Improved Linear Time Approximation Algorithms for Weighted Matchings*

Doratha E. Drake and Stefan Hougardy

Institut für Informatik, Humboldt-Universität zu Berlin, 10099 Berlin, Germany
{drake,hougardy}@informatik.hu-berlin.de

Abstract. The weighted matching problem is to find a matching in a weighted graph that has maximum weight. The fastest known algorithm for this problem has running time $O(nm + n^2 \log n)$. Many real world problems require graphs of such large size that this running time is too costly. We present a linear time approximation algorithm for the weighted matching problem with a performance ratio of $\frac{2}{3} - \varepsilon$. This improves the previously best performance ratio of $\frac{1}{2}$.

1 Introduction

A *matching* M in a graph $G = (V, E)$ is a subset of the edges of G such that no two edges in M are adjacent. In a graph $G = (V, E)$ with edge weights given by a function $w : E \to \mathbb{R}_+$ the *weight of a matching* is defined as $w(M) := \sum_{e \in M} w(e)$. The weighted matching problem is to find a matching M in G that has maximum weight. The first polynomial time algorithm for the weighted matching problem is due to Edmonds [4]. A straightforward implementation of this algorithm requires a running time of $O(n^2 m)$, where n and m denote the number of vertices and edges in the graph. Lawler [8] and Gabow [6] improved the running time to $O(n^3)$. The fastest known algorithm to date for solving the weighted matching problem in general graphs is due to Gabow [7] and has a running time of $O(nm + n^2 \log n)$.

Many real world problems require graphs of such large size that the runtime of Gabow's algorithm is too costly. Examples of such problems are the refinement of FEM nets [9], the partitioning problem in VLSI-Design [10], and the gossiping problem in telecommunications [2]. There also exist applications were the weighted matching problem has to be solved extremely often on only moderately large graphs. An example of such an application is the virtual screening of protein databases containing the three dimensional structure of the proteins [5]. The graphs appearing in such applications only have about 10,000 edges. But the weighted matching problem has to be solved more than 100,000,000 times for a complete database scan.

Therefore, there is considerable interest in approximation algorithms for the weighted matching problem that are very fast, have ideally linear runtime, and that nevertheless produce very good results even if these results are not optimal.

* supported by DFG research grant 296/6-3

The quality of an approximation algorithm for solving the weighted matching problem is measured by its so-called *performance ratio*. An approximation algorithm has a performance ratio of c, if for all graphs it finds a matching with a weight of at least c times the weight of an optimal solution.

A simple approximation algorithm for the weighted matching problem with performance ratio $\frac{1}{2}$ is obtained by the following greedy approach [1]: Start with an empty matching and extend it in each step by the heaviest edge currently available. The running time of this algorithm is $O(m \log n)$ as it requires sorting the edges by decreasing weight. The first linear time $\frac{1}{2}$-approximation algorithm for the weighted matching problem was proposed by Preis [11] using the idea of locally heaviest edges. Drake and Hougardy [3] obtained a simpler linear time approximation algorithm with the same performance ratio by using a completely different approach. In this paper we improve these results by proving the existence of linear time approximation algorithms for the weighted matching problem which have approximation ratios arbitrarily close to $\frac{2}{3}$.

Main Theorem. *For each $\varepsilon > 0$ there exists a linear time approximation algorithm for the weighted matching problem with a performance ratio of $\frac{2}{3} - \varepsilon$.*

The main idea of our algorithm is to start with a maximal matching M and to increase its weight by local changes. These local changes which we call short augmentations add in each step at most two new edges to M while up to four edges of M will be removed. A graph can possess up to $\Omega(n^4)$ short augmentations. To achieve linear running time only some part of these can be looked at. For each edge of the maximal matching M our algorithm only looks at all short augmentations that involve the endpoints of this edge. This way the short augmentations considered by the algorithm are in some sense spread evenly over the graph and their number is linearly bounded.

As the short augmentations are partly overlapping it can happen that after performing one short augmentation several others are no longer available. For the performance ratio it is therefore important to be able to reject short augmentations that achieve only minor improvements in the weight of the matching. This is achieved by only taking short augmentations into considerations that gain at least some constant factor β and that additionally yield the largest possible gain from all these. Such augmentations will be called β-augmentations. In linear time it seems not to be possible to find the best β-augmentation. However we will show that in linear time a constant factor approximation of the best β-augmentation can be found.

To prove the performance ratio of our algorithm we use an amortized analysis. The idea is that the gain that is achieved by an augmentation is not realized immediately but part of it is stored in certain edges of the graph for later use. This way we are able to prove that the algorithm increases the weight of the given matching by some constant factor. By repeating the algorithm a constant number of times and choosing β sufficiently small the resulting matching will have a weight that comes arbitrarily close to $\frac{2}{3}$.

The paper is organized as follows. In Section 2 we give basic definitions. In Section 3 we define short augmentations and use these to prove the existence of the set of local improvements upon which our algorithm is based. In Section 4 we present the algorithm and prove that its performance ratio is $\frac{2}{3} - \varepsilon$ for any $\varepsilon > 0$.

2 Preliminaries

Let $G = (V, E)$ be a weighted graph with weight function $w : E \to \mathbb{R}_+$. For a subset $F \subseteq E$ the weight of F is defined as $w(F) := \sum_{f \in F} w(f)$. A matching $M \subseteq E$ is called *maximal* if no proper superset of M in E is a matching. By M_{opt} we denote a maximum weight matching in G, i.e. a matching that satisfies $w(M_{opt}) \geq w(M)$ for all other matchings M. A path or cycle is called *M-alternating* if it uses alternately edges from M and $E \setminus M$. Note that alternating cycles must contain an even number of edges. Let P be an alternating path such that if it ends in an edge not belonging to M then this endpoint of P is not covered by an edge of M. The path P is called *M-weight-augmenting* if

$$w\left(E(P) \cap M\right) \;<\; w\left(E(P) \setminus M\right) \quad .$$

If P is an M-weight-augmenting path then $M \triangle P$ (the symmetric difference between M and P) is again a matching with strictly larger weight than M. The notion of M-weight-augmenting cycles is defined similarly. More generally we call an *augmentation* any process that replaces some edges of a matching M by some new edges and increases the weight of the matching.

3 Short Augmentations

A weight-augmenting path or cycle with respect to a matching M is called *short* if it contains at most two edges not belonging to M. The only weight-augmenting short cycle is by this definition an alternating cycle of length four and there exist six different types of weight-augmenting short paths. The following result shows that it is indeed enough to consider such short augmenting paths and cycles to obtain a $\frac{2}{3}$-approximation of the maximum weight matching.

Lemma 1. *For any matching M there exists a node disjoint set of weight-augmenting short paths and cycles such that augmenting along all these paths and cycles results in a matching of weight at least $\frac{2}{3} \cdot w(M_{opt})$.*

Proof. Consider the symmetric difference $M \triangle M_{opt}$. It consists of even length alternating cycles and of alternating paths. Order these paths and cycles arbitrarily and number the edges of M_{opt} in the order in which they appear in these paths and cycles. Now partition M_{opt} into three sets by taking the edge numbers modulo 3. By removing any of these

three sets from $M \triangle M_{opt}$ one obtains a set of alternating paths and cycles each of which contains at most two edges of M_{opt}. Removing the lightest of these three sets shows that M can be augmented to a matching of weight at least $\frac{2}{3} \cdot w(M_{opt})$ by paths and cycles each of which contain at most two edges not in M. □

In the following we need the notion of a *β-augmentation*. For a constant $\beta > 1$ a β-augmentation of a matching M is an augmentation that has the property that the weight of the edges that are removed from M is at least by the factor β smaller than the weight of the edges that are added to M by the augmentation. The following result shows that for small enough β any matching M can be augmented by short paths and cycles each of which is a β-augmentation to a matching that has a weight close to $\frac{2}{3} \cdot w(M_{opt})$.

Lemma 2. *Let M be an arbitrary matching and $\beta > 1$ be constant. Then there exists a node disjoint set of weight-augmenting short paths and cycles each of which is a β-augmentation such that augmenting along all these paths and cycles results in a matching of weight at least $\frac{2}{3\beta} \cdot w(M_{opt})$.*

Proof. By Lemma 1 we know that there exists a node disjoint set of augmenting paths and cycles each of which contains at most two edges not in M such that augmenting along all these paths and cycles results in a matching $\tilde{M}$ of weight at least $\frac{2}{3} \cdot w(M_{opt})$. We now claim that if we take the subset of these augmenting paths and cycles that are β-augmentations, we get a matching of the desired weight.

Partition the set $\tilde{M}$ into two sets $\tilde{M}_{\geq\beta}$ and $\tilde{M}_{<\beta}$ such that $\tilde{M}_{\geq\beta}$ contains all edges of $\tilde{M}$ that are obtained by β-augmentations and let $\tilde{M}_{<\beta}$ be all other edges of $\tilde{M}$. The set M similarly can also be partitioned into two sets $M_{<\beta}$ and $M_{\geq\beta}$ according to the augmenting paths and cycles in $M \triangle \tilde{M}$ that contain these edges. By performing only the β-augmentations one obtains the matching $M_{<\beta} \cup \tilde{M}_{\geq\beta}$. The weight of this set can be bounded from below as follows:

$$w(M_{<\beta}) + w(\tilde{M}_{\geq\beta}) \;\geq\; \frac{1}{\beta} w(\tilde{M}_{<\beta}) + w(\tilde{M}_{\geq\beta}) \;\geq\; \frac{1}{\beta} w(\tilde{M}) \;\geq\; \frac{2}{3\beta} w(M_{opt}) \;.$$

□

Let M be an arbitrary matching and let $\tilde{M}$ be a matching of weight at least $\frac{2}{3} \cdot w(M_{opt})$ such that the symmetric difference of M and $\tilde{M}$ consists of weight-augmenting short paths and cycles. The existence of $\tilde{M}$ is guaranteed by Lemma 1. For each cycle or path in $M \triangle \tilde{M}$ choose an edge in M that is adjacent to all edges of $\tilde{M}$ in this path or cycle. Call the set of all these chosen edges M^*. For each edge $e \in M^*$ denote by S_e the (at most two) edges of $\tilde{M}$ that are adjacent to e. For an arbitrary set F of edges denote by $inc(F)$ all edges in M that are incident to the endpoints of edges in F. Then $inc(S_e)$ contains at most three edges of M and $S_e \cup inc(S_e)$ is the set of edges of the path or cycle in $M \triangle \tilde{M}$ that contains the edge e.

For a given constant $\beta > 1$ we partition the set M^* into two subsets $M^*_{<\beta}$ and $M^*_{\geq\beta}$ such that $M^*_{<\beta}$ contains all edges of M^* such that $w(S_e) < \beta \cdot w(inc(S_e))$ and $M^*_{\geq\beta}$ contains all other edges of M^*.

The following result shows that if an algorithm achieves at least a constant fraction of the value $\frac{1}{\beta} \cdot w(S_e) - w(inc(S_e))$ for all $e \in M^*_{\geq\beta}$ then it will improve a given matching by a constant factor.

Lemma 3. *Let M be a matching of weight $w(M) \geq \alpha \cdot w(M_{opt})$. If the matching M' has a weight that is larger than the weight of M by at least $\varepsilon \cdot \sum_{e \in M^*_{\geq\beta}} \left(\frac{1}{\beta} \cdot w(S_e) - w(inc(S_e))\right)$ then*

$$w(M') \;\geq\; \left(\alpha + \varepsilon \cdot \left(\frac{2}{3\beta} - \alpha\right)\right) \cdot w(M_{opt}) \;.$$

Proof. By the definition of $M^*_{<\beta}$ we have $w(inc(S_e)) > \frac{1}{\beta} \cdot w(S_e)$ for $e \in M^*_{<\beta}$ and $w(M) = \sum_{e \in M^*} w(inc(S_e))$. Applying these two facts we get

$$\begin{aligned}
w(M') &\geq w(M) + \varepsilon \cdot \sum_{e \in M^*_{\geq\beta}} \left(\frac{1}{\beta} \cdot w(S_e) - w(inc(S_e))\right) \\
&= (1-\varepsilon) \cdot w(M) + \varepsilon \cdot \sum_{e \in M^*_{<\beta}} w(inc(S_e)) + \varepsilon \cdot \sum_{e \in M^*_{\geq\beta}} \frac{1}{\beta} \cdot w(S_e) \\
&> (1-\varepsilon) \cdot w(M) + \varepsilon \cdot \sum_{e \in M^*_{<\beta}} \frac{1}{\beta} \cdot w(S_e) + \varepsilon \cdot \sum_{e \in M^*_{\geq\beta}} \frac{1}{\beta} \cdot w(S_e) \\
&= (1-\varepsilon) \cdot w(M) + \frac{\varepsilon}{\beta} \cdot \sum_{e \in M^*} w(S_e) \\
&\geq (1-\varepsilon) \cdot \alpha \cdot w(M_{opt}) + \frac{\varepsilon}{\beta} \cdot \frac{2}{3} \cdot w(M_{opt}) \\
&= \left(\alpha + \varepsilon \cdot \left(\frac{2}{3\beta} - \alpha\right)\right) \cdot w(M_{opt})
\end{aligned}$$

□

4 The Algorithm

For the algorithm we have to extend the notion of weight-augmenting short paths and cycles slightly. Let $S \subset E$ be a set of at most two non-adjacent edges such that there exists an edge e in E that is adjacent to all edges in S. Then by removing all edges from a matching M' that are adjacent with some edge in S and by adding S one obtains a new matching M''. If $w(M'') > w(M')$ we say that S is a *(short) augmenting set centered at e* with respect to M'. As S contains at most two edges there are at most four edges in M'

that will be removed. Note that the sets S_e introduced in Section 3 are augmenting sets centered at e with respect to M.

For the description of the algorithm and the proof of its performance ratio we need the following additional definitions. Let M denote the maximal matching that the algorithm begins with and which is the matching that defines the set S_e as described in Section 3. Let M' denote the matching that is continuosly updated by the algorithm by means of augmentations. Let $aug(e)$ denote the set of edges that the algorithm chooses for a β-augmentation at $e \in M$. For an arbitrary set F of edges denote by $inc'(F)$ all edges in M' that are incident to the endpoints of edges in F.

Algorithm improve_matching $(\boldsymbol{G = (V, E)}, \boldsymbol{w : E \to \mathbb{R}_+}, \mathbf{M})$

```
1  make M maximal
2  M' := M
3  for e ∈ M do begin
4    if there exists a β-augmentation with center e
5      then augment M' by a good β-augmentation with center e
6  end
7  return M'
```

Fig. 1. Algorithm improve_matching for increasing the weight of a matching.

The algorithm, which we call improve_matching, is shown in Figure 1. Starting from a maximal matching M the algorithm visits each edge $e \in M$ exactly once. For each $e \in M$ the algorithm determines if there is any β-augmenting set centered at e in M'. If there is none then the algorithm moves on to the next edge in M. Otherwise, there is a β-augmenting set centered at e. The algorithm then tries to find the best β-augmenting set centered at e. The *gain* of an augmenting set S is defined to be $w(S) - w(inc'(S))$ which is the amount by which M' increases by augmenting S. We define the *best β-augmenting set* centered at e to be the β-augmenting set centered at e with the largest gain.

However, the algorithm is not guaranteed to find the best β-augmenting set centered at e but rather it finds a good β-augmenting set at e. We define a *good β-augmenting set* centered at e to be a β-augmenting set centered at e with a gain of at least $\frac{\beta-1}{4}$ times the gain of the best β-augmenting set centered at e. For technical reasons we assume from now on that $1 < \beta \leq \frac{3}{2}$ which is no restriction as in the end β will turn out to be very close to 1.

Figure 2 shows our algorithm for finding a good β-augmentation. It takes an edge e as input and returns a good β-augmenting set centered at e if any such set exists. We need a few more definitions to describe the algorithm. For an arbitrary edge x let $w_M(x)$ be 0, if $x \notin M$ and define $w_M(x) = w(x)$ otherwise. Arbitrarily label the endnodes of e as

left and *right*. Then any edge $a \notin M'$ that is incident to left together with $inc'(a) \setminus \{e\}$ form a *left arm* of e. The definition of a *right arm* of e is symmetrical to this. The *gain* of an arm of e that consists of $a \notin M'$ together with $inc'(a) \setminus \{e\}$ is defined as $gain_a := w_M(a) - w_M(inc'(a) \setminus \{e\})$. The gain of an arm of e that consists of just $a \notin M'$ is defined in the obvious way as just $gain_a := w_M(a)$.

Algorithm good-β-augmentation ($G = (V, E), w : E \to \mathbb{R}_+, e \in E$)

```
1  find the right and left arms of e
2  determine the gains and surpluses of the left and right arms
3  left := largest left allowable arm and its best extension
4  right := largest right allowable arm and its best extension
5  if left= ∅ and right= ∅
6      then return ∅
7      else return max(left, right)
```

Fig. 2. Algorithm for finding a good β-augmentation.

We define a left arm $a \cup (inc'(a) \setminus \{e\})$ to be *allowable* if there exists a right arm $b \cup (inc'(b) \setminus \{e\})$ such that $a \cup b$ or a alone forms a β-augmenting set at e. We calculate the left allowable arms as follows: First, we calculate the greatest surplus from among the right arms, where we define the *surplus* of the right arm $b \cup (inc'(b) \setminus \{e\})$ as $w_M(b) - \beta \cdot (w_M(inc'(b) \setminus \{e\}) + w_M(e))$. One can think of the largest surplus from among all the right arms, denoted $surp_r$, as the maximum value that a right arm can loan to a left arm in order to make it part of a β-augmenting set at e. A left arm is allowable if and only if $w_M(a) - \beta \cdot w_M(inc'(a) \setminus \{e\}) + surp_r \geq 0$. The definition of a right allowable arm, the maximum left surplus $surp_l$, as well as the process for calculating these is symmetrical.

Once the algorithm has calculated the left and right allowable arms of e it chooses from among these the one with the largest gain. Without loss of generality let it be a left arm. Let $a \notin M'$ be the uncovered edge in this arm. Then the algorithm returns the best β-augmenting set centered at e that contains a.

Lemma 4. *If there exists a β-augmenting set centered at e then the algorithm good-β-augmentation (Figure 2) returns a good β-augmenting set centered at e. The running time is proportional to the sum of the degrees of the end-vertices of the edge e.*

Proof. Sketch of proof: If the largest possible gain of an arm is larger than twice the weight of e then one easily gets that the algorithm finds a β-augmentation that achieves at least $\frac{1}{3}$ of the largest possible gain. This is because then the best β-augmentation gains at most 3 times the weight of e and the algorithm finds a β-augmentation that gains at least the weight of e.

In the other case, using the fact that the algorithm finds a β-augmentation that does not share an arm with the best possible β-augmentation, one can show that $\beta - 1$ must

be sufficiently small such that $\frac{4}{\beta-1}$ is big enough to scale the gain found by the algorithm. For $\beta < \frac{3}{2}$ the latter is larger. □

Lemma 5. *The algorithm improve_matching improves the matching M by at least*

$$\sum_{e \in M^*_{\geq\beta}} \frac{(\beta-1)^2}{8\beta} \cdot \left(\frac{1}{\beta} \cdot w(S_e) - w(inc(S_e)) \right).$$

Proof. Define $\delta := \frac{\beta-1}{4}$. The algorithm visits every $e \in M$ and hence every $e \in M^*_{\geq\beta}$. If it finds any β-augmenting set for e it also finds and augments a β-augmenting set $aug(e)$ which yields at least δ of the gain of the currently best β-augmenting set at e. Even though the algorithm cannot distinguish between the edges of $M_{\geq\beta}$ and $M_{<\beta}$ or know the previously defined β-augmenting sets S_e we show by means of amortization that for each of these sets S_e it finds a constant proportion of $\frac{1}{\beta} \cdot w(S_e) - w(inc(S_e))$. The idea is that for each $e \in M^*_{\geq\beta}$ the algorithm can either find an augmenting set as good as S_e in M or the matching M' has increased by enough weight to already assign a constant proportion of this gained weight to e.

The idea of the amortized analysis is that when the algorithm augments at $e \in M^*_{\geq\beta}$ then M' either gains a constant proportion of $\frac{1}{\beta} \cdot w(S_e) - w(inc(S_e))$ right away or M' can additionally make a withdrawal of weight that has been added to M''s savings in the past in a way that brings a total win of some constant proportion of $\frac{1}{\beta} \cdot w(S_e) - w(inc(S_e))$ to M'. One builds up M''s savings as follows. For each $e \in M^*_{<\beta}$ the matching M' gets charged all of the augmentation that the algorithm finds at e and this amount gets put in savings. This is not a problem because there are no sets S_e associated with these edges anyway. If $e \in M^*_{\geq\beta}$ then M' keeps $\frac{1}{2}$ of the augmentation that the algorithm finds at e and M' gets charged the other half to savings. If this is done for all $e \in M$ then M' can later make withdrawals from savings when necessary. This is necessary when for instance one needs to augment at $e \in M^*_{\geq\beta}$ but the edges incident to S_e all have greater weight in M' than they had in M. Let $E \subset (M' \setminus M)$ denote the set of new edges incident with the nodes of S_e for some such $e \in M^*_{\geq\beta}$. Then a withdrawal from M''s savings of $\frac{1}{2} \cdot \frac{\beta-1}{\beta} \cdot w(E)$ can be made for the augmentation at e. The factor $\frac{\beta-1}{\beta}$ comes from the fact that the edges in the set E were added to M' during β-augmentations that occured in the past and so there must have been at least this much put in savings in the past. The factor $\frac{1}{2}$ comes from the fact that each edge in E has two endnodes and therefore each e in E can be involved in at most two withdrawals since each of the S_e are node disjoint.

More concretely, when the algorithm visits $e \in M^*_{\geq\beta}$ there are three possibilities for the set S_e: The first is that S_e is still β-augmenting in M' with $w(inc'(S_e)) \leq w(inc(S_e))$, the second is that S_e is still β-augmenting in M' with $w(inc'(S_e)) > w(inc(S_e))$, and the

third is that S_e is no longer β-augmenting in M'.

For the first possibility for $e \in M^*_{\geq\beta}$ we have S_e is still β-augmenting with $w(inc'(S_e)) \leq w(inc(S_e))$ when the algorithm visits it. Since the algorithm always finds a β-augmentation $aug(e)$ of at least δ the gain of the largest β-augmentation at e in M' it follows that after M' has been charged $\frac{1}{2}$ of the augmentation found at e the amount of weight that M' increases by is

$$\begin{aligned}\frac{1}{2}(w(aug(e)) - w(inc'(aug(e)))) &\geq \frac{\delta}{2}(w(S_e) - w(inc'(S_e))) \\ &\geq \frac{\delta}{2}(w(S_e) - w(inc(S_e))) \\ &\geq \frac{\delta}{2}(\frac{1}{\beta} \cdot w(S_e) - w(inc(S_e))).\end{aligned}$$

For the second possibility for $e \in M^*_{\geq\beta}$ we have that S_e is still β-augmenting with $w(inc'(S_e)) > w(inc(S_e))$ when the algorithm visits it. Let A denote the set $inc'(S_e) \setminus inc(S_e)$. The set A contains only *new* edges, i.e., edges that were in augmentations, therefore M''s has increased in the past by at least $\frac{\beta-1}{\beta} \cdot w(A)$ of which at least half can be withdrawn by M'. This together with the augmentation that the algorithm will find at e, one half of which M' gets to keep, means that M''s total win at e is at least

$$\begin{aligned}&\frac{1}{2}(w(aug(e)) - w(inc'(aug(e)))) + \frac{1}{2}\frac{\beta-1}{\beta} \cdot w(A) \\ \geq\ &\frac{\delta}{2}(w(S_e) - w(inc'(S_e))) + \frac{1}{2}\frac{\beta-1}{\beta}(w(inc'(S_e)) - w(inc(S_e))) \\ \geq\ &\frac{\delta}{2}\frac{\beta-1}{\beta}(w(S_e) - w(inc'(S_e)) + w(inc'(S_e)) - w(inc(S_e))) \\ \geq\ &\frac{\delta}{2}\frac{\beta-1}{\beta}(\frac{1}{\beta}w(S_e) - w(inc(S_e))).\end{aligned}$$

For the third and final possibility for $e \in M^*_{\geq\beta}$ we have that S_e is no longer β-augmenting when the algorithm visits it, i.e., $w(inc'(S_e)) > \frac{1}{\beta} \cdot w(S_e)$. Therefore, the set of edges $A = inc'(S_e) \setminus inc(S_e)$ has weight $w(A) > \frac{1}{\beta} \cdot w(S_e) - w(inc(S_e))$. Then M''s savings must have increased by at least $\frac{\beta-1}{\beta} \cdot w(A)$ of which at least $\frac{1}{2}$ can be withdrawn by M'. So independently of wether the algorithm finds a β-augmenting set at e in M', M' gets a total win in this step of at least

$$\frac{1}{2}\frac{\beta-1}{\beta}w(A) \geq \frac{1}{2}\frac{\beta-1}{\beta}(\frac{1}{\beta}w(S_e) - w(inc(S_e))).$$

The minimum weight that M' increases by at each $e \in M^*_{\geq\beta}$ over all three cases is $\frac{\delta}{2}\frac{\beta-1}{\beta}(\frac{1}{\beta}w(S_e) - w(inc(S_e)))$ which proves the lemma since we defined δ as $\frac{\beta-1}{4}$.

□

Theorem 1. *If M is any matching with $w(M) \geq \alpha \cdot w(M_{opt})$ then after applying the algorithm improve_matching one obtains a matching M' with weight at least*

$$w(M') \geq \left(\alpha + \frac{(\beta-1)^2}{8\beta}\left(\frac{2}{3\beta} - \alpha\right)\right) \cdot w(M_{opt})$$

Proof. This is an immediate consequence of Lemma 3 and Lemma 5. □

We are now able to prove the main theorem.

Main Theorem. *For each $\varepsilon > 0$ there exists a linear time approximation algorithm for the weighted matching problem with a performance ratio of $\frac{2}{3} - \varepsilon$.*

Proof. Theorem 1 shows that by repeating algorithm improve_matching one gets a matching with weight arbitrarily close to $\frac{2}{3\beta} \cdot w(M_{opt})$. Now by choosing $\beta > 1$ small enough one gets a matching with weight arbitrarily close to $\frac{2}{3} \cdot w(M_{opt})$. Note that β and the number of repeats of algorithm improve_matching are constants depending on ε. As the algorithm improve_matching has linear running time the total running time stays linear. □

References

1. D. Avis, A Survey of Heuristics for the Weighted Matching Problem, Networks, Vol. 13 (1983), 475–493
2. R. Beier, J.F. Sibeyn, A Powerful Heuristic for Telephone Gossiping, Proc. 7th Colloquium on Structural Information and Communication Complexity, Carleton Scientific (2000), 17–35
3. D.E. Drake, S. Hougardy, A Simple Approximation Algorithm for the Weighted Matching Problem, Information Processing Letters 85 (2003), 211–213
4. J. Edmonds, Maximum matching and a polyhedron with 0,1-vertices, J. Res. Nat. Bur. Standards 69B (1965), 125–130
5. C. Frömmel, C. Gille, A. Goede, C. Gröpl, S. Hougardy, T. Nierhoff, R. Preißner, M. Thimm, Accelerating screening of 3D protein data with a graph theoretical approach, to appear in Bioinformatics
6. H.N. Gabow, An efficient implementation of Edmond's algorithm for maximum matching on graphs, Journal of the ACM 23 (1976), 221–234
7. H.N. Gabow, Data Structures for Weighted Matching and Nearest Common Ancestors with Linking, SODA 1990, 434–443
8. E.L. Lawler, Combinatorial Optimization: Networks and Matroids, Holt, Rinehart and Winston, New York, 1976
9. R.H. Möhring, M. Müller-Hannemann, Complexity and Modeling Aspects of Mesh Refinement into Quadrilaterals, Algorithmica 26 (2000), 148–171
10. B. Monien, R. Preis, R. Diekmann, Quality Matching and Local Improvement for Multilevel Graph-Partitioning, Parallel Computing, 26(12), 2000, 1609–1634
11. R. Preis, Linear Time 1/2-Approximation Algorithm for Maximum Weighted Matching in General Graphs, Symposium on Theoretical Aspects of Computer Science, STACS 99, C. Meinel, S. Tison (eds.), Springer, LNCS 1563, 1999, 259–269