Approximating Maximum Weight Matching in Near-linear Time

Ran Duan
EECS Department
University of Michigan
Ann Arbor, MI
duanran@umich.edu

Seth Pettie
EECS Department
University of Michigan
Ann Arbor, MI
pettie@umich.edu

***Abstract*—Given a weighted graph, the *maximum weight matching* problem (MWM) is to find a set of vertex-disjoint edges with maximum weight. In the 1960s Edmonds showed that MWMs can be found in polynomial time. At present the fastest MWM algorithm, due to Gabow and Tarjan, runs in $\tilde{O}(m\sqrt{n})$ time, where m and n are the number of edges and vertices in the graph. Surprisingly, restricted versions of the problem, such as computing $(1-\epsilon)$-approximate MWMs or finding maximum cardinality matchings, are not known to be much easier (on sparse graphs). The best algorithms for these problems also run in $\tilde{O}(m\sqrt{n})$ time.**

In this paper we present the first near-linear time algorithm for computing $(1-\epsilon)$-approximate MWMs. Specifically, given an arbitrary real-weighted graph and $\epsilon > 0$, our algorithm computes such a matching in $O(m\epsilon^{-2}\log^3 n)$ time. The previous best approximate MWM algorithm with comparable running time could only guarantee a $(2/3-\epsilon)$-approximate solution. In addition, we present a faster algorithm, running in $O(m \log n \log \epsilon^{-1})$ time, that computes a $(3/4-\epsilon)$-approximate MWM.

***Keywords*-graph, matching, approximation**

I. Introduction

The history of the maximum matching problem is intertwined with the development of modern graph theory, combinatorial optimization, matroid theory, and the conflation of polynomial time computation with feasibility [8]. After decades of research on the problem, the computational complexity of finding an optimal matching remains quite open. (We recommend [33], [48], [1] for a review of definitions and algorithms.) In bipartite graphs, finding a maximum cardinality matching in polynomial time is trivial, but the same is not true in general graphs. In 1965 Edmonds presented elegant polynomial time algorithms for finding matchings in general graphs with maximum cardinality (MCM) [8] and maximum weight (MWM) [7]. In the intervening years we have seen a succession of faster and more complex algorithms for solving these problems, usually based on new structural characterizations and more sophisticated data structures.

Algorithms: Early implementations of Edmonds's algorithm required $O(n^3)$ time [26], [12], [31], [3] using elementary data structures. Following the approach of Hopcroft and Karp's MCM algorithm for bipartite graphs [25], Micali and Vazirani [38] presented an MCM algorithm for general graphs running in $O(m\sqrt{n})$ time, where m and n are the number of edges and vertices.[1] Over the years others have proposed alternative $O(m\sqrt{n})$-time MCM algorithms that generalize Micali and Vazirani's [16], [18]. All of the algorithms cited above are based on incrementally improving a matching via *augmenting paths*. Using an algebraic characterization of the problem [45], Mucha and Sankowski [39] and Harvey [22] presented MCM algorithms whose running time is roughly $O(n^\omega)$, the complexity of square matrix multiplication.

If the input graph is weighted one may naturally ask for a matching whose weight is minimum or maximum, possibly with the restriction that it simultaneously have maximum cardinality. All these variants are equivalent; see [16]. For bipartite graphs, the implementation of the Hungarian algorithm [30] using Fibonacci heaps [11] runs in $O(mn + n^2 \log n)$ time, a bound that is matched in general graphs by Gabow [13] using more complex data structures. Faster algorithms are known when the edge weights are bounded integers in $[-N, \ldots, N]$, where a word RAM model is assumed, with $\log(\max\{N, n\})$-bit words. Gabow and Tarjan [15], [16] gave bit-scaling algorithms for MWM running in $O(m\sqrt{n}\log(nN))$ time in bipartite graphs and $O(m\sqrt{n\log n}\log(nN))$ time in general graphs. Extending [39], Sankowski [46] gave an $O(Nn^\omega)$-time MWM algorithm for bipartite graphs.

Approximation Algorithms: Let a δ-MWM be a matching whose weight is at least a δ fraction of the maximum weight matching, where $0 < \delta \leq 1$, and let δ-MCM be defined analogously. The $O(m\sqrt{n})$-time MCM algorithms [25], [38] are all actually $(1-1/k)$-MCM algorithms running in $O(km)$ time. In each phase they augment the current matching using a maximal set of augmenting paths with minimum length. It is easy to show [25] that (i) the length of the augmenting paths increases in each phase, (ii) any matching without length $2k-3$ augmenting paths is a $(1-1/k)$-MCM, and that (i,ii) imply that $O(\sqrt{n})$ phases

[1] A complete description and proof of correctness of the Micali-Vazirani algorithm appears in [49]. The Micali-Vazirani algorithm was preceded by one of Even and Kariv [10] who claimed a running time of $O(\min\{n^{5/2}, m\sqrt{n}\log n\})$ in an extended abstract. However, a complete description of the algorithm was never published.

suffice to compute an MCM.

Surprisingly, the best approximate MWM algorithms do not achieve anything close to the accuracy and efficiency of the decades old approximate MCM algorithms [25], [38]. On real weighted graphs the Gabow-Tarjan algorithm [16] gives a $(1 - n^{-\Theta(1)})$-MWM in $O(m\sqrt{n}\log^{3/2} n)$ time, simply by retaining the $O(\log n)$ high order bits in each edge weight, treating them as polynomial size integers. It is well known that the *greedy* algorithm—iteratively choose the maximum weight edge not incident to previously chosen edges—produces a $\frac{1}{2}$-MWM. A straightforward implementation of this algorithm takes $O(m \log n)$ time. Preis [43], [5], reviving the δ-MWM problem from its long slumber, gave a $\frac{1}{2}$-MWM algorithm running in linear time. Vinkemeier and Hougardy [50] and Pettie and Sanders [42] proposed several $(\frac{2}{3} - \epsilon)$-MWM algorithms (see also [37]) running in $O(m \log \epsilon^{-1})$ time; each is based on iteratively improving a matching by identifying sets of short weight-augmenting paths and cycles. No linear time algorithms with approximation ratio better than $\frac{2}{3}$ are known.

New Results: Our main result is the first $(1-\epsilon)$-MWM algorithm for arbitrary weighted graphs whose running time is nearly linear. In particular, we show that such a matching can be found in $O(m\epsilon^{-2}\log^3 n)$ time. We also present a faster algorithm that finds $(\frac{3}{4} - \epsilon)$-MWMs in time $O(m \log n \log \epsilon^{-1})$, which improves the accuracy of $(\frac{2}{3} - \epsilon)$-MWM algorithms [50], [42] though is a logarithmic factor slower. (In independent and unpublished work, Hanke and Hougardy [21], [20] obtained a similar $(\frac{3}{4} - \epsilon)$-MWM algorithm, as well as a $(\frac{4}{5} - \epsilon)$-MWM algorithm running in $O(m \log^2 n \log \epsilon^{-1})$ time.)

Technical Challenges: After one surveys the state-of-the-art in exact and approximate MWM algorithms, one finds two natural avenues to a $(1 - \epsilon)$-MWM algorithm. The first is to extend the $\frac{2}{3}$-MWM algorithms [42], [50] in a straightforward way, by looking for weight-augmenting paths and cycles with length on the order of $1/\epsilon$. Due to the haphazard way edges are placed in the current matching it is very difficult to find long augmentations, augmenting cycles in particular. We are able to improve the approximation ratio of [42], [50] to $\frac{3}{4} - \epsilon$ by better handling augmenting 6-cycles. However, new obstacles arise that prevent us from going further. A more principled approach to the problem is to start with Gabow and Tarjan's [16] scaling algorithm, which solves a version of the problem at each of $\log N$ scales, each of which consists of $O(\sqrt{n})$ iterations. The goal of each scale is not to compute a matching per se, but to compute a hierarchy of nested blossoms and a set of nearly tight dual variables on the vertices and blossoms. Is it necessary to perform all $O(\sqrt{n})$ iterations at one scale? The answer, unfortunately, seems to be yes. Performing $\tilde{O}(\epsilon^{-1})$ iterations before prematurely jumping to the next scale leaves us with essentially useless dual variables. This difficulty arises in the absence of blossoms and is exacerbated by their presence. We develop a new scaling technique that does not require that dual variables be carried from one scale to the next. This technique is fit for finding approximate-MWMs but not exact ones. At each scale we apply a new and simple $(1-\epsilon)$-MWM algorithm for small integer weights, which follows the standard primal-dual approach to the problem.

Some Applications of Approximate Matching: In input-queued switches packets are routed across a "switch fabric" from input to output ports. In each cycle one partial permutation can be realized. Existing algorithms for choosing these matchings, such as iSLIP [35] and PIM [2], guarantee $\frac{1}{2}$-MCMs and it has been shown [36], [17] that (approximate) maximum weight matchings, where edge weights are based on queue-length, have good throughput guarantees. (Of course, computing *exact* MWMs is unrealistic in this application.)

Approximate MWM algorithms are a key component in several clustering libraries.[2] These clustering algorithms are used, for example, to partition a graph across many parallel processors so as to minimize the communication cost in certain algorithms [28], [29], [24]. Maximum weight matching algorithms are used as a heuristic preprocessing step in several sparse linear system solvers [40], [6], [47], [19]. The goal is to permute the rows/columns to maximize the weight on or near the main diagonal.

Definitions and Conventions: The input is a graph $G = (V, E, w)$ where $|V| = n, |E| = m$, and $w : E \to \mathbb{R}$. A *matching* M is a set of vertex-disjoint edges. Vertices not incident to an M edge are *free*. We use v' to refer to the *mate* of a matched vertex v, that is, if $(v, v') \in M$. An alternating path (or cycle) is one whose edges alternate between M and $E\backslash M$. An alternating path/cycle P is *augmenting* if $M \oplus P$ is a matching, where $\oplus$ is symmetric difference, and $w(M \oplus P) > w(M)$, where $w(M) = \sum_{e\in M} w(e)$. If weights are not discussed, an *augmenting path* is one whose ends are free vertices. Since we only seek $(1 - \epsilon)$ approximate solutions, we can afford to scale and round edge weights to small integers. Henceforth, $w : E \to \{1, \ldots, N\}$, where $N \leq n^2$. In Section II N is regarded as being a much smaller number.

Overview: In Section II we give a $(1-\epsilon)$-MWM algorithm whose running time depends linearly on the maximum edge weight. In Section III we present a scaling algorithm for $(1-\epsilon)$-MWM running in $O(m \log^3 n)$ time, for fixed ϵ. Section IV gives a $(3/4 - \epsilon)$-MWM algorithm running in $O(m \log n)$ time.

II. Approximate MWMs for Small Weights

In this section we describe a simple algorithm for finding $(1-\epsilon)$-MWMs in graphs with integer weights in $\{1, \ldots, N\}$

[2]The clustering libraries METIS [27], PARTY [44], PT-SCOTCH [41] CHACO [23], and JOSTLE [51] all use some approximate matching routine. PARTY, for example, builds a hierarchical clustering by iteratively finding and contracting approximate MWMs. Preis's algorithm [43] was specifically motivated by this application.

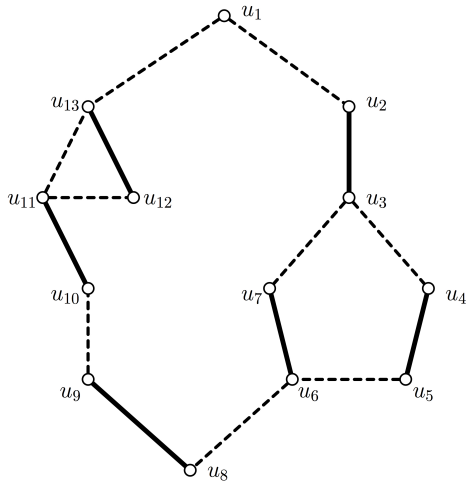

Figure 1. A blossom $(u_1, u_2, S_1, u_8, u_9, u_{10}, S_2)$ containing non-trivial sub-blossoms $S_1 = (u_3, u_4, u_5, u_6, u_7)$ and $S_2 = (u_{11}, u_{12}, u_{13})$. Solid edges are in the matching.

that runs in $O(Nm/\epsilon)$ time. This algorithm serves as a component in our $(1-\epsilon)$-MWM algorithm for arbitrarily large weights, presented in Section III. We use the standard LP formulation of Edmonds [7], but maintain only "ϵ-optimal" dual variables (see Bertsekas [4] and Gabow and Tarjan [15], [16]) with respect to the current matching. Roughly speaking, they satisfy the complementary slackness conditions up to an additive ϵ.

Preliminaries: The algorithm maintains a dynamic set Ω of nested *blossoms*. Blossoms are formed inductively as follows. If $v \in V$ then the set $\{v\}$ is a trivial blossom, and not kept in Ω. An odd length sequence $(A_0, A_1, \ldots, A_\ell)$ forms a blossom $B = \bigcup_i A_i$ (with respect to a matching M) if the $\{A_i\}_i$ are blossoms and there is a sequence of edges $e_0, \ldots, e_\ell$ where $e_i \in A_i \times A_{i+1}$ (modulo $\ell+1$) and $e_i \in M$ iff i is odd, that is, A_0 is incident to unmatched edges e_0, e_ℓ. See Figure 1. The set of *blossom edges* E_B are $\{e_0, \ldots, e_\ell\}$ and those used in the formation of $A_0, \ldots, A_\ell$. The set $E(B) = E \cap (B \times B)$ may include many non-blossom edges. At any time the nested structure of blossoms in Ω is represented as a forest of rooted trees, where the children of a node correspond to its constituent blossoms. The roots in the blossom forest are *root blossoms*. The *contracted graph* is formed by contracting all root blossoms to single vertices. To *dissolve* a root blossom B means to delete its node in the blossom forest and, in the contracted graph, to replace B with individual vertices $A_0, \ldots, A_\ell$. Observe that if M is a matching in G then what remains of M in the contracted graph is also a matching. Furthermore, an alternating path in the contracted graph extends to an alternating path in G.

We maintain two potential functions $y : V \to \mathbb{R}$ and $z : \Omega \to \mathbb{R}$ that satisfy Property 1 with respect to the current matching M. For $(u, v) \in E$ let $yz(u, v) = y(u) + y(v) + \sum_{B \in \Omega, (u,v) \in E(B)} z(B)$.

Property 1. (Relaxed Complementary Slackness) Let k be a fixed positive integer.

1) $z(B) \geq 0$ for all $B \in \Omega$ and $z(B) > 0$ if B is a root blossom.
2) $yz(e) \geq w(e) - 1/k$ for all $e \in E$.
3) $yz(e) \leq w(e)$ when $e \in M$ or $e \in E_B$ for some $B \in \Omega$.
4) The y-values of free vertices are equal; the y-value of a matched vertex is at least that of a free vertex.

Lemma 1 shows that when y-values of free vertices are reduced to zero, Property 1 guarantees that we have found a sufficiently good matching.

Lemma 1. *Suppose Property 1 holds for a matching M where y-values of free vertices are zero, and let M' be any other matching, possibly the MWM.*

1) $w(M) \geq w(M') - |M'|/k$.
2) *M is a $(1 - 1/k)$-MWM.*
3) *Let $P \subseteq M \oplus M'$ be an alternating cycle or an alternating path whose ends are free vertices or edges in M. Then $w(M \cap P) \geq w(M' \cap P) - |M' \cap P|/k$.*

Proof: By the integrality of edge weights, Part 1 implies Part 2. We prove Part 1 first. By Property 1 we have:

$$w(M') \leq \sum_{u \in V(M')} y(u) + \sum_{\substack{e \subseteq M' \cap E(B), \\ B \in \Omega}} z(B) + |M'|/k \quad (1)$$

$$\leq \sum_{u \in V} y(u) + \sum_{B \in \Omega} z(B)|M' \cap E(B)| + |M'|/k \quad (2)$$

$$\leq \sum_{u \in V} y(u) + \sum_{B \in \Omega} z(B)\lfloor |B|/2 \rfloor + |M'|/k \quad (3)$$

$$\leq w(M) + |M'|/k \quad (4)$$

Inequalities (1–3) follow from Property 1 parts 2, 4, and 1, respectively. Inequality (4) follows from part 3 of Property 1 and that fact that $y(u) = 0$ for free u. Part 3 is proved similarly:

$$\begin{aligned} & w(M' \cap P) \\ & \leq \sum_{u \in V(M' \cap P)} y(u) + \sum_{B \in \Omega} z(B)|M' \cap P \cap E(B)| \\ & \qquad + |M' \cap P|/k && \text{Property 1(2)} \\ & \leq \sum_{u \in V(P)} y(u) + \sum_{B \in \Omega} z(B)|M' \cap P \cap E(B)| \\ & \qquad + |M' \cap P|/k && y\text{-values are non-neg.} \end{aligned}$$

The last line follows since y-values are non-negative and the ends of P, if P is a path, are unmatched vertices or edges in M. Furthermore, observe that $M' \cap P \cap E(B)$ consists of subpaths of P, each of which cannot have *both* end edges in M', since there is only one unmatched vertex with respect

to M' in the subgraph on B. Thus $|M \cap P \cap E(B)| \geq |M' \cap P \cap E(B)|$. Continuing:

$$\begin{aligned} &\leq \sum_{u \in V(P)} y(u) + \sum_{B \in \Omega} z(B)|M \cap P \cap E(B)| \\ &\quad + |M' \cap P|/k \qquad \text{From the obs. above} \\ &\leq w(M \cap P) + |M' \cap P|/k \qquad \text{Property 1(3)} \end{aligned}$$

■

A. The High-Level Algorithm

Initially $M = \emptyset, \Omega = \emptyset$, and $y(v) = N$ for all $v \in V$, which clearly satisfies Property 1. The algorithm repeatedly finds sets of augmenting paths of *eligible edges*, creates and destroys blossoms, and performs dual adjustments on y, z in order to maintain Property 1 and increase the number of eligible edges.

Definition 1. An edge e is *eligible* if $e \in M$ and $yz(e) = w(e)$, if $e \notin M$ and $yz(e) = w(e) - 1/k$, or if $e \in E_B$ for some $B \in \Omega$. Let G' be the graph of eligible edges and H be G' after contracting root blossoms in Ω.

Note that the eligible edges in H satisfy one of the first two criteria since all blossom edges are contracted. Also note that G' and H are initially the graph $(V, \emptyset)$.

In contrast to Edmonds's [7] primal-dual weighted perfect matching algorithm, we do not halt when M is perfect, but when the y-values of free vertices reach zero. One iteration of the algorithm consists of *Augmentation*, *Blossom Shrinking*, and *Dual Adjustment* steps:

1) **Augmentation:** Find a maximal set Ψ of augmenting paths in H and set $M \leftarrow M \oplus (\bigcup_{P \in \Psi} P)$. Update G' and H.
2) **Blossom Shrinking:** Let $V_{out} \subseteq V(H)$ be the vertices (that is, root blossoms) reachable from free vertices by even-length alternating paths; let Ω' be a maximal set of (nested) blossoms on V_{out}[3]. Let $V_{in} \subseteq V(H) \backslash V_{out}$ be those vertices reachable from free vertices by odd-length alternating paths. Set $z(B) \leftarrow 0$ for $B \in \Omega'$ and set $\Omega \leftarrow \Omega \cup \Omega'$. Update G' and H.
3) **Dual Adjustment:** Let $\hat{V}_{in}, \hat{V}_{out} \subseteq V$ be original vertices represented by vertices in V_{in} and V_{out}. The y- and z-values for some vertices and root blossoms are adjusted:
   Set $y(v) \leftarrow y(v) - \frac{1}{2k}$ for all $v \in \hat{V}_{out}$.
   Set $y(v) \leftarrow y(v) + \frac{1}{2k}$, for all $v \in \hat{V}_{in}$.
   Set $z(B) \leftarrow z(B) + \frac{1}{k}$, if $B \in \Omega$ is a root blossom containing vertices in $\hat{V}_{out}$.
   Set $z(B) \leftarrow z(B) - \frac{1}{k}$, if $B \in \Omega$ is a root blossom containing vertices in $\hat{V}_{in}$.
   After dual adjustments some root blossoms may have zero z-values. Dissolve such blossoms (remove them from Ω) as long as they exist. Note that non-root blossoms are allowed to have zero z-values. Update G' and H by the new Ω.

The augmentations performed in Step 1 take time linear in their length in H, not G. We do not need to consider the parts of augmenting paths inside shrunken blossoms until such blossoms are dissolved, in Step 3. Using a slightly modified depth-first search[4] and a standard union-find data structure one can execute Step 1 in $O(m\alpha(m, n))$ time, or in $O(m)$ time using the incremental-tree set-union structure [14]; see [16, §8].[5] Steps 2 and 3 can be implemented in linear time using a modified depth first search. When the input graph is bipartite Step 1 is easy to implement in linear time (there being no need to detect or deal with blossoms), Step 2 is unnecessary, and Step 3 remains trivial.

Lemma 2. *After the Augmentation and Blossom Shrinking steps H contains no augmenting path, nor is there a path from a free vertex to a blossom.*

Proof: If there is an augmenting path P in H after augmenting along paths in Ψ, since Ψ is maximal, P must intersect some $P' \in \Psi$ at a vertex v. (View P' as a path in H. Portions inside Ω blossoms are contracted.) However, every edge in P' will become ineligible after augmenting, so the matching edge (v, v') in M will not be in H. Thus there is no augmenting path in H post-augmentation. Since Ω' is maximal there can be no blossom reachable from a free vertex in H after the blossom shrinking step. ■

Lemma 3. *After the Dual Adjustment step, all edges will not offend Property 1.*

Proof: Property 1(4) is obviously maintained. Property 1(1) is also maintained since all the new root blossoms discovered in the Blossom Shrinking step are in V_{out} and will have positive z-values after adjustment. Furthermore, each root blossom whose z-value drops to zero is removed.

The algorithm clearly ensures that y- and z-values are multiples of $\frac{1}{2k}$ and $\frac{1}{k}$, respectively, and that y-values of free vertices are equal. Thus, the y-values of all vertices in $\hat{V}_{in} \cup \hat{V}_{out}$ must have the same parity (that is, they have the same parity in multiples of $\frac{1}{2k}$), since each is connected by a path of eligible edges to a free vertex. Recall that an eligible edge e must have $yz(e) = w(e)$ or $w(e) - 1/k$. With these observations we can show that Property 1(2,3) is not offended. Let $e = (u, v)$ be an edge, and suppose first that both u, v are in $\hat{V}_{in} \cup \hat{V}_{out}$. If $u, v \in B \in \Omega$ then $yz(e)$ is

[3]That is, if $(u, v) \in E(H) \backslash M$ and $u, v \in V_{out}$, then u and v must be in a common blossom.

[4]The chief modification is to visit a vertex u after an even-length alternating path from a free vertex to u is detected. For example, in Figure 1 the vertices could be visited in the order $u_1, u_3, u_6, u_9, u_{11}, u_{13}, u_{12}, u_4, u_5, u_7, u_{10}, u_8, u_2$.

[5]We are not aware of a linear time pointer machine implementation of Step 1 using "elementary" data structures or an $O(m\sqrt{n})$-time MCM algorithm using elementary data structures. The set-union structure of [14] uses precomputed tables and world-packing techniques, which are not available on a pointer machine.

unchanged, preserving the property, so we can assume that u and v are in different root blossoms. If $e \notin M$ is ineligible then, due to parity, $yz(e) \geq w(e)$ before adjustment and $yz(e) \geq w(e)-1/k$ afterward, which preserves the property. If $e \notin M$ is eligible then at least one of u, v is in $\hat{V}_{in}$ (otherwise another blossom or augmenting path would have been formed), so $yz(e)$ cannot be reduced. If $e \in M$ then it must be eligible, so $u \in \hat{V}_{in}$ and $v \in \hat{V}_{out}$ and $yz(e)$ is unchanged. Now suppose u, but not v, is in $\hat{V}_{in} \cup \hat{V}_{out}$. If $e \notin M$ is eligible then $u \in \hat{V}_{in}$ and $yz(e)$ will increase; if it is ineligible then $yz(e) \geq w(e) - \frac{1}{2k}$ before adjustment and $yz(e) \geq w(e) - 1/k$ afterward. If $e \in M$ then it must be ineligible, so $u \in \hat{V}_{in}$, $yz(e) \leq w(e) - \frac{1}{2k}$ before adjustment and $yz(e) \leq w(e)$ afterward. ■

Thus, in $O(kNm)$ time we can obtain a matching M satisfying Property 1, whose free vertices have zero y-values. Lemma 1 implies that $w(M^*) - w(M) \leq |M^*|/k$, and, consequently, that M is a $(1 - 1/k)$-MWM.

III. A Scaling Technique for Large Weights

We introduce a scaling approach tailored to the approximate maximum weight matching problem that has a logarithmic dependence on the maximum edge weight N, rather than linear, as in Section II. The standard scaling technique [16], [15] is not easily applicable. Between scales it performs a dual variable adjustment on a near-optimal perfect matching and near-tight set of dual variables. Unless the matching satisfies these properties it is not obvious how to proceed at the next scale. It is also not clear how to transform our algorithm from Section II into a scaling algorithm. Among other obstacles, its running time depended on the *positiveness* and *uniqueness* of the dual variables of free vertices, which would be destroyed in a series of scaling steps.

A. The Algorithm

Our $(1 - \epsilon)$-MWM algorithm uses the algorithm from Section II as a black box. Recall the input graph is $G = (V, E, w)$, where weights are at most N. At the ith iteration in the algorithm we have a matching M_i and a current graph G_i on the vertex set $V \backslash V(M_i)$ whose edge weights are bounded by N_i. We find an approximate MWM $\tilde{M}$ in a reweighted version of G_i, and add all sufficiently heavy edges in $\tilde{M}$ to M_i, yielding M_{i+1}.

Initially $M_0 = \emptyset$, $G_0 = G$, $N_0 = N$. Perform iterations $i = 0, \ldots, \log N$ and return $M_{\log N+1}$.

Iteration i:

1) Let G'_i have the same structure as G_i, but with weight function $w'(e) = \lfloor x \cdot w(e)/N_i \rfloor$, where x will be determined later.
2) Let $\tilde{M}$ be the approximate MWM of G'_i returned by the algorithm in Section II, with parameter $k = 1$. The running time will be $O(xm)$. (In our analysis we only use Lemma 1(1,3), not (2). $\tilde{M}$ is trivially a $(1-1/k)$-MWM = 0-MWM.)
3) Prepare $M_{i+1}, N_{i+1}, G_{i+1}$ for the next iteration:
   Set $M_{i+1} = M_i \cup \{e \mid e \in \tilde{M} \text{ and } w(e) > N_i/2\}$.
   Set $N_{i+1} = N_i/2$.
   Set G_{i+1} to be the graph with vertex set $V \backslash V(M_{i+1})$ and edge set $\{e \in E \mid w(e) \leq N_{i+1}\}$.

Since w' assigns integer weights at most x, independent of i, the running time is dominated by the cost of computing $\tilde{M}$, which is $O(xm \log N) = O(xm \log n)$ over all iterations. To analyze the approximation ratio of the matching obtained by this algorithm, we need a modification of a lemma of Pettie and Sanders [42]. The proof of Lemma 4 appears in the appendix.

Lemma 4. *Let M be a matching, M^* the MWM, and p an integer. There exists a collection A of vertex-disjoint alternating paths/cycles w.r.t M and M^* with the following properties. Each augmentation in A has at most $2p$ edges from M^*; the ends of augmenting paths in A are either free vertices with respect to M or edges from M; $w(M \oplus A) \geq \frac{p}{p+1} w(M^*)$.*

B. The Analysis

We begin by focusing on the first iteration, where we find a matching $\tilde{M}$ of $G = G_0$ under the weight function w'. Let A be the set of augmenting paths/cycles guaranteed by Lemma 4, for some p to be determined later. Following Step 2, Lemma 1 implies that for each $P \in A$, $w'(M^* \cap P) \leq w'(\tilde{M} \cap P) + |M^* \cap P|$, so $w'(M^* \cap P) \leq w'(\tilde{M} \cap P) + 2p$. Since $w'(e) = \lfloor x \cdot w(e)/N \rfloor$, $w(e) < Nw'(e)/x + N/x$, and we can bound $w(M^* \cap P)$ as follows:

$$\begin{aligned} w(M^* \cap P) &\leq \tfrac{N}{x} w'(M^* \cap P) + 2Np/x \\ &\qquad \text{Since } |M^* \cap P| \leq 2p \text{ from Lemma 4} \\ &\leq \tfrac{N}{x} w'(\tilde{M} \cap P) + 4Np/x \qquad \text{Lemma 1} \\ &\leq w(\tilde{M} \cap P) + 4Np/x \qquad \text{Defn. of } w' \end{aligned}$$

If there is an edge $e \in P$ satisfying $w(e) > N/2$ then either $w(\tilde{M} \cap P) > N/2$ or $w(M^* \cap P) > N/2$; the bound above implies that in either case $w(\tilde{M} \cap P) > N/2 - 4Np/x$. If P does have an edge with weight greater than $N/2$ it follows that:

$$\begin{aligned} \frac{w(M^* \cap P)}{w(\tilde{M} \cap P)} &\leq \frac{w(\tilde{M} \cap P) + 4Np/x}{w(\tilde{M} \cap P)} \\ &< 1 + \frac{4Np/x}{N/2 - 4Np/x} \;=\; (1 - 8p/x)^{-1} \end{aligned}$$

To analyze the approximation ratio we need to show there is little "loss" by permanently putting the edge set $M_1 = \{e \in \tilde{M} \mid w(e) > N/2\}$ into our final matching. Let M_i^* be the maximum weight matching of G_i. We first show that $w(M_1) + w(M_1^*)$ is a good approximation to $w(M^*)$.

Imagine constructing a matching Q of G that contains edges in G_1 and $\tilde{M}$ satisfying one of three conditions:

Condition 1. If $P \in A$ contains an edge with weight larger than $N/2$, include $\tilde{M} \cap P$ in Q.

Condition 2. For other $P \in A$, include $M^* \cap P$ in Q.

Condition 3. Include $\tilde{M} \setminus A$ in Q.

It is easy to see that Q is a matching. Since M_1 is disjoint from G_1 and $M_1 \subseteq Q$ (due to Conditions 1 and 3), it follows that $w(Q) \leq w(M_1^*) + w(M_1)$. Combining the inequalities we have obtained so far, we can lower bound $w(Q)$ as:

$$\begin{aligned}
w(Q) &= w(\tilde{M} \setminus A) + \sum_{P \text{ in Condition } 1} w(\tilde{M} \cap P) \\
&\quad + \sum_{P \text{ in Condition } 2} w(M^* \cap P) \\
&\geq w(\tilde{M} \setminus A) + \sum_{P \text{ in Cond. } 1} (1 - \tfrac{8p}{x}) w(M^* \cap P) \\
&\quad + \sum_{P \text{ in Condition } 2} w(M^* \cap P) \\
&\geq w(\tilde{M} \setminus A) + (1 - \tfrac{8p}{x}) \sum_{P \in A} w(M^* \cap P) \\
&\geq (1 - \tfrac{8p}{x}) w(\tilde{M} \oplus A) \qquad (5) \\
&\geq (1 - \tfrac{8p}{x})(1 - \tfrac{1}{p+1}) w(M^*) \qquad (6)
\end{aligned}$$

Inequality (5) follows from the identity $\tilde{M} \oplus A = (\tilde{M} \setminus A) \cup (A \setminus \tilde{M}) = (\tilde{M} \backslash A) \cup (A \cap M^*)$ and Inequality (6) follows from Lemma 4. Thus, $w(M_1^*) + w(M_1) \geq w(Q) \geq (1 - \frac{8p}{x})(1 - \frac{1}{p+1}) w(M^*)$. This proof clearly applies to any level i, that is, if we just restrict our attention to G_i (rather than all of G_0, which we did above),

$$w(M_{i+1}^*) + w(M_{i+1} \backslash M_i) \geq (1 - \tfrac{8p}{x})(1 - \tfrac{1}{p+1}) w(M_i^*) \quad (7)$$

The bounds given in Lemma 5 let us select the parameters p, x in the final algorithm:

Lemma 5. $w(M_i) + w(M_i^*) \geq [(1 - \frac{8p}{x})(1 - \frac{1}{p+1})]^i w(M^*)$.

Proof: The proof for $i = 1$ was shown above. Let $\delta = (1 - \frac{8p}{x})(1 - \frac{1}{p+1})$. Assuming the lemma is true for M_i we show it holds for M_{i+1} as well.

$$\begin{aligned}
&w(M_{i+1}) + w(M_{i+1}^*) \\
&= w(M_i) + w(M_{i+1} \backslash M_i) + w(M_{i+1}^*) \\
&\geq w(M_i) + \delta \cdot w(M_i^*) && \text{Inequality (7)} \\
&\geq \delta(w(M_i) + w(M_i^*)) && \delta < 1 \\
&\geq \delta^{i+1} w(M^*) && \text{Inductive hypothesis}
\end{aligned}$$

■

Theorem 1. *A $(1 - \epsilon)$-MWM can be computed in $O(\epsilon^{-2} m \log^3 n)$ time.*

Proof: By Lemma 5 the matching $M_{\log N+1}$ returned by the algorithm satisfies $w(M_{\log N+1}) \geq [(1 - \frac{8p}{x})(1 - \frac{1}{p+1})]^{\log N+1} w(M^*)$. If we intend to find a $(1 - \epsilon)$-approximate MWM, set $p = 2\epsilon^{-1}(\log N + 1) - 1$ and $x = 8p(p+1)$, so the approximation ratio is:

$$\begin{aligned}
\frac{w(M_{\log N+1})}{w(M^*)} &\geq [(1 - \tfrac{8p}{x})(1 - \tfrac{1}{p+1})]^{\log N+1} \\
&= (1 - 1/(p+1))^{2(\log N+1)} \\
&= (1 - \tfrac{1}{2\epsilon^{-1}(\log N+1)})^{2(\log N+1)} \\
&> 1 - \epsilon
\end{aligned}$$

Thus, the total running time is $O(xm \log N) = O(\epsilon^{-2} m \log^3 n)$. ■

IV. A 3/4-Approximation to MWM

A k-augmentation a with respect to a matching M is one for which $|a \backslash M| \leq k$. Our algorithm starts with an empty matching M and, in successive iterations, finds a set A of disjoint 3-augmentations and applies them to M. Define the *gain* $g(a)$ with respect to M as $w(a \backslash M) - w(a \cap M)$; the gain of a set of augmentations is the sum of their individual gains. Say a k-augmenting path a is *centered* at $e \in a \backslash M$ if the number of non-M edges on either side of e is at most $\lfloor k/2 \rfloor$. For the time being a k-augmenting cycle is centered at any of its non-M edges.

Whereas 3-augmenting paths and 2-augmenting cycles are relatively easy to find, 3-augmenting cycles (that is, cycles with length 6) are not. Lemma 6 states that there always exists a set A_3 of disjoint 3-augmenting paths/cycles that brings us geometrically closer to a 3/4-MWM, even if the gain of an augmenting cycle is *devalued*. We can implement each iteration of our algorithm in near-linear time by not specifically looking for 3-augmenting cycles; Lemma 6 essentially lets us look for 3-augmenting paths using an edge e whose gain is at least as good as the devalued gain of a 3-augmenting cycle using e. This distinction will become more clear in the analysis.

Lemma 6. *(see Pettie and Sanders [42]) Let M be a matching in a graph and M^* be the maximum weight matching. Let $g_k(a) = \frac{k}{k+1} w(a \backslash M) - w(a \cap M)$ if a is an k-augmenting cycle and $g_k(a) = g(a)$ if a is a k-augmenting path. There is a set A_k of* vertex disjoint *k-augmentations such that $\sum_{a \in A_k} g_k(a) \geq \frac{k+1}{2k+1}(\frac{k}{k+1} w(M^*) - w(M))$.*

Our algorithm looks for a high-gain augmenting path centered at each edge $e = (u, v) \in E \backslash M$, i.e., one consisting of (u', u, v, v') and two *arms* (x', x, u') and (v', y, y') *anchored* at u' and v', respectively. See Figure 3. Let $\mathrm{arms}(u')$ be a list of arms (i.e., alternating paths of two edges) anchored at u', sorted by gain. Using the arms lists the procedure $\mathrm{aug}(u, v)$ (Figure 2) will find the best augmenting path centered at (u, v), in $O(1)$ time. (As we will see, vertices progressively become ineligible; $\mathrm{arms}(u')$ actually stores arms containing only eligible vertices V_{elig}.) If, through luck, $\mathrm{aug}(u, v)$ finds a 3-augmenting cycle (that is, if two arms intersect properly)

ApproxMWM($G = (V, E)$)
1. $M \leftarrow \emptyset$.
2. Repeat $\log_{14/13} \epsilon^{-1}$ times:
3. $\quad A \leftarrow$ Improve(G, M).
4. $\quad M \leftarrow M \oplus A$.
5. Return M.

Improve$(G = (V, E), M)$
Returns a set of augmentations A w.r.t. M, yielding a new matching geometrically closer to a $\frac{3}{4}$-MWM.
1. $A \leftarrow \emptyset$.
2. $V_{elig} \leftarrow V$ {arms(v) *restricted to vertices in* V_{elig}.}
3. For $e \in E \backslash M$, $r(e) \leftarrow \lfloor \log g(\text{aug}(e)) \rfloor$.
4. Repeat:
5. $\quad$ Let $e \in E \backslash M$ maximize $r(e)$.
6. $\quad$ If $r(e) > \lfloor \log g(\text{aug}(e)) \rfloor$ then $r(e) \leftarrow \lfloor \log g(\text{aug}(e)) \rfloor$.
7. $\quad$ Else $A \leftarrow A \cup \{\text{aug}(e)\}$, $V_{elig} \leftarrow V_{elig} \backslash V(\text{aug}(e))$.
8. Until $r(e) < 0$ or $V_{elig} = \emptyset$.
9. Return A.

aug(u, v)
Returns an augmentation centered at $(u, v) \in E$, using only vertices in a set $V_{elig} \subseteq V$.
1. Let $c = (u', u, v, v')$.
2. Select highest-gain arms p_1, p_2, p_3, p_4 from arms(u') and q_1, q_2, q_3, q_4 from arms(v').
3. Return the highest-gain augmentation among: $\emptyset$, the 4-cycle (u', u, v, v', u') (if it exists), and any augmentation using $p_1, \cdots, p_4, c, q_1, \ldots, q_4$.

Figure 2. Given that arms(u') and arms(v') are sorted by gain, aug(u, v) takes $O(1)$ time.

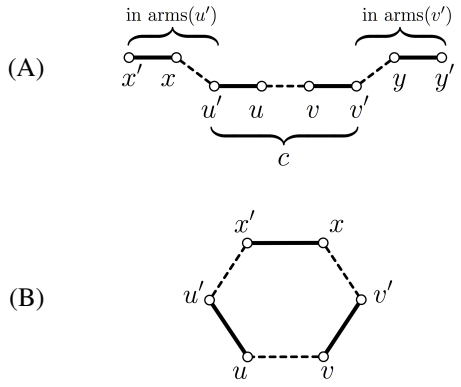

Figure 3. (A) A 3-augmenting path centered at (u, v). In general some prefix of x', x, u' and suffix of v', y, y' may not exist. (B) A 3-augmenting cycle centered at (u, v) whose arms (u', x', x) and (v', x, x') intersect at (x', x). Solid edges are in the matching.

it will return such a cycle. The procedure *Improve* (Figure 2) takes a graph and matching and returns a set A of 3-augmentations that approximates the gain of A_3 guaranteed by Lemma 6. The procedure *ApproxMWM* takes a graph and calls Improve until the current matching is a $(\frac{3}{4} - \epsilon)$-MWM. The correctness and running time of these procedures are analyzed in Lemma 7 and Theorem 2.

Lemma 7. *Let b^* be a 3-augmentation centered at (u, v) and let b be the augmentation returned by* aug(u, v)*, if any. If b^* is a path or 4-cycle then $g(b) \geq g(b^*)$. If b^* is a 3-augmenting cycle (u, v, v', x, x', u', u) then $g(b) \geq g(b^*) - w(x, x')$.*

Proof: If b^* is a 4-cycle then the claim is trivial. Suppose b^* is a path consisting of c and two arms $p^* \in \text{arms}(u')$ and $q^* \in \text{arms}(v')$ and let b consist of c and arms p_i, q_j. At most three arms in arms(v') can be incompatible with $c \cup p^*$; a symmetric statement holds for arms in arms(u') and $c \cup q^*$. Since the top four arms in arms(u') and arms(v') are considered in forming b, $g(b) \geq g(b^*)$. If b^* is a 3-augmenting cycle (u, v, v', x, x', u', u) then, by the same argument $g(p_i) \geq g(u', x', x)$ and $g(q_j) \geq g(v', x, x')$, which implies $g(b) \geq g(b^*) - w(x, x')$ since only the edge (x, x') is double counted. See Figure 3(B). ■

Theorem 2. *Let G be a graph and M^* a maximum weight matching of G. Then ApproxMWM(G) returns a matching M in $O(m \log n \log \epsilon^{-1})$ time such that $w(M) \geq (\frac{3}{4} - \epsilon) w(M^*)$.*

Proof: It suffices to show that Improve(G, M) runs in $O(m \log n)$ time and returns a set of augmentations A such that $g(A) \geq \delta(\frac{3}{4} w(M^*) - w(M))$ for some constant $\delta \geq 1/14$. The procedure maintains several invariants: (i) that arms(v) is a list of v's arms on the vertex set V_{elig}, sorted in decreasing order of gain, (ii) that A is a set of vertex disjoint augmentations on the set $V \backslash V_{elig}$, and (iii) that $r(e) \leq \log(3N) = O(\log n)$ is nondecreasing. Each $e \in E \backslash M$ is contained in 2 arms, so maintaining (i) takes $O(m \log n)$ time using any standard search tree. Each $e \in E \backslash M$ can be examined in line 5 $O(\log n)$ times before aug(e) is included in A, in line 7. The overall running time is therefore $O(m \log n)$. Let A_3 be the set of vertex disjoint augmentations guaranteed by Lemma 6, for $k = 3$, such that $g_3(A_3) \geq \frac{4}{7}(\frac{3}{4} w(M^*) - w(M))$. Define the center of a 3-augmenting cycle $a \in A_3$ as the edge in $a \backslash M$ opposite

the lightest edge in $a \cap M$. It follows from Lemma 7 that if $a \in A_3$ is centered at e, still eligible (contained in V_{elig}), and not a 3-augmenting cycle, then $g(\mathrm{aug}(e)) \geq g(a)$. If a is a 3-augmenting cycle with $w(a \cap M) < \frac{3}{4} w(a \backslash M)$ (the contrary case being trivial), Lemma 7 guarantees that:

$$\begin{aligned} g(\mathrm{aug}(e)) &\geq g(a) - w(a \cap M)/3 && (8) \\ &= w(a \backslash M) - \tfrac{4}{3} w(a \cap M) \\ &> \tfrac{3}{4} w(a \backslash M) - w(a \cap M) && (9) \\ &= g_3(a) \end{aligned}$$

Inequality (8) follows since a is centered at e and inequality (9) follows from the fact that $w(a \cap M) < \frac{3}{4} w(a \backslash M)$. Consider the moment an augmentation $\mathrm{aug}(e)$ is added to A. Since $|\mathrm{aug}(e) \cap M| \leq 4$ this augmentation intersects and makes ineligible up to four augmentations $a_1, \ldots, a_4 \in A_3$ centered at, say, $e_1, \ldots, e_4$. Since $r(e_1), \ldots, r(e_4) \leq r(e)$, $\sum_i g_3(a_i) < 8 g(\mathrm{aug}(e))$. Thus $g(A) = \sum_{a \in A} g(a) \geq \frac{1}{8} \sum_{a \in A_3} g_3(a) \geq \frac{1}{14}(\frac{3}{4} w(M^*) - w(M))$. Thus, after $\log_{14/13}(\frac{3}{4}\epsilon^{-1})$ iterations of Improve, the matching obtained is a $(\frac{3}{4} - \epsilon)$-approximate MWM. (The number of iterations can be made roughly $\log_{7/6}(\frac{1}{4}\epsilon^{-1})$ by starting with a $\frac{1}{2}$-MWM and making the base in the definition of $r(e)$ close to 1.) ■

V. Conclusion

We presented the first near-linear time $(1 - \epsilon)$-approximation algorithm for maximum weight matching (MWM), as well as a faster $(3/4 - \epsilon)$-approximation algorithm for MWM. Our results are a major improvement over the exact MWM algorithms, which run in $\Omega(m\sqrt{n})$ time [13], [16], and the previous best linear time algorithms [42], [50], which guaranteed only $(2/3 - \epsilon)$-approximations.

Our algorithm is close to optimal inasmuch as its running time can only be improved in the logarithmic factors and dependence on ϵ. However, some applications of approximate matching [35], [2], [28], [29], [24] require a high degree of parallelism, which both of our algorithms lack, as do previous $(2/3 - \epsilon)$-approximations [42], [50]. At the moment the best efficient parallel/distributed approximate MWM algorithm guarantees only $1/2$-approximations [32].

.

Acknowledgment

This work is supported by NSF CAREER grant no. CCF-0746673 and a grant from the US-Israel Binational Science Foundation.

References

[1] R. K. Ahuja, T. L. Magnati, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[2] T. Anderson, S. Owicki, J. Saxe, and C. Thacker, "High speed switch scheduling for local area networks," *ACM Trans. Comput. Syst.*, vol. 11, no. 4, pp. 319–352, 1993.

[3] M. L. Balinski, "Labelling to obtain a maximum matching," in *Combinatorial Mathematics and its Application, Eds.: R. C. Bose and T. A. Downing*. University of North Carolina Press, 1969, pp. 585–602.

[4] D. P. Bertsekas, "Auction algorithms for network flow problems: A tutorial introduction," *Computational Optimization and Applications*, vol. 1, pp. 7–66, 1992.

[5] D. Drake and S. Hougardy, "A simple approximation algorithm for the weighted matching problem," *Info. Proc. Lett.*, vol. 85, pp. 211–213, 2003.

[6] I. S. Duff and J. R. Gilbert, "Maximum-weighted matching and block pivoting for symmetric indefinite systems," in *Abstract Book of Householder Symposium XV*, 2002, pp. 73–75.

[7] J. Edmonds, "Maximum matching and a polyhedron with 0, 1-vertices," *J. Res. Nat. Bur. Standards Sect. B*, vol. 69B, pp. 125–130, 1965.

[8] ——, "Paths, trees, and flowers," *Canadian Journal of Mathematics*, vol. 17, pp. 449–467, 1965.

[9] L. Epstein, A. Levin, J. Mestre, and D. Segev, "Improved approximation guarantees for weighted matching in the semi-streaming model," in *Proceedings 27th International Symposium on Theoretical Aspects of Computer Science (STACS)*, 2010, pp. 347–358.

[10] S. Even and O. Kariv, "An $O(n^{2.5})$ algorithm for maximum matching in general graphs," in *Proceedings 16th ACM Symposium on Foundations of Computer Science*, 1975, pp. 100–112.

[11] M. L. Fredman and R. E. Tarjan, "Fibonacci heaps and their uses in improved network optimization algorithms," *J. ACM*, vol. 34, no. 3, pp. 596–615, 1987.

[12] H. N. Gabow, "An efficient implementation of Edmonds' algorithm for maximum matching on graphs," *J. ACM*, vol. 23, no. 2, pp. 221–234, 1976.

[13] ——, "Data structures for weighted matching and nearest common ancestors with linking," in *Proceedings First Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 1990, pp. 434–443.

[14] H. N. Gabow and R. E. Tarjan, "A linear-time algorithm for a special case of disjoint set union," *J. Comput. Syst. Sci.*, vol. 30, no. 2, pp. 209–221, 1985.

[15] ——, "Faster scaling algorithms for network problems," *SIAM J. Comput.*, vol. 18, no. 5, pp. 1013–1036, 1989.

[16] ——, "Faster scaling algorithms for general graph-matching problems," *J. ACM*, vol. 38, no. 4, pp. 815–853, 1991.

[17] P. Giaccone, E. Leonardi, and D. Shah, "On the maximal throughput of networks with finite buffers and its application to buffered crossbars," in *Proceedings 24th INFOCOM*, vol. 2, 2005, pp. 971–980.

[18] A. V. Goldberg and A. V. Karzanov, "Maximum skew-symmetric flows and matchings," *Math. Program., Ser. A*, vol. 100, pp. 537–568, 2004.

[19] M. Hagemann and O. Schenk, "Weighted matchings for preconditioning symmetric indefinite linear systems," *SIAM J. Sci. Comput.*, vol. 28, no. 2, pp. 403–420, 2006.

[20] S. Hanke, "Zwei approximative Algorithmen für das Matchingproblem in gewichteten Graphen," Ph.D. dissertation, Humboldt-Universität zu Berlin, 2004.

[21] S. Hanke and S. Hougardy, "New approximation algorithms for the weighted matching problem," Research Institute for Discrete Mathematics, University of Bonn, Research Report No. 101010, 2010.

[22] N. Harvey, "Algebraic algorithms for matching and matroid problems," *SIAM J. Comput.*, vol. 39, no. 2, pp. 679–702, 2009.

[23] B. Hendrickson and R. Leland, "The Chaco user's guide: Version 2.0," Sandia, Tech Report SAND94–2692, 1994.

[24] ——, "A multi-level algorithm for partitioning graphs," in *Proceedings Supercomputing (SC)*, 1995.

[25] J. E. Hopcroft and R. M. Karp, "An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs," *SIAM J. Comput.*, vol. 2, pp. 225–231, 1973.

[26] T. Kameda and J. I. Munro, "A $O(|V| * |E|)$ algorithm for maximum matching of graphs," *Computing*, vol. 12, no. 1, pp. 91–98, 1974.

[27] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM J. Sci. Comput.*, vol. 20, no. 1, pp. 359–392, 1998.

[28] ——, "Multilevel k-way partitioning scheme for irregular graphs," *J. Parallel and Distributed Comput.*, vol. 48, no. 1, pp. 96–129, 1998.

[29] ——, "A parallel algorithm for multilevel graph partitioning and sparse matrix ordering," *J. Parallel and Distributed Comput.*, vol. 48, no. 1, pp. 71–95, 1998.

[30] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, pp. 83–97, 1955.

[31] E. Lawler, *Combinatorial Optimization: Networks and Matroids*. Holt, Rinehart & Winston, New York, 1976.

[32] Z. Lotker, B. Patt-Shamir, and S. Pettie, "Improved distributed approximate matching," in *Proceedings 20th ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, 2008.

[33] L. Lovász and M. D. Plummer, *Matching Theory*. North Holland, Amsterdam, 1986, vol. 29 of Annals of Discrete Mathematics Series.

[34] A. McGregor, "Finding graph matchings in data streams," in *Proceedings 8th Int'l Workshop on Approximation Algorithms for Combinatorial Optimization Problems (RANDOM-APPROX)*, 2005, pp. 170–181.

[35] N. McKeown, "The iSLIP scheduling algorithm for input-queued switches," *IEEE/ACM Transactions on Networking*, vol. 7, no. 2, pp. 188–201, 1999.

[36] N. McKeown, V. Anantharam, and J. C. Walrand, "Achieving 100% throughput in an input-queued switch," in *INFOCOM*, 1996, pp. 296–302.

[37] J. Mestre, "Greedy in approximation algorithms," in *Proceedings 14th Annual European Symposium on Algorithms*, 2006, pp. 528–539.

[38] S. Micali and V. Vazirani, "An $O(\sqrt{|V|} \cdot |E|)$ algorithm for finding maximum matching in general graphs," in *Proc. 21st IEEE Symposium on Foundations of Computer Science (FOCS)*, 1980, pp. 17–27.

[39] M. Mucha and P. Sankowski, "Maximum matchings via Gaussian elimination," in *Proc. 45th Symp. on Foundations of Computer Science (FOCS)*, 2004, pp. 248–255.

[40] M. Olschowka and A. Neumaier, "A new pivoting strategy for Gaussian elimination," *Linear Algebra and its Applications*, vol. 240, pp. 131–151, 1996.

[41] F. Pellegrini, "SCOTCH 5.1 user's guide," LaBRI, Technical Report, September 2008.

[42] S. Pettie and P. Sanders, "A simpler linear time $2/3 - \epsilon$ approximation to maximum weight matching," *Info. Proc. Lett.*, vol. 91, no. 6, pp. 271–276, 2004.

[43] R. Preis, "Linear time $1/2$-approximation algorithm for maximum weighted matching in general graphs," in *Proc. 16th Symp. on Theoretical Aspects of Computer Science (STACS), LNCS 1563*, 1999, pp. 259–269.

[44] R. Preis and R. Diekmann, "PARTY – A software library for graph partitioning," in *Advances in Computational Mechanics with Parallel and Distributed Processing, B. H. V. Topping (ed.)*. Civil-Comp Press, 1997, pp. 63–71.

[45] M. O. Rabin and V. V. Vazirani, "Maximum matchings in general graphs through randomization," *J. Algor.*, vol. 10, no. 4, pp. 557–567, 1989.

[46] P. Sankowski, "Maximum weight bipartite matching in matrix multiplication time," *Theoretical Computer Science*, vol. 410, no. 44, pp. 4480–4488, 2009.

[47] O. Schenk, A. Wächter, and M. Hagemann, "Matching-based preprocessing algorithms to the solution of saddle-point problems in large-scale nonconvex interior-point optimization," *Comput. Optim. Appl.*, vol. 36, no. 2–3, pp. 321–341, 2007.

[48] A. Schrijver, *Combinatorial Optimization: Polyhedra and Efficiency.*, ser. Algorithms and Combinatorics. Berlin: Springer-Verlag, 2003, vol. 24.

[49] V. V. Vazirani, "A theory of alternating paths and blossoms for proving correctness of the $O(\sqrt{V}E)$ general graph maximum matching algorithm," *Combinatorica*, vol. 14, no. 1, pp. 71–109, 1994.

[50] D. E. D. Vinkemeier and S. Hougardy, "A linear-time approximation algorithm for weighted matchings in graphs," *ACM Trans. on Algorithms*, vol. 1, no. 1, pp. 107–122, 2005.

[51] C. Walshaw and M. Cross, "JOSTLE: Parallel multilevel graph-partitioning software – an overview," in *Mesh Partitioning Techniques and Domain Decomposition Techniques*, 2007, pp. 27–58.

APPENDIX

Lemma 4 *Let M be a matching, M^* the MWM, and p an integer. There exists a collection A of vertex-disjoint alternating paths/cycles w.r.t M and M^* with the following properties. Each augmentation in A has at most $2p$ edges from M^*; the ends of augmenting paths in A are either free vertices with respect to M or edges from M; $w(M \oplus A) \geq \frac{p}{p+1} w(M^*)$.*

Proof: The graph $C = M \oplus M^*$ consists of alternating paths and cycles with respect to M and M^*. We assume without loss of generality that C is a single path/cycle; our argument is applied to each separately. If C is a cycle of length $2l \geq 4p+2$, list its edges in cyclic order: $e_0, e_0^*, e_1, e_1^*, ..., e_{l-1}, e_{l-1}^*$, where $e_i \in M, e_i^* \in M^*$. Let $q = 2p$ and let $A_i, (0 \leq i \leq l-1)$ be the set of disjoint alternating paths of length $2q+1$ ending in edges in M:

$$\left\{e_i, \ldots, e_{i+q}\right\}, \left\{e_{i+q+1}, \ldots, e_{i+2(q+1)-1}\right\}, \\ \ldots, \left\{e_{i+(q+1)(\lfloor \frac{l}{q+1} \rfloor - 1)}, \ldots, e_{i+(q+1)(\lfloor \frac{l}{q+1} \rfloor)-1}\right\}$$

plus another leftover alternating path with $l \mod (q+1)$ M-edges and $\max\{0, (l \mod (q+1)) - 1\}$ M^*-edges.

Clearly $\max_i w(M \oplus A_i) \geq \sum_i w(M \oplus A_i)/l$. Note that each M^*-edge is counted $l - \lceil l/(q+1) \rceil$ times and that $M \oplus A_i$ consists solely of M^*-edges. Therefore

$$\begin{aligned} \frac{1}{l} \sum_i w(M \oplus A_i) &= \frac{1}{l} \left(l - \lceil \tfrac{l}{q+1} \rceil\right) w(M^*) \\ &\qquad \text{minimized at } l = 2p+2 \\ &\geq \frac{1}{l} \left(l - \left(2 - \tfrac{1}{p+1}\right) \tfrac{l}{2p+1}\right) w(M^*) \\ &\geq \tfrac{p}{p+1} w(M^*) \end{aligned}$$

The proof when C is a path is similar. ∎