable interest in simpler and faster algorithms—ideally linear time—that return a solution of some guaranteed quality. For example, weighted matchings are a crucial subroutine for partitioning large networks like finite element meshes and VLSI circuits (see, e.g., [8]). We define the δ-MWM problem to be that of finding any matching whose weight is at least $\delta \cdot w(M^*)$.

There is a well known $\frac{1}{2}$-MWM algorithm based on a simple greedy strategy: scan the edges in increasing order of weight, selecting the current edge if it is vertex-disjoint from previously selected edges. This algorithm requires $O(m \log n)$ time.[1] Preis [8], and

* Corresponding author.
E-mail address: pettie@mpi-sb.mpg.de (S. Pettie).

[1] One can get a $(\frac{1}{2} - m^{-k})$-MWM algorithm in time $O(km)$ using base m radix sorting of weights rounded to multiples of $\max_{e \in E} w(e)/m^{k+1}$.

0020-0190/$ – see front matter
doi:10.1016/j.ipl.2004.05.007

later Drake and Hougardy [3], presented linear-time $\frac{1}{2}$-MWM algorithms.

While the $\frac{1}{2}$-MWM algorithms above only compare adjacent edges, it is possible to achieve better approximations by examining short augmenting paths. Drake and Hougardy [2] observed that if a matching M is such that any weight-augmenting path or cycle has more than 2 unmatched edges, then $w(M) \geqslant \frac{2}{3}w(M^*)$. In a subsequent paper [1] Drake and Hougardy developed a $(\frac{2}{3}-\varepsilon)$-MWM algorithm running in time $\mathrm{O}(m \cdot \varepsilon^{-1})$.

The Drake–Hougardy algorithm is somewhat complicated and requires a very detailed analysis. Moreover, it converges on a $\frac{2}{3}-\varepsilon$ solution very slowly. In this paper we give two simple $(\frac{2}{3}-\varepsilon)$-MWM algorithms, each running in $\mathrm{O}(m \log \frac{1}{\varepsilon})$ time. Our first algorithm is randomized and admits a simple analysis. It rivals all previous matching algorithms in terms of simplicity and promises to be a good choice in practice. Our deterministic algorithm is slightly more complicated and requires a more sophisticated analysis. Both algorithms converge on a $\frac{2}{3}-\varepsilon$ solution in exponentially fewer iterations than the Drake–Hougardy approach.

Although we can only obtain a linear running time for the $(\frac{2}{3}-\varepsilon)$-MWM problem, both our algorithms can be extended in purely mechanical ways to the δ-MWM problem, for any δ. For graphs with sufficiently low degree, our δ-MWM algorithms are faster than the $\mathrm{O}(m\sqrt{n} \cdot \log(n(1-\delta)^{-1}))$ algorithm of Gabow and Tarjan [6].[2] It is not clear to us whether the Drake–Hougardy approach can be easily extended to the δ-MWM problem, for $\delta > \frac{2}{3}$.

2. Terminology and notation

Most of our definitions are implicitly with respect to some matching called M, which in our algorithms is the matching currently under consideration. The maximum weight matching is M^*. A path or cycle is *alternating* if it consists of edges drawn alternately from M and $E \backslash M$. An alternating path or cycle P is an *augmentation* if $M \oplus P$ is also a matching, where $A \oplus B = (A \backslash B) \cup (B \backslash A)$. The *gain* of an alternating path/cycle P is $g(P) = w(P \backslash M) - w(P \cap M)$. The gain of a set of (not necessarily disjoint) paths/cycles is the sum of their individual gains. A *k-augmentation* is one containing at most k non-M edges.

It is well known that if a matching admits no positive-gain k-augmentations then it must have weight at least $k/(k+1)$ of the maximum. See [7] for the unweighted version of this theorem and [2] for the weighted version. Theorem 2.1 shows that any matching can be brought geometrically closer to a $k/(k+1)$-optimal one using *disjoint* k-augmentations.

Theorem 2.1. *For any matching M, there exists a collection A of vertex-disjoint k-augmentations such that*

$$w(M \oplus A) \geqslant w(M) + \frac{k+1}{2k+1} \times \left(\frac{k}{k+1} w(M^*) - w(M)\right).$$

Proof. The graph $C = M \oplus M^*$ consists of alternating paths and cycles w.r.t. M or M^*. We may assume w.l.o.g. that C is a single path/cycle; our argument is applied to each separately. If C is a (2ℓ)-cycle, list its edges in cyclic order: $e_0, e_0^*, e_1, e_1^*, \ldots, e_{\ell-1}, e_{\ell-1}^*$, where $e_i \in M$, $e_i^* \in M^*$. To conserve ink, let $k^+ = k+1$. Let A_i be the set of disjoint k-augmentations

$$\Big\{\{e_i, \ldots, e_{i+k^+-1}\}, \{e_{i+k^+}, \ldots, e_{i+2k^+-1}\}, \ldots, \{e_{i+k^+\lfloor(\ell-k^+)/k^+\rfloor}, \ldots, e_{i+k^+\lfloor(\ell-k^+)/k^+\rfloor+k^+-1}\}\Big\}.$$

That is, the augmentations in A_i are disjoint and the only M-edges not in A_i are the $(\ell \bmod k^+)$ ones at the end of the list, when starting at e_i. We wish to lower bound the gain of the best set of augmentations. Clearly $\max_i g(A_i) \geqslant \sum_i g(A_i)/\ell$. One can easily see that in $\sum_i g(A_i)$, each M^*-edge is counted $k\lfloor \ell/k^+ \rfloor$ times, and each M-edge $k^+\lfloor \ell/k^+ \rfloor$ times. Therefore,

$$\begin{aligned}\sum_{i=0}^{\ell-1} g(A_i)/\ell &= \left[k\left\lfloor \frac{\ell}{k^+}\right\rfloor w(M^*) - k^+\left\lfloor \frac{\ell}{k^+}\right\rfloor w(M)\right]\Big/\ell \\ &= \frac{k^+}{\ell}\left\lfloor \frac{\ell}{k^+}\right\rfloor\left(\frac{k}{k^+}w(M^*) - w(M)\right) \\ &\geqslant \frac{k^+}{2k^+-1}\left(\frac{k}{k^+}\,w(M^*) - w(M)\right)\end{aligned}$$

[2] As Gabow and Tarjan [5,6] note, their weighted matching algorithm can be viewed either as an exact algorithm for integer-weighted graphs or as an approximation algorithm for arbitrary graphs.

$$= \frac{k+1}{2k+1}\left(\frac{k}{k+1}\,w(M^*) - w(M)\right).$$

If C is a path, list the edges as before. Let $M_i^* = \{e_j^*\colon\ j = i \pmod{k+1}\}$ and $A_i = C \backslash M_i^*$. A_i consists of disjoint k-augmentations and $\sum_{i=0}^{k} g(A_i) = k\,w(M^*) - (k+1)\,w(M)$. Thus, for at least one i:

$$w(M \oplus A_i) \geqslant \frac{k}{k+1} w(M^*). \quad \square$$

Before moving on we give a little more notation used in both our matching algorithms. If v is matched in M let $M(v) = u$ where $(v, u) \in M$; otherwise $M(v) = v$. A 2-augmentation is *centered* at vertex v if all its non-M edges are incident to v or $M(v)$. We may also say the augmentation is centered at the edge $(v, M(v))$. Note that every 2-augmentation has at least two center vertices. Let $A_{\frac{2}{3}}$ be a set of vertex-disjoint 2-augmentations such that $g(A_{\frac{2}{3}}) \geqslant \frac{3}{5}(\frac{2}{3}w(M^*) - w(M))$; Theorem 2.1 implies that $A_{\frac{2}{3}}$ exists. Let $\mathrm{aug}^*(v)$ be the 2-augmentation in $A_{\frac{2}{3}}$ centered at v (if any) and let $\mathrm{aug}(v)$ be the maximum-gain 2-augmentation centered at v.

3. A randomized matching algorithm

Our randomized matching algorithm can be described very succinctly. Choose a random vertex v and augment the current matching with the highest-gain 2-augmentation centered at v. Repeat as many times as you wish. See Fig. 1 for a more formal description.

We first examine the expected time of Steps 2–3. Let $\deg(v)$ denote the degree of v in G; w.l.o.g. assume deg is strictly positive.

Lemma 3.1. *The time required to find* $\mathrm{aug}(v)$ *is* $\mathrm{O}(\deg(v) + \deg(M(v)))$.

```
0. M := ∅   (or initialize M to any matching)
1. repeat k times:
2.     Let X ∈ V(G) be selected uniformly at random
3.     M := M ⊕ aug(X)
4. return M
```

Fig. 1. Algorithm Random-Match (G, k): G is a graph, k is an integer.

Proof. If v is an isolated vertex in M, i.e., if $M(v) = v$, then finding $\mathrm{aug}(v)$ is trivially accomplished in $\deg(v)$ time. To find the alternating 4-cycles centered at v we first mark all vertices u s.t. $(v, u) \in E\backslash M$. For each edge $(M(v), x)$, if $M(x)$ is marked then $\langle v, M(v), x, M(x), v\rangle$ is an alternating 4-cycle. This procedure clearly runs in $\mathrm{O}(\deg(v) + \deg(M(v)))$ time.

The procedure for alternating paths is slightly more complicated. An *arm* of v consists of an edge $(v, u) \notin M$ plus $(u, M(u)) \in M$, if it exists. We find the two highest-gain arms of v, P and P', where $g(P) \geqslant g(P')$. For each arm Q of $M(v)$ we determine the highest-gain 2-augmenting path centered at v that uses Q. This will be $P \cup \{(v, M(v))\} \cup Q$ if Q and P are vertex disjoint and $P' \cup \{(v, M(v))\} \cup Q$ otherwise. Again, this procedure clearly takes $\mathrm{O}(\deg(v) + \deg(M(v)))$ time and detects the best 2-augmenting path centered at v. $\square$

Lemma 3.1 is essentially the same as Theorem 2 in [2]. We now examine the expected performance of Random-Match.

Lemma 3.2. *If* $v \in V$ *is chosen uniformly at random then*

$$\mathbb{E}\big[g\big(\mathrm{aug}(v)\big)\big] \geqslant \frac{6}{5n}\left(\frac{2}{3}w(M^*) - w(M)\right).$$

Proof. Let $V_{\frac{2}{3}}$ be the set of center vertices for the augmenting paths/cycles in $A_{\frac{2}{3}}$. Note that $|V_{\frac{2}{3}}| \geqslant 2 \cdot |A_{\frac{2}{3}}|$ since every 2-augmentation has at least two centers.

$$\begin{aligned}\mathbb{E}\big[g\big(\mathrm{aug}(v)\big)\big] &\geqslant \Pr[v \in V_{\frac{2}{3}}] \cdot \mathbb{E}\big[g\big(\mathrm{aug}(v)\big) \mid v \in V_{\frac{2}{3}}\big] \\ &\geqslant \sum_{v \in V_{\frac{2}{3}}} \frac{g(\mathrm{aug}^*(v))}{n} \\ &\geqslant \frac{6}{5n}\left(\frac{2}{3}w(M^*) - w(M)\right). \quad \square\end{aligned}$$

Lemma 3.3, given below, shows that by repeating the randomized augmentation step n times we obtain an expected geometric decrease in the gap between $w(M)$ and $\frac{2}{3}w(M^*)$.

Lemma 3.3. *The expected weight of M after k iterations of Steps 2–3 is at least $\frac{2}{3}w(M^*)(1 - \mathrm{e}^{-6k/5n})$.*

Proof. Let $\widetilde{w} = \frac{2}{3}w(M^*)$ and let Y_i be the weight of M after i iterations. Clearly $Y_0 = 0$ and by Lemma 3.2, $\mathbb{E}[Y_{i+1}] \geqslant Y_i + \frac{6}{5}(\widetilde{w} - Y_i)/n$. By linearity of expectation we have the more usable inequality $\mathbb{E}[Y_{i+1}] \geqslant \mathbb{E}[Y_i] + \frac{6}{5}(\widetilde{w} - \mathbb{E}[Y_i])/n$. Assuming inductively that $\mathbb{E}[Y_i] \geqslant \widetilde{w} \cdot (1 - \mathrm{e}^{-6i/5n})$ (it holds for $i = 0$), we have:

$$\begin{aligned}\mathbb{E}[Y_{i+1}] &\geqslant \widetilde{w} \cdot \left(1 - \mathrm{e}^{-6i/5n}\right) + 6\widetilde{w} \cdot \mathrm{e}^{-6i/5n}/5n \\ &= \widetilde{w} \cdot \left(1 - (1 - 6/5n)\,\mathrm{e}^{-6i/5n}\right) \\ &\geqslant \widetilde{w} \cdot \left(1 - \mathrm{e}^{-6(i+1)/5n}\right). \quad \square\end{aligned}$$

Theorem 3.4. *In expected time* $\mathrm{O}(m \log \frac{1}{\varepsilon})$ *Random-Match returns a matching whose expected weight is at least $\frac{2}{3} - \varepsilon$ that of the maximum weight matching.*

Proof. The Theorem follows by setting $k = \frac{5}{6}n \ln \frac{1}{\varepsilon}$. By Lemma 3.3 the expected weight of the returned matching is $\frac{2}{3}(1 - \mathrm{e}^{-6k/5n}) > \frac{2}{3} - \varepsilon$ times the optimum. By Lemma 3.1 the time for Steps 2 and 3 is proportional to $\deg(X) + \deg(M(X))$. Averaged over all $X \in V(G)$ the expected time per iteration is $4m/n$. $\square$

Remark. Rather than appealing to Theorem 2.1 to prove Lemma 3.2, one can show directly that $\mathbb{E}[\mathrm{aug}(v)] \geqslant 3(\frac{2}{3}w(M^*) - w(M))/n$. This implies that only $\frac{1}{3}n \ln \frac{1}{\varepsilon}$ iterations of Random-Match suffice.

4. A deterministic algorithm

The deterministic algorithm operates in phases, each taking linear time. If M and M' are the matchings before and after some phase we guarantee that $w(M') \geqslant w(M) + c \cdot (\frac{2}{3}w(M^*) - w(M))$, for a constant c. Therefore, executing $\mathrm{O}(\log \frac{1}{\varepsilon})$ phases yields a $\frac{2}{3} - \varepsilon$ matching.

In each phase we generate a set of at most n candidate augmentations (one centered at each vertex) then choose from this set, in a greedy manner, a subset of non-overlapping augmentations. In the analysis we show that the total gain of the augmentations selected is $\Omega(g(A_{\frac{2}{3}})) = \Omega(\frac{2}{3}w(M^*) - w(M))$. The obvious ways to choose the candidate set can perform very poorly in the worst case. For instance, if we choose $\{\mathrm{aug}(v){:}\ v \in V(G)\}$, or in general the best k augmentations centered at each vertex, it is impossible to guarantee a gain of $\Omega(g(A_{\frac{2}{3}}))$.

Recall that nearly all definitions are with respect to the matching M. An *atom* is either a matched edge or an unmatched vertex. We will think of augmentations as either being sets of atoms or sets of edges, whichever is more convenient. If e is an atom and S a set of augmentations then $S(e)$ is the maximum-gain augmentation in S that contains e, if any. If $a = \{e_1, e_2, \ldots\}$ is a set of atoms (e.g., an augmentation) then $S(a) = \{S(e_1), S(e_2), \ldots\}$. An arm[3] r of v is *eligible* if $g(r) \geqslant \gamma \cdot g(S(r))$, for a constant $\gamma > 1$ to be specified later. An augmentation a is eligible if $g(a) \geqslant \gamma \cdot g(S(a))$ and, if a is a three-atom augmentation centered at edge $(u, M(u))$, both the arms of u and $M(u)$ are also eligible.

We denote by $\mathrm{greedy}(S)$ a subset of the augmentations S selected by the greedy algorithm. Specifically the algorithm repeatedly selects the maximum-gain augmentation in S that is vertex/atom disjoint with previously chosen augmentations.

Theorem 4.1. *Deterministic-Match* (*Fig.* 2) *runs in* $\mathrm{O}(km)$ *time and returns a matching weighing at least* $\frac{2}{3}(1 - (\frac{19}{20})^k)$ *of the maximum weight matching.*

Proof. Let $a(e)$ be the augmentation centered at e that is selected in Line 4 (if any), and let $S(a(e))$ and $S(\mathrm{aug}^*(e))$ be w.r.t. the set S when e is considered in Line 4. In isolation S refers to the set S after the phase,

0. $M := \emptyset$
1. **repeat** k times: (*Lines* 2–6 = 1 *Phase*)
2. $\quad S := \emptyset$
3. $\quad$ **foreach** atom e (*Either* $e \in M$ or $e \in V \setminus \bigcup_{c \in M} c$)
4. $\quad\quad$ Find an eligible augm. a centered at e maximizing $g(a)$
5. $\quad\quad$ $S := S \cup \{a\}$
6. $\quad$ $M := M \oplus \mathrm{greedy}(S)$
7. **return** M

Fig. 2. Algorithm Deterministic-Match (G, k): G is a graph, k an integer.

[3] Recall that an arm consists of an unmatched edge (v, u) plus the matched edge $(u, M(u))$ if it exists.

at Line 6. We will prove that after every phase of the algorithm the following two inequalities hold.

$$g(S) \geqslant g(A_{\frac{2}{3}})/3\gamma, \tag{1}$$

$$g\big(\text{greedy}(S)\big) \geqslant (\gamma - 1)g(S)/\gamma. \tag{2}$$

We obtain the sharpest bound by setting $\gamma = 2$, giving:

$$\begin{aligned} g\big(\text{greedy}(S)\big) &\geqslant (\gamma - 1)g(A_{\frac{2}{3}})/3\gamma^2 \\ &\geqslant \frac{1}{20}\left(\frac{2}{3}w(M^*) - w(M)\right). \end{aligned}$$

We now consider (1). If

$$g\big(\text{aug}^*(e)\big) < \gamma \cdot g\big(S\big(\text{aug}^*(e)\big)\big)$$

then $\text{aug}^*(e)$ was ineligible when it was considered at Line 4. If

$$g\big(\text{aug}^*(e)\big) \geqslant \gamma \cdot g\big(S\big(\text{aug}^*(e)\big)\big)$$

and $\text{aug}^*(e)$ was eligible, then $g(\text{aug}^*(e)) \leqslant g(a(e))$. There is only one more case, when $\text{aug}^*(e)$ is an ineligible three-atom augmentation with $\text{aug}^*(e) \geqslant \gamma \cdot g(S(\text{aug}^*(e)))$. Let $\text{aug}^*(e) = \{r_1, e, r_2\}$, where r_1, r_2 are arms. One can see that $\{r_1, e\}$ must be an eligible augmentation and r_2 an ineligible arm (or the reverse), implying that $g(\{r_1, e\}) \leqslant g(a(e))$ and therefore that $g(\{r_1, e, r_2\}) = g(\text{aug}^*(e)) < g(a(e)) + \gamma \cdot g(S(r_2))$. Combining all cases we have:

$$g\big(\text{aug}^*(e)\big) \leqslant \gamma \cdot g\big(S\big(\text{aug}^*(e)\big)\big) + g\big(a(e)\big). \tag{3}$$

Notice that $a(e)$ and each element of $S(\text{aug}^*(e))$ share at least one atom with $\text{aug}^*(e)$. Let C be a set of centers representing the augmentations in $A_{\frac{2}{3}}$, with $|C| = |A_{\frac{2}{3}}|$. Each augmentation in S can appear on the right side of (3) for at most three distinct $e \in C$ since all augmentations in $A_{\frac{2}{3}}$ are atom-disjoint. Summing (3) over $e \in C$ we have:

$$\begin{aligned} g(A_{\frac{2}{3}}) &= \sum_{e \in C} g\big(\text{aug}^*(e)\big) \\ &\leqslant \sum_{e \in C} \big[\gamma \cdot g\big(S\big(\text{aug}^*(e)\big)\big) + g\big(a(e)\big)\big] \\ &\leqslant 3\gamma \cdot g(S). \end{aligned}$$

Before turning to (2) we make a few observations. If an augmentation $a(e)$ is added to S in Line 5 we will say $a(e)$ *supersedes* each element of $S(a(e))$. We claim that if a_1 and a_2 both supersede a_3, where $a_1, a_2, a_3 \in S$, then $a_1 \cap a_2 \cap a_3 = \emptyset$. Suppose that a_1 was added to S before a_2. Because $a_3 \in S(a_2)$, a_3 must be the maximum-gain augmentation already in S that intersects a_2. Let $e \in a_2 \cap a_3$. Since $g(a_1) > g(a_3)$, it must be that $e \notin a_1$. This observation implies that in the acyclic graph $\mathcal{S} = (S, \{(a, a'){:}\ a \text{ supersedes } a'\})$, the subset $\{a{:}\ e \in a \in S\}$ forms a directed path, for any atom e. Moreover, the in and out degree of vertices in $\mathcal{S}$ are both bounded by 3.

Suppose that $a \in S$ was selected by greedy(S). This removes from consideration any other augmentation b for which $a \cap b \neq \emptyset$. Let $A_0 = \{a\}$ and let A_i be those augmentations in S superseded by some augmentation in A_{i-1}. Finally let $A = \bigcup_i A_i$. It follows from the observations above that A are exactly those augmentations removed from consideration by the selection of a. By the definition of eligibility $g(A_i) \leqslant g(A_{i-1})/\gamma$. Therefore:

$$\begin{aligned} g(A) &\leqslant \sum_{i=0}^{\infty} g(A_i) \leqslant g(a) \cdot \sum_{i=0}^{\infty} \gamma^{-i} \\ &= \frac{\gamma}{\gamma - 1} \cdot g(a). \end{aligned} \tag{4}$$

Summing over all $a \in \text{greedy}(S)$ we have

$$g\big(\text{greedy}(S)\big) \geqslant (\gamma - 1)\gamma^{-1} \cdot g(S),$$

which proves (2).

One can readily see that $\mathrm{O}(m)$ time suffices to compute the set S in each phase; see Lemma 3.1 for the details. The set greedy(S) can be computed in $\mathrm{O}(n)$ time by performing a topological sort of the acyclic graph $\mathcal{S}$. □

5. Arbitrarily good approximations

Both our randomized and deterministic algorithms can be generalized in straightforward ways to yield δ-MWM algorithms, for any $\delta < 1$. For $\delta \geqslant 2/3 - \mathrm{o}(1)$ the running time is super-linear; however, for degree bounded graphs and any constant δ the running time remains linear. In general graphs our algorithms are faster than the previous best [6] for sparse to moderately dense graphs.

Theorem 5.1. *There is a* $(1 - 1/k - \varepsilon)$*-MWM algorithm running in time* $\mathrm{O}(m(\Delta - 1)^{k-3} \log \varepsilon^{-1})$*, where*

$\varepsilon > 0$, $k \geqslant 3$, $\Delta > 1$ *is the maximum degree, and* m *the number of edges.*

The time bound follows from a generalized version of Lemma 3.1. One can easily show that the best $(k-1)$-augmentation centered[4] at a vertex v can be found in $\mathrm{O}((\deg(v) + \deg(M(v)))(\Delta - 1)^{k-3})$ time. For regular graphs the time bound of Theorem 5.1 improves on [5,6] for all k, whenever $\Delta = \mathrm{O}(n^{1/2(k-3)})$.

Acknowledgement

We would like to thank Stefan Hougardy for pointing out an error in an earlier version of Theorem 2.1.

References

[1] D. Drake, S. Hougardy, Improved linear time approximation algorithms for weighted matchings, in: 7th International Workshop on Randomization and Approximation Techniques in Computer Science (APPROX), in: Lecture Notes in Comput. Sci., vol. 2764, Springer-Verlag, Berlin, 2003, pp. 14–23.

[2] D. Drake, S. Hougardy, Linear time local improvements for weighted matchings in graphs, in: International Workshop on Experimental and Efficient Algorithms (WEA), in: Lecture Notes in Comput. Sci., vol. 2647, Springer-Verlag, Berlin, 2003, pp. 107–119.

[3] D. Drake, S. Hougardy, A simple approximation algorithm for the weighted matching problem, Inform. Process. Lett. 85 (2003) 211–213.

[4] H.N. Gabow, Data structures for weighted matching and nearest common ancestors with linking, in: First Annual ACM–SIAM Symposium on Discrete Algorithms (SODA), 1990, pp. 434–443.

[5] H.N. Gabow, R.E. Tarjan, Faster scaling algorithms for network problems, SIAM J. Comput. 18 (5) (1989) 1013–1036.

[6] H.N. Gabow, R.E. Tarjan, Faster scaling algorithms for general graph-matching problems, J. ACM 38 (4) (1991) 815–853.

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

[8] R. Preis, Linear time 1/2-approximation algorithm for maximum weighted matching in general graphs, in: Proc. 16th Ann. Symp. on Theoretical Aspects of Computer Science (STACS), in: Lecture Notes in Comput. Sci., vol. 1563, Springer-Verlag, Berlin, 1999, pp. 259–269.

[4] An augmenting cycle is centered at any of its atoms, an augmenting path at the second atom in the path.