Parallel Approximation Algorithms for Maximum Weighted Matching in General Graphs

Ryuhei Uehara[1] and Zhi-Zhong Chen[2]

[1] Natural Science Faculty, Komazawa University
[2] Department of Mathematical Sciences, Tokyo Denki University.

Abstract. The problem of computing a matching of maximum weight in a given edge-weighted graph is not known to be P-hard or in RNC. This paper presents four parallel approximation algorithms for this problem. The first is an RNC-approximation scheme, i.e., an RNC algorithm that computes a matching of weight at least $1-\epsilon$ times the maximum for any given constant $\epsilon > 0$. The second one is an NC approximation algorithm achieving an approximation ratio of $\frac{1}{2+\epsilon}$ for any fixed $\epsilon > 0$. The third and fourth algorithms only need to know the total order of weights, so they are useful when the edge weights require a large amount of memories to represent. The third one is an NC approximation algorithm that finds a matching of weight at least $\frac{2}{3\Delta+2}$ times the maximum, where Δ is the maximum degree of the graph. The fourth one is an RNC algorithm that finds a matching of weight at least $\frac{1}{2\Delta+4}$ times the maximum on average, and runs in $O(\log \Delta)$ time, not depending on the size of the graph.

Key words: Graph algorithm, maximum weighted matching, approximation algorithm, parallel algorithm.

1 Introduction

Throughout this paper, a graph means an edge-weighted graph, unless explicitly specified otherwise. A *matching* in a graph G is a set M of edges in G such that no two edges in M share an endpoint. The *weight* of a matching M is the total weight of the edges in M. The *maximum weighted matching* (MWM) problem is to find a matching of maximum weight of a given graph. The *maximum cardinality matching* (MCM) problem is the special case of the MWM problem where all edges of the input graph have the same weight.

For the MWM problem, Edmonds' algorithm [Edm65] has stood as one of the paradigms in the search of polynomial-time algorithms for integer programming problems (see also [Gal86]). Some sophisticated implementations of his algorithm have improved its time complexity: for example, Gabow [Gab90] has given an algorithm using $O(n(m + n \log n))$ time and $O(m)$ space.

The parallel complexity of the MWM problem is still open; Galil asks if the MWM problem is P-complete [Gal86]. The best known lower bound is the CC-hardness result pointed out in [GHR95] (see [MS92] for the definition of the class CC). On the other hand, the best known upper bound is the RNC algorithm of

Karp, Upfal, and Wigderson [KUW86] for the special case where the weights on the edges of the input graph G are bounded from above by a fixed polynomial in the number of vertices in G. We call this case the *polynomial-weight case* of the MWM problem.

From the practical point of view, Edmonds' algorithm is too slow. Faster algorithms are required, and heuristic algorithms and approximation algorithms have been widely investigated. A survey of heuristic algorithms can be found in [Avi83], and some approximation algorithms can be found in [Ven87]. In particular, a $\frac{1}{2}$-approximation algorithm can be obtained by a greedy strategy that picks up the heaviest edge e and deletes e and its incident edges, and repeats this until the given graph becomes empty. Recently, Preis [Pre99] gave a linear-time implementation of this greedy algorithm.

For the MCM problem, several NC approximation algorithms are known (see [KR98] for comprehensive reference). In particular, Fischer *et al.* [FGHP93] showed an NC algorithm that computes a matching with cardinality at least $1-\epsilon$ times the maximum for any fixed constant $\epsilon > 0$. This NC approximation algorithm is based on essential properties of the MCM problem that do not belong to the MWM problem.

Our first result is an RNC approximation scheme for the MWM problem, i.e., an RNC algorithm which computes a matching of weight at least $1-\epsilon$ times the maximum for any fixed constant $\epsilon > 0$. This scheme uses the RNC algorithm for the polynomial-weight case of the MWM problem due to Karp *et al.* as a subroutine.

Our second result is an NC approximation algorithm for the MWM problem. This algorithm can be viewed as a parallelized version of the greedy strategy. For any fixed constant $\epsilon > 0$, this algorithm computes a matching of weight at least $\frac{1}{2+\epsilon}$ times the maximum. The work done by the algorithm is optimal up to a polylogarithmic factor.

In the results above, the size of the problem depends not only on the number of edges and vertices, but also on space to represent the edge weights. Thus the algorithms are not useful when the edge weights require a large amount of memories to represent. We next consider the algorithms that only use the total order of the weights, and they do not need to store each weight itself.

Our third result is an NC approximation algorithm for the MWM problem. This algorithm computes a matching of weight at least $\frac{2}{3\Delta+2}$ times the maximum weight, where Δ is the maximum degree of given graph. It runs in $O(\log n)$ time using n processors.

Our fourth result is an RNC approximation algorithm for the MWM problem. This algorithm computes a matching whose expected weight is at least $\frac{1}{2\Delta+4}$ times the maximum weight. It runs in $O(\log \Delta)$ time using n processors. Remark that the time complexity does not depend on the size of the graph; it can be performed efficiently in parallel even on a large scale distributed system.

The rest of the paper is organized as follows. In section 2, we give basic definitions of the problem. The first and second algorithms are discussed in section 3. The third and fourth algorithms for graphs with heavy weights are

stated in section 4.

2 Preliminaries

We will deal only with graphs without loops or multiple edges. Let $G = (V, E)$ be a graph. For each edge e of G, $w_G(e)$ denotes the weight of e in G. We assume that each weight is positive, and it is different from other weights. For a subset F of edges of G, $w_G(F)$ denotes the total weight of the edges in F, i.e., $w_G(F) = \sum_{e \in F} w_G(e)$. A *maximal matching* of G is a matching of G that is not properly contained in another matching. A *maximum weighted matching* of G is a matching whose weight is maximized over all matchings of G. Note that a maximum weighted matching must be maximal. Let M be a matching and $\alpha = (e_1, e_2, \cdots, e_l)$ be a path of length l in a graph. We call α an *alternating path admitted by* M if edges on α are alternately in M, that is, either $\{e_1, e_3, \cdots\} \subseteq M$ or $\{e_2, e_4, \cdots\} \subseteq M$. We sometimes unify the alternating path and the set of edges in the matching.

The *neighborhood* of a vertex v in G, denoted by $N_G(v)$, is the set of vertices in G adjacent to v. The *degree* of a vertex v in G is $|N_G(v)|$, and denoted by $\deg_G(v)$. The *maximum degree* of a graph G is $\max_{v \in V} \deg_G(v)$, and denoted by Δ_G. Without loss of generality, we assume that $\Delta_G > 2$. The notations $w_G(e)$, $\deg_G(v)$, $N_G(v)$, and Δ_G are sometimes denoted by just $w(e)$, $\deg(v)$, $N(v)$, and Δ if G is understood. For $F \subseteq E$, $G[F]$ denotes the graph (V, F).

The model of parallel computation used here is the PRIORITY PRAM where the processors operate synchronously and share a common memory, both simultaneous reading and simultaneous writing to the same cell are allowed; in case of simultaneous writing, the processor with lowest index succeeds. An NC *algorithm* (respectively, RNC *algorithm*) is one that can be implemented to run in polylogarithmic time (respectively, expected polylogarithmic time) using a polynomial number of processors on a PRIORITY PRAM. More details about the model, NC algorithms and RNC algorithms can be found in [Joh90, KR90].

An algorithm $\mathcal{A}$ for the MWM problem *achieves a ratio* of ρ if for every graph G, the matching M found by $\mathcal{A}$ on input G has weight at least $\rho \cdot w_G(M^*)$, where M^* is a maximum weighted matching of G. An RNC *approximation scheme* for the MWM problem is a family $\{\mathcal{A}_i : i \geq 2\}$ of algorithms such that for any fixed integer $i \geq 2$, $\mathcal{A}_i$ is an RNC algorithm and achieves a ratio of $1 - \frac{1}{i}$.

3 Approximation Algorithms

3.1 Common Preprocess

Throughout the rest of this paper, let G be the input graph, M^* be a maximum weighted matching of G, and n and m be the number of vertices and edges in G, respectively. Let $w_{\max}$ be the maximum weight of an edge in G. Let k be a fixed positive integer and σ be a fixed positive real with $1 > \sigma > 0$; the actual values of k and σ will become clear later.

For each edge e of G, we define an integer $r(e)$ as follows:

– If $\frac{kn}{w_{\max}} \cdot w_G(e) \leq 1$, then let $r(e) = 0$.
– Otherwise, let $r(e)$ be the smallest integer i such that $\frac{kn}{w_{\max}} \cdot w_G(e) \leq (1+\sigma)^i$.

We call $r(e)$ the *rank* of e. Note that $r(e) \in \{0, 1, \cdots, \lceil \log_{1+\sigma} kn \rceil\}$.

Let E_i be the set of edges of G with rank i. Let $M_i^* = M^* \cap E_i$. The following lemma shows that the total weight of edges in M_0^* is significantly small.

Lemma 1. $\sum_{e \in M_0^*} w_G(e) \leq \frac{1}{2k} w_G(M^*)$.

Proof. For each edge $e \in E_0$, $w_G(e) \leq \frac{w_{\max}}{kn}$. Thus, $\sum_{e \in M_0^*} w(e) \leq \frac{n}{2} \frac{w_{\max}}{kn} \leq \frac{w_{\max}}{2k} \leq \frac{1}{2k} w(M^*)$. □

Let G' be the graph obtained from G by deleting all edges of rank 0 and modifying the weight of each rest edge e to be $(1+\sigma)^{r(e)}$.

3.2 RNC approximation scheme

In this section, we present the RNC approximation scheme for the MWM problem. It is based on the following result due to Karp *et al.*

Lemma 2. [KUW86] *There is an* RNC *algorithm for the polynomial-weight case of the MWM problem.*

Theorem 3. *There exists an* RNC *approximation scheme for the MWM problem.*

Proof. Let ℓ be an integer larger than 1. Suppose we want to compute a matching M of G with $w_G(M) \geq (1 - \frac{1}{\ell}) \cdot w_G(M^*)$. Then, we fix $\sigma = \frac{1}{\ell}$ and $k = \lceil \frac{\ell^2}{2} \rceil$, and construct G' as in Section 3.1. Note that the weight of each edge of G' is polynomial in n. So, we use the RNC algorithm in Lemma 2 to compute a maximum weighted matching M' of G'. Note that M' is also a matching of G. It remains to show that $w_G(M') \geq (1 - \frac{1}{\ell}) w_G(M^*)$.

For each $i \geq 1$, let $M_i' = M' \cap E_i$. Since M' is a maximum weighted matching of G' and $\cup_{i \geq 1} M_i^*$ is a matching of G', we have $\sum_{i \geq 1} \sum_{e \in M_i^*} w_{G'}(e) \leq w_{G'}(M') = \sum_{i \geq 1} \sum_{e \in M_i'} w_{G'}(e) = \sum_{i \geq 1} |M_i'| (1+\sigma)^i$.

On the other hand, since M^* is a maximum weighted matching of G, we have

$$w_G(M^*) \geq \sum_{i \geq 1} \sum_{e \in M_i'} w_G(e) > \sum_{i \geq 1} |M_i'| \frac{w_{\max}}{kn} (1+\sigma)^{i-1}$$

since $\frac{w_{\max}}{kn}(1+\sigma)^{i-1} < w_G(e)$ if $e \in M_i'$ with $i \geq 1$.

Thus, $w_G(M') = \sum_{i \geq 1} \sum_{e \in M_i'} w_G(e) \geq \frac{w_{\max}}{kn} \cdot \frac{\sum_{i \geq 1} |M_i'| (1+\sigma)^i}{(1+\sigma)}$
$\geq \frac{w_{\max}}{kn(1+\sigma)} \sum_{i \geq 1} \sum_{e \in M_i^*} w_{G'}(e) \geq \frac{w_{\max}}{kn(1+\sigma)} \sum_{i \geq 1} \sum_{e \in M_i^*} \frac{kn}{w_{\max}} w_G(e)$
$= \frac{w_G(M^* - M_0^*)}{1+\sigma}$.

Now, by Lemma 1, $w_G(M') \geq \frac{w_G(M^* - M_0^*)}{1+\sigma} \geq \frac{1}{1+\sigma} \left(1 - \frac{1}{2k}\right) w_G(M^*)$. Clearly, $\frac{1}{1+\sigma} \left(1 - \frac{1}{2k}\right) \geq 1 - \frac{1}{\ell}$. This completes the proof. □

3.3 NC algorithm

In this section, we parallelize the greedy algorithm to obtain an NC approximation algorithm for the MWM problem that achieves a ratio of $\frac{1}{2+\epsilon}$ for any fixed $\epsilon > 0$.

The NC approximation algorithm will work on graph G' defined in Section 3.1. Recall that each edge e of G' inherits a rank $r(e)$ from G. Let $\ell = \lceil \log_{1+\sigma} kn \rceil$. The highest rank is ℓ. The algorithm works as follows:

NC Algorithm

1. For each $i = \ell, \ell-1, \cdots, 1$, perform the following steps:
   1.1. Find a maximal matching M_i in $G'_i = (V, F_i)$, where F_i is the set of edges of G' with rank i.
   1.2. Remove all edges e from G' such that $e \in M_i$ or e is incident to an endpoint of an edge in M_i.
2. Output $\cup_{1 \le i \le \ell} M_i$.

Let $M = \cup_{1 \le i \le \ell} M_i$. By Steps 1.1 and 1.2, M is clearly a maximal matching of G'.

Lemma 4. $w_G(M) \ge \frac{1}{2(1+\sigma)} w_G(M^* - M_0^*)$.

Proof. The idea is to distribute the weights of all edges of $M^* - M_0^*$ to the edges of M. Let e be an edge in $M^* - M_0^*$. Let $i = r(e)$. We distribute the weight $w_G(e)$ of e as follows:

- Case (1): $e \in M_i$. Then, we distribute $w_G(e)$ to e itself.
- Case (2): $e \notin M_i$ but one or both endpoints of e are incident to an edge of M_i. Then, we distribute $w_G(e)$ to an arbitrary edge $e' \in M_i$ such that e and e' share an endpoint. Note that $\frac{1}{1+\sigma} \le \frac{w_G(e)}{w_G(e')} \le 1 + \sigma$.
- Case (3): None of cases (1) and (2) occurs. Then, by the algorithm, e must share an endpoint with at least one edge $e' \in M_j$ such that $j > i$. We distribute $w_G(e)$ to an arbitrary such edge e'. Note that $w_G(e) \le (1+\sigma)^i \le (1+\sigma)^{j-1} < w_G(e')$.

Consider an edge $e' \in M$. Since $M^* - M_0^*$ is a matching, at most two edges $e \in M^* - M_0^*$ can distribute their weights to e'. Moreover, by Cases (1) through (3), the total weight newly distributed to e' is at most $2(1+\sigma)w_G(e')$. Thus, $\sum_{e' \in M} 2(1+\sigma) w_G(e') \ge w_G(M^* - M_0^*)$. Consequently, $w_G(M) \ge \frac{1}{2(1+\sigma)} w_G(M^* - M_0^*)$. □

Thus we have the following theorem.

Theorem 5. *There is an* NC *approximation algorithm for the MWM problem that achieves a ratio of* $\frac{1}{2+\epsilon}$ *for any fixed* $\epsilon > 0$. *It runs in* $O(\log^4 n)$ *time using* $n + m$ *processors on the PRIORITY PRAM.*

Proof. Fix a positive real number ϵ. Suppose we want to compute a matching M of G with $w_G(M) \geq \frac{1}{2+\epsilon} \cdot w_G(M^*)$. Then, we fix $\sigma = \frac{\epsilon}{3}$ and $k = \lceil \frac{3}{\epsilon} + 1.5 \rceil$, construct G' as in Section 3.1, and run the above NC algorithm on input G' to obtain a matching M. By Lemmas 1 and 4, $w_G(M) \geq \frac{1}{2(1+\sigma)} \left(1 - \frac{1}{2k}\right) w_G(M^*) \geq \frac{1}{2+\epsilon} w_G(M^*)$.

We next analyze the complexity needed to compute M. G' can be constructed from G in $O(1)$ time using $n + m$ processors. M can be computed from G' in $O(\log_{1+\sigma}(kn) \cdot (\log n + T(n, m))$ time using $\max\{(n + m), P(n, m)\}$ processors on the PRIORITY PRAM, provided that a maximal matching of a given n-vertex m-edge graph can be computed in $T(n, m)$ using $P(n, m)$ processors on the PRIORITY PRAM. According to [IS86], $T(n, m) = \log^3 n$ and $P(n, m) = n + m$. So, M can be computed in $O(\log^4 n)$ time using $n + m$ processors on the PRIORITY PRAM. □

4 Approximation Algorithms for Graphs with Heavy Weights

We next show two algorithms that only use the total order of the weights. The first one is an NC algorithm, and the second one is an RNC algorithm. Both algorithms contain three phases:

Outline of Algorithms for heavy weights

1. For given G, construct a heavy spanning forest F (defined later) of G.
2. Construct a set of paths P in $G[F]$.
3. Produce a matching in $G[P]$.

The algorithms are the same except the phase 3. We now describe each phase, and analyze their complexity and approximation ratio.

4.1 The First Phase

The first phase contains two steps:

1.1. In parallel, each vertex marks the heaviest edge incident to the vertex;
1.2. F is the set of marked edges.

We first show that $G[F]$ is acyclic.

Proposition 6. *$G[F]$ is acyclic.*

Proof. Assume $G[F]$ is not acyclic and $e_1, e_2, \cdots, e_l$ are edges producing a cycle in $G[F]$. We let $v_1, v_2, \cdots, v_l$ be vertices on the cycle. If two consecutive vertices mark the same edge, there should be an edge on the cycle not marked by any vertices. Hence each vertex marks different edge. Thus we can assume that v_i marks e_i, and $w(e_1) < w(e_2)$. However, this implies that $w(e_1) < w(e_2) < \cdots < w(e_l) < w(e_1)$, that is a contradiction. □

Thus, $G[F]$ is a set of trees. Moreover, it is trivial that $\deg_{G[F]}(v) > 0$ for all v in V. Hence we call F *heavy spanning forest* of G. We now introduce some notions for the heavy spanning forest F. Let T be a tree in $G[F]$, and n_T be the number of vertices in T. Then, in the first step, each of n_T vertices in T marks one edge, and T has $n_T - 1$ edges. This implies that T has exactly one edge marked by its both endpoints. We call the edge and two vertices *a root edge* and *two roots* of T, respectively. That is, each tree has one root edge and two roots. We can show the following lemma by a simple induction.

Lemma 7. *Let T be a tree in F, and e_r be the root edge of T. Then for any leaf-root path $(e_l, e_1, e_2, \cdots, e_r)$ in T, $w(e_l) < w(e_1) < w(e_2) < \cdots < w(e_r)$.*

That is, the root edge is the heaviest edge in the tree. We now show the theorem for the relation between $w(M^*)$ and $w(F)$.

Theorem 8. $w(F) \geq w(M^*)$.

Proof. Let $e = \{u, v\}$ be an edge in M^*, but not in F. Then, since $e \notin F$, e is not marked by both u and v. Let e_u and e_v be edges marked by u and v, respectively. Since M^* is a matching, neither e_u nor e_v is not in M^*. That is, $\{e_u, e_v\} \subseteq F - M^*$. Now we divide the weight $w(e)$ in two weights $\frac{1}{2}w(e)$, and distribute them to e_u and e_v, respectively. Since e is not marked, $w(e) < w(e_u), w(e_v)$. Moreover, e_u and e_v are not distributed by the other edges in M^* at the points u and v since e is an element in the matching M^*. That is, no edge e' in $F - M^*$ will be distributed more than $w(e')$ by the edges in $M^* - F$. Thus each edge in M^* is either in F or it can be divided and distributed to two edges in $F - M^*$. This implies that $w(F) \geq w(M^*)$. □

We moreover analyze the proof of Theorem 8 in detail. Let $C = F \cap M^*$, $\hat{F} = F - C$, and $\hat{M} = M^* - C$. We let R be the set of the root edges of F. Then we have the following corollary.

Corollary 9. $w(F) \geq 2w(M^*) - w(C) - w(R)$.

Proof. In the proof of Theorem 8, each weight of an edge in $\hat{M}$ is *divided* and distributed to two edges in $\hat{F}$, because corresponding edges in $\hat{F}$ can be distributed at both endpoints. However, only root edges can be distributed at both endpoints. Now we distribute each weight of an edge in $\hat{M}$ onto two edges in $\hat{F}$ without division. In the case, root edges may be distributed twice, hence we have $w(\hat{F}) \geq 2w(\hat{M}) - w(R)$. Thus $w(F) = w(\hat{F}) + w(C) \geq 2w(\hat{M}) - w(R) + w(C) = 2w(M^* - C) - w(R) + w(C) = 2w(M^*) - w(C) - w(R)$. □

4.2 The Second Phase

2.1. In parallel, each vertex v with $\deg_{G[F]}(v) > 2$ deletes all edges incident to v except two heaviest edges. Let P be the set of remaining edges. (Comment: It is easy to see that $G[P]$ is a set of paths, and $\deg_{G[P]}(v) > 0$ still holds if v was not a leaf in F.)

We show a proposition and a lemma for trees, and main theorem in this subsection.

Proposition 10. *Let T be a tree with n vertices. We let n_i be the number of vertices of degree i with $1 \leq i \leq \Delta_T$. Then (a) $\sum_{i=1}^{\Delta_T} n_i = n$; and (b) $\sum_{i=1}^{\Delta_T} i n_i = 2(n-1)$.*

Proof. (a) is trivial. When each vertex counts up its degree, each edge is counted exactly twice. Since any tree with n vertices has $n-1$ edges [Har72, Chapter 4], (b) follows. □

Lemma 11. *Let $L(n, \Delta)$ be the maximum number of leaves in a tree of maximum degree Δ with n vertices. Then $L(n, \Delta) \leq \frac{(\Delta-2)n-2}{\Delta-1}$.*

Proof. Let T be an n vertex tree with $L(n, \Delta)$ leaves. Let V_I be the set of internal vertex of T. Then, the graph induced by V_I is also a tree. The induced tree contains $|V_I|$ vertices and $|V_I| - 1$ edges. Each leaf of T is incident to one internal vertex. Moreover, each vertex in V_I can be incident to at most Δ vertices. Thus we have $L(n, \Delta) \leq \Delta |V_I| - 2(|V_I| - 1)$. We also have $L(n, \Delta) + |V_I| = n$. Hence $L(n, \Delta) \leq \frac{(\Delta-2)n-2}{\Delta-1}$. □

Theorem 12. $w(P) > \frac{1}{\Delta_{G[F]}-1} w(F)$.

Proof. We first assume that $G[F]$ contains only one tree. We let n_i be the number of vertices of degree i in $G[F]$ with $1 \leq i \leq \Delta$. Each vertex does not delete the nearest edge to the root edge. Thus no edge will be deleted by two different vertices. Hence, by Proposition 10, the number of edges deleted in step 2.1 is equal to $\sum_{i=3}^{\Delta}(i-2)n_i = \sum_{i=3}^{\Delta} i n_i - 2\sum_{i=3}^{\Delta} n_i = 2(n-1) - n_1 - 2n_2 - 2(n - n_1 - n_2) = n_1 - 2$. Using Lemma 11, we have $\frac{|P|}{|F|} = \frac{n-1-n_1+2}{n-1} \geq \frac{n+1-L(n,\Delta)}{n-1} \geq \frac{(\Delta-1)(n+1)-(\Delta-2)n+2}{(n-1)(\Delta-1)} = \frac{n+\Delta+1}{(n-1)(\Delta-1)} = \frac{1}{\Delta-1} + \frac{\Delta+2}{(n-1)(\Delta-1)} > \frac{1}{\Delta-1}$.

We then consider the weights of deleted edges. For each deleted edge e, there exists at least one edge e' in P with $w(e') > w(e)$. On the other hand, for each remaining edge e' in P, it is corresponded by such deleted edges e at most $\Delta - 1$ times. This together with $\frac{|P|}{|F|} > \frac{1}{\Delta-1}$ implies that $\frac{w(P)}{w(F)} > \frac{1}{\Delta-1}$.

When $G[F]$ contains two or more trees, the discussion above can be applied on each tree. More precisely, let F contain k trees $T_1, T_2, \cdots, T_k$, and P_i be the path set obtained from T_i with $1 \leq i \leq k$. Using the discussion above, we have $\frac{w(P_i)}{w(T_i)} > \frac{1}{\Delta-1}$, consequently, $w(P_i) > \frac{w(T_i)}{\Delta-1}$ with $1 \leq i \leq k$. Thus $w(P) = \sum_{i=1}^{k} w(P_i) > \frac{1}{\Delta-1}\sum_{i=1}^{k} w(T_i) = \frac{1}{\Delta-1} w(F)$, consequently, $\frac{w(P)}{w(F)} > \frac{1}{\Delta-1}$. □

4.3 The Third Phase

NC Algorithm

We define the distance of edges to describe the third phase of NC algorithm. Let $\alpha = (e_1, e_2, \cdots, e_l)$ be a path of length l. Then the *distance* of e_i from e_1 on α, denoted by $D(e_i, e_1)$, is defined by $D(e_1, e_1) = 0$, and $D(e_i, e_1) = D(e_{i-1}, e_1) + 1$ for $1 < i \leq l$. The third phase of NC algorithm contains the following steps:

3.1. In parallel, find the heaviest edge in each path in P.
3.2. M_N is the set of edges having even distance from the heaviest edge on the same path.

As a result, each path in P becomes alternating path containing the heaviest edge on the path admitted by M_N. We here show a proposition and a useful lemma for paths with special properties.

Proposition 13. *Let $\alpha = (e_1, e_2, \cdots, e_l)$ be a path with $w(e_1) > w(e_2) > \cdots > w(e_l)$. Then the maximum weighted matching of α, say M_α, is either $\{e_1, e_3, \cdots, e_l\}$ for odd l, or $\{e_1, e_3, \cdots, e_{l-1}\}$ for even l. Moreover, $w(M_\alpha) \geq \frac{1}{2} w(\alpha)$.*

Proof. When l is even, considering $w(e_1) > w(e_2)$, $w(e_3) > w(e_4)$, $\cdots$, $w(e_{l-1}) > w(e_l)$, we immediately have the proposition. When l is odd, the last edge e_l just increases the weight of M_α. □

Lemma 14. *Let $\alpha = (e_1, e_2, \cdots, e_l)$ be a path such that $w(e_1) < w(e_2) < \cdots < w(e_{i-1}) < w(e_i) > w(e_{i+1}) > \cdots > w(e_l)$ for some i with $1 < i < l$. Let A_1 be the alternating path containing e_i, A_2 be the other alternating path, and M_α be the maximum weighted matching of α.*

(1) Either $A_1 = M_\alpha$ or $A_2 = M_\alpha$. (Hence $w(M_\alpha) \geq \frac{1}{2} w(\alpha)$.)
(2) When $A_2 = M_\alpha$, $w(e_{i-1}) + w(e_{i+1}) \geq w(e_i)$.
(3) $w(A_1) \geq \frac{1}{3} w(\alpha)$.

Proof. (1) We first show that M_α satisfies either (a) $e_i \in M_\alpha$ or (b) $\{e_{i-1}, e_{i+1}\} \subseteq M_\alpha$. To derive a contradiction, assume that $e_i \notin M_\alpha$, $e_{i-1} \notin M_\alpha$, and $e_{i+1} \in M_\alpha$. Then $(M_\alpha - \{e_{i+1}\}) \cup \{e_i\}$ is a matching heavier than M_α since $w(e_i) > w(e_{i+1})$, that is a contradiction. The other symmetric case ($e_i \notin M_\alpha$, $e_{i-1} \in M_\alpha$, and $e_{i+1} \notin M_\alpha$) can be shown by the same argument. In the case (a), we have $e_i \in M_\alpha$, $e_{i-1} \notin M_\alpha$, and $e_{i+1} \notin M_\alpha$. Then we can consider α as two paths $(e_1, e_2, \cdots, e_{i-2})$ and $(e_{i+2}, \cdots, e_l)$ separately. Using Proposition 13, we have $M_\alpha = \{e_i\} \cup \{e_{i-2}, e_{i-4}, \cdots\} \cup \{e_{i+2}, e_{i+4}, \cdots\}$, consequently, $M_\alpha = A_1$. In the case (b), we have $e_i \notin M_\alpha$, $e_{i-1} \in M_\alpha$, and $e_{i+1} \in M_\alpha$. Thus we have $M_\alpha = \{e_{i-1}, e_{i+1}\} \cup \{e_{i-3}, e_{i-5}, \cdots\} \cup \{e_{i+3}, e_{i+5}, \cdots\}$, consequently, $M_\alpha = A_2$.
(2) Let A_2' be $(A_2 - \{e_{i-1}, e_{i+1}\}) \cup \{e_i\}$. Then A_2' is a matching. Since A_2 is the maximum weighted matching, $w(A_2') \leq w(A_2)$. This implies that $w(e_{i-1}) + w(e_{i+1}) \geq w(e_i)$.
(3) By (1), either $A_1 = M_\alpha$ or $A_2 = M_\alpha$. When $A_1 = M_\alpha$, the claim follows from Proposition 13. Thus we assume that $A_2 = M_\alpha$.

We here consider two paths $\alpha_1 = (e_1, e_2, \cdots, e_{i-1}, e_i)$ and $\alpha_2 = (e_i, e_{i+1}, \cdots, e_l)$. Let A_1^1 (and A_1^2) be the alternating path of α_1 (and α_2, resp.) containing e_i. That

is, A_1^1 is the former half of A_1, A_1^2 is the latter half of A_1, and $A_1^1 \cap A_1^2 = \{e_i\}$. Then, by Proposition 13, $w(A_1^1) \geq \frac{1}{2}w(\alpha_1)$ and $w(A_1^2) \geq \frac{1}{2}w(\alpha_2)$.

Thus, $w(A_1) = w(A_1^1 \cup A_1^2) = w(A_1^1) + w(A_1^2) - w(A_1^1 \cap A_1^2) \geq \frac{1}{2}(w(\alpha_1) + w(\alpha_2)) - w(e_i) = \frac{1}{2}(w(\alpha) + w(e_i)) - w(e_i) = \frac{1}{2}(w(\alpha) - w(e_i)) \geq \frac{1}{2}(w(\alpha) - w(A_1))$. This implies that $w(A_1) \geq \frac{1}{3}w(\alpha)$. □

We here remark that, in Lemma 14(1), we cannot determine which alternating path is heavier in general. (For example, a path (e_1, e_2, e_3) has different answers when $w(e_1) = 1, w(e_2) = 3, w(e_3) = 1$ and $w(e_1) = 2, w(e_2) = 3, w(e_3) = 2$). We also remark that Lemma 14(1) does not hold for general weighted path (for example, each alternating path of (e_1, e_2, e_3, e_4) is not the maximum weighted matching for $w(e_1) = 5, w(e_2) = 1, w(e_3) = 1, w(e_4) = 5$).

We now show the relation between $w(M_N)$ and $w(P)$.

Lemma 15. $w(M_N) \geq \frac{1}{3}w(P)$.

Proof. We first observe the following claim: in step 2.1, if vertex v delete an edge $\{u, v\}$, the edge was marked by u in step 1.1. This is easy because each vertex marked the heaviest edge in step 1.1, and remains the heaviest edge(s) in step 2.1. Using the claim and simple induction, we can show that each path in P is either

(1) a part of some leaf-root path in some tree in F; or
(2) two leaf-root paths connected by the root edge in a tree in F.

In the case (1), combining Lemma 7 and Proposition 13, M_N contains the maximum weighted matching of the path, and that has at least half weight of the path. Thus it is sufficient to show for the case (2). By Lemma 7, the path $\alpha = (e_1, e_2, \cdots, e_l)$ satisfies that $w(e_1) < w(e_2) < \cdots < w(e_{r-1}) < w(e_r) > w(e_{r+1}) > \cdots > w(e_l)$, where e_r is the root edge. Thus, according to Lemma 14(3), for the alternating path A_r containing e_r, $w(A_r) \geq \frac{1}{3}w(\alpha)$. Thus $w(M_N) \geq \frac{1}{3}w(P)$. □

Combining Theorem 8, Theorem 12, and Lemma 15, we can show that the NC algorithm is a $\frac{1}{3(\Delta-1)}$-approximation algorithm. But the better approximation ratio $\frac{2}{3\Delta+2}$ will be stated in Section 4.5.

RNC Algorithm

Phases 1 and 2 are performed "locally". That is, all computations can be performed within the neighbors. Using randomization, RNC algorithm finds a matching in $G[P]$ locally. The third phase of the RNC algorithm contains the following steps:

3.1'. In parallel, each vertex randomly choose one of two edges incident to the vertex in $G[P]$. (The vertices of degree one choose the unique edge incident to the vertex.)

3.2'. M_R is the set of edges chosen by both endpoints.

Since each vertex choose one edge, the resulting M_R is a matching. Moreover, since each edge in P is chosen with probability at least $\frac{1}{4}$, we immediately have the following lemma.

Lemma 16. *The expected value of $w(M_R)$ is at least $\frac{1}{4}w(P)$.*

4.4 Complexity of Algorithms

Each algorithm uses n processors; every vertex in G has a processor associated with it. As the input representation of G, we assume that each vertex has a list of the edges incident to it. Thus, each edge $\{i, j\}$ has two copies - one in the edge list for vertex i and the other in the edge list for vertex j.

Theorem 17. *The* NC *algorithm runs in $O(\log n)$ time using n processors on the PRIORITY PRAM. The algorithm only requires the total order of the weights.*

Proof. Each processor uses two memory cells to store the edges in P. The first and second phases can be efficiently implemented modifying as follows:

1.1'. In parallel, each vertex v finds the heaviest edge $e = \{v, u\}$ incident to v;
1.2'. In parallel, v stores the first cell of v with e.
2.1'. In parallel, each vertex v checks the contents of the first cell of u. If it is e, then the process is end. If it is not e, v tries to store the second cell of u with e. This trial will succeed if $w(e)$ is the heaviest among the other edges that are tried to store the same cell.

The step 1.2' can be done in a unit time. Moreover, it is not difficult to see that the steps 1.1' and 2.1' can be done in $O(\log \Delta)$ time using standard technique with comparison operation.

In the third phase, we can easy to see the following:
(1) if $e = \{u, v\}$ is a root edge, e is stored in the first cells of both u and v; and (2) otherwise, e is stored in the first cell of one endpoint, and in the second cell of the other.

Moreover, each non-root edge knows which endpoint is close to the root edge; the endpoint storing the second cell with the edge. Thus step 3.1 can be done in $O(1)$ time, and step 3.2 can be done in $O(\log n)$ time using standard list ranking technique (see e.g., [KR90]).

Throughout the computation, the algorithm only compares two weights of edges. Thus the algorithm only requires to know the total order of the weights. □

The third phase of the RNC algorithm can be performed in $O(1)$ time. This immediately implies the following theorem.

Theorem 18. *The* RNC *approximation algorithm runs in $O(\log \Delta)$ time using n processors on the PRIORITY PRAM. The algorithm only requires the total order of the weights.*

4.5 Approximation Ratios

We remind that M^* is a maximum weighted matching, F is the heavy spanning forest, R is the set of the root edges of F, and P is a set of paths obtained in step 2.1. Moreover we let $C = F \cap M^*$, $\hat{F} = F - C$, and $\hat{M} = M^* - C$.

To derive good approximation ratios, we define two maximum matchings: M_P denotes a maximum weighted matching of $G[P]$, and M_F denotes a maximum weighted matching of $G[F]$.

Lemma 19. $w(M_P) \geq \frac{1}{2(\Delta-1)} w(F)$.

Proof. As seen in the proof of Lemma 15, each path in P is either

(1) a part of some leaf-root path in some tree in F; or
(2) two leaf-root paths connected by a root edge in a tree in F.

For each path, by Lemma 14(1), M_P contains heavier alternating path that has at least half weight of the path. This together with Theorem 12 implies that $w(M_P) \geq \frac{1}{2} w(P) \geq \frac{1}{2(\Delta-1)} w(F)$. □

Lemma 20. $w(M_F) \geq \frac{1}{\Delta} w(M^*)$.

Proof. We first remind that M_F is the maximum weighted matching in F. Thus, since C is a matching in F, $w(M_F) \geq w(C)$. It is easy to see that R is a matching in F. This implies that $w(M_F) \geq w(R)$. It is also easy to see that M_P is a matching in F, and thus $w(M_F) \geq w(M_P)$. Hence, combining Corollary 9, we have $w(F) \geq 2w(M^*) - w(C) - w(R) \geq 2w(M^*) - 2w(M_F)$. On the other hand, by Lemma 19, $w(M_F) \geq w(M_P) \geq \frac{1}{2(\Delta-1)} w(F)$. Combining the equations, we have $(2(\Delta-1)+2)w(M_F) \geq 2w(M^*)$, consequently, $w(M_F) \geq \frac{1}{\Delta} w(M^*)$. □

Lemma 21. $w(C) \leq w(M_F) \leq 2w(M_P)$.

Proof. It is clear that $w(C) \leq w(M_F)$. Thus we show $w(M_F) \leq 2w(M_P)$. We are going to show that the weight of each edge in M_F can be distributed to an edge in M_P, and the weight of each edge in M_P is distributed by such edges at most twice. Let $e = \{u, v\}$ be any edge in M_F. Then three cases occur according to e.
(1) $e \in M_P$. We distribute $w(e)$ to itself.
(2) $e \in P - M_P$. We first assume that e is not a root edge. We assume that u is closer to the root edge than v on $G[F]$. In the case, e is incident to e' in P at the vertex u with $w(e') > w(e)$. Thus we distribute $w(e)$ to e'. We next assume that e is a root edge. That is, e is a root edge not in the maximum weighted matching of $G[P]$. Then, by Lemma 14, M_P contains two edges e' and e'' such that e' and e'' are the edges incident to e at vertex u and v, respectively, and $w(e') + w(e'') \geq w(e)$. Thus we divide $w(e)$ into $w(e')$ and $w(e) - w(e') (\leq w(e''))$, and distribute them to e' and e'', respectively.
(3) $e \notin M_P$. We assume that u is closer to the root edge than v on $G[F]$. In the case, e was deleted by u in step 2.1. The vertex u remains two edges e' and

e'' in P with $w(e'), w(e'') > w(e)$. Moreover, either e' or e'' is in M_P. Thus we distribute $w(e)$ to the edge in M_P.

Since M_F is a matching, no two edges are distributed at the same endpoint. Thus each edge in M_P is distributed at most twice at both endpoints. This implies that $w(M_F) \leq 2w(M_P)$. □

Theorem 22. $w(M_P) \geq \frac{2}{2\Delta+1} w(M^*)$.

Proof. Combining Corollary 9 and Lemma 19, we get $w(M_P) \geq \frac{1}{2(\Delta-1)} w(F) \geq \frac{1}{2(\Delta-1)}(2w(M^*) - w(R) - w(C))$. Using Lemma 21, we have $2w(M_P) \geq w(C)$. On the other hand, since $R \subseteq M_N$, $w(M_P) \geq w(M_N) \geq w(R)$. Thus, $w(M_P) \geq \frac{1}{2(\Delta-1)}(2w(M^*) - w(R) - w(C)) \geq \frac{1}{2(\Delta-1)}(2w(M^*) - w(M_P) - 2w(M_P))$. Thus $w(M_P) \geq \frac{2}{2\Delta+1} w(M^*)$. □

Theorem 23. *The approximation ratio of the* NC *algorithm is* $\frac{2}{3\Delta+2}$.

Proof. We first show that $w(M_N) \geq \frac{1}{2} w(M_P)$. As seen in the proof of Lemma 15, each path in P is either

(1) a part of some leaf-root path in some tree in F; or
(2) two leaf-root paths connected by a root edge in a tree in F.

In the case (1), both M_N and M_P contain the same alternating path that contains the heaviest edge. We consider the paths in the case (2). Let α be the path in P, and A_1 be the alternating path of α containing the root edge, and A_2 be the other alternating path. According to Lemma 14, A_1 or A_2 is the maximum weighted matching of α. When A_1 is the maximum weighted matching, both M_N and M_P contain it. Now we assume that A_2 is the maximum weighted matching of α. Then, by Lemma 14(3), $w(A_1) \geq \frac{1}{3} w(\alpha)$, consequently, $w(A_2) \leq \frac{2}{3} w(\alpha)$. Thus $w(A_1) \geq \frac{1}{2} w(A_2)$. Therefore, in any cases, $w(M_N) \geq \frac{1}{2} w(M_P)$.

Combining Corollary 9, Theorem 12, and Lemma 15, we have $w(M_N) \geq \frac{1}{3} w(P) \geq \frac{1}{3(\Delta-1)} w(F) \geq \frac{1}{3(\Delta-1)}(2w(M^*) - w(C) - w(R))$. It is clear that $w(M_N) \geq w(R)$ since $R \subseteq M_N$. Thus, using Lemma 21, we have $w(M_N) \geq \frac{1}{3(\Delta-1)}(2w(M^*) - w(C) - w(R)) \geq \frac{1}{3(\Delta-1)}(2w(M^*) - 2w(M_P) - w(M_N)) \geq \frac{1}{3(\Delta-1)}(2w(M^*) - 5w(M_N))$, consequently, $w(M_N) \geq \frac{2}{3\Delta+2} w(M^*)$. □

Theorem 24. *The approximation ratio of the* RNC *algorithm is* $\frac{1}{2\Delta+4}$.

Proof. Using Corollary 9, Theorem 12, and Lemma 16, we have $E(w(M_R)) \geq \frac{1}{4(\Delta-1)} w(F) \geq \frac{1}{4(\Delta-1)}(2w(M^*) - w(C) - w(R))$.

We now compare $w(M_R)$ with $w(M_P)$. Each edge in M_P appears in M_R with probability at least $\frac{1}{4}$. This implies that the expected value of $w(M_R)$ is at least $\frac{1}{4} w(M_P)$. Thus, using Lemma 21, we have $E(w(M_R)) \geq \frac{1}{4(\Delta-1)}(2w(M^*) - w(C) - w(R)) \geq \frac{1}{4(\Delta-1)}(2w(M^*) - 3w(M_P)) \geq \frac{1}{4(\Delta-1)}(2w(M^*) - 12E(w(M_R)))$, consequently, $E(w(M_R)) \geq \frac{1}{2\Delta+4} w(M^*)$. □

References

[Avi83] D. Avis. A Survey of Heuristics for the Weighted Matching Problem. *Networks*, 13:475–493, 1983.

[Edm65] J. Edmonds. Paths, Trees and Flowers. *Canad. J. Math.*, 17:449–467, 1965.

[FGHP93] T. Fischer, A.V. Goldberg, D.J. Haglin, and S. Plotkin. Approximating matchings in parallel. *Information Processing Letters*, 46:115–118, 1993.

[Gab90] H.N. Gabow. Data Structures for Weighted Matching and Nearest Common Ancestors with Linking. In *Proc. 1st Ann. ACM-SIAM Symp. on Discrete Algorithms*, pages 434–443. ACM, 1990.

[Gal86] Z. Galil. Efficient Algorithms for Finding Maximum Matching in Graphs. *Computing Surveys*, 18(1):23–38, 1986.

[GHR95] R. Greenlaw, H.J. Hoover, and W.L. Ruzzo. *Limits to Parallel Computation.* Oxford University Press, 1995.

[Har72] F. Harary. *Graph Theory.* Addison-Wesley, 1972.

[IS86] A. Israeli and Y. Shiloach. An Improved Parallel Algorithm for Maximal Matching. *Information Processing Letters*, 22:57–60, 1986.

[Joh90] D.S. Johnson. A Catalog of Complexity Classes. In J. van Leeuwen, editor, *The Handbook of Theoretical Computer Science, Vol. I: Algorithms and Complexity*, pages 69–161. Elsevier, 1990.

[KR90] R.M. Karp and V. Ramachandran. Parallel Algorithms for Shared-Memory Machines. In J. van Leeuwen, editor, *The Handbook of Theoretical Computer Science, Vol. I: Algorithms and Complexity*, pages 870–941. Elsevier, 1990.

[KR98] M. Karpinski and W. Rytter. *Fast Parallel Algorithms for Graph Matching Problems.* Clarendon Press, 1998.

[KUW86] R.M. Karp, E. Upfal, and A. Wigderson. Constructing a Perfect Matching is in Random NC. *Combinatorica*, 6(1):35–48, 1986.

[MS92] E.W. Mayr and A. Subramanian. The Complexity of Circuit Value and Network Stability. *Journal of Computer and System Science*, 44:302–323, 1992.

[Pre99] R. Preis. Linear Time $\frac{1}{2}$-Approximation Algorithm for Maximum Weighted Matching in General Graphs. In *STACS '99*, pages 259–269. Lecture Notes in Computer Science Vol. 1563, Springer-Verlag, 1999.

[Ven87] S.M. Venkatesan. Approximation Algorithms for Weighted Matching. *Theoretical Computer Science*, 54:129–137, 1987.

This article was processed using the LaTeX macro package with LLNCS style