Linear-Time Approximation for Maximum Weight Matching

RAN DUAN, Max-Planck-Institut für Informatik
SETH PETTIE, University of Michigan

The *maximum cardinality* and *maximum weight matching* problems can be solved in $\tilde{O}(m\sqrt{n})$ time, a bound that has resisted improvement despite decades of research. (Here m and n are the number of edges and vertices.) In this article, we demonstrate that this "$m\sqrt{n}$ barrier" can be bypassed by approximation. For any $\epsilon > 0$, we give an algorithm that computes a $(1 - \epsilon)$-approximate maximum weight matching in $O(m\epsilon^{-1} \log \epsilon^{-1})$ time, that is, optimal *linear time* for any fixed ϵ. Our algorithm is dramatically simpler than the best exact maximum weight matching algorithms on general graphs and should be appealing in all applications that can tolerate a negligible relative error.

Categories and Subject Descriptors: G.2.2 [**Discrete Mathematics**]: Graph Theory—*Graph algorithms*

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Matching, assignment, approximation

ACM Reference Format:
Duan, R. and Pettie, S. 2014. Linear-time approximation for maximum weight matching. *J. ACM* 61, 1, Article 1 (January 2014), 23 pages.
DOI: http://dx.doi.org/10.1145/2529989

1. INTRODUCTION

Graph matching is one of the most well-studied problems in combinatorial optimization. The original motivations of the problem were to minimize transportation costs [Hitchcock 1941; Kantorovitch 1942] and optimally assign personnel to job positions [Easterfield 1946; Thorndike 1950]. Over the years, matching algorithms have found applications in scheduling, approximation algorithms, network switching, and as key subroutines in other optimization algorithms, for example, undirected shortest paths [Lawler 1976], planar max cut [Hadlock 1975; Orlova and Dorfman 1972], Chinese postman tours [Edmonds and Johnson 1973; Kwan 1962], and metric traveling salesman [Christofides 1976]. In some applications it is not critical that the algorithm produce an exactly optimum solution. In this article, we explore the extent to which this freedom—not demanding exact solutions—allows us to design simpler and more efficient algorithms.[1]

[1]Many of the articles cited in this article can be found on the second author's homepage: web.eecs.umich.edu/~pettie/matching/.

This work was supported by NSF Career grant no. CCF-0746673, NSF grant no. CCF-1217338, and a grant from the US-Israel Binational Science Foundation. R. Duan was supported by an Alexander von Humboldt Postdoctoral Fellowship.
Authors' addresses: R. Duan, Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany; email: duanr02@gmail.com; S. Pettie, Department of Electrical Engineering and Computer Science, University of Michigan, 2260 Hayward Street, Ann Arbor, MI 48109; email: pettie@umich.edu.

DOI: http://dx.doi.org/10.1145/2529989

In order to discuss prior work with precision we must introduce some notation and terminology. The input is a weighted graph $G = (V, E, w)$ where $n = |V|$ and $m = |E|$ are the number of vertices and edges and w is the edge weight function. If w assigns integer (rather than real) weights, let N be the largest magnitude of a weight. An unweighted graph is one for which $w(e) = 1$ for all $e \in E$. A *matching* is a set of vertex-disjoint edges and a *perfect* matching is one in which all vertices are matched. The weight of a matching is the sum of its edge weights. We use MWM (and MWPM) to denote the problem of finding a maximum weight (perfect) matching, as well as the matching itself. We use MCM for the cardinality (unweighted) version of the problem. The MWPM problem on bipartite graphs is often called the *assignment* problem.

The MWPM and MWM problems are reducible to each other. Given an instance G of MWM, let G' consist of two copies of G with zero-weight edges connecting copies of the same vertex. Clearly a MWPM in G' corresponds to a pair of MWMs in G. In the reverse direction, if G is an instance of MWPM with weight function w, one can find the MWM of G using the weight function $w'(e) = w(e) + nN$. Maximum weight matchings with respect to w' necessarily have maximum cardinality. Call a matching δ-*approximate*, where $\delta \in [0, 1]$, if its weight is at least a factor δ of the optimum matching. Let δ-MWM (and δ-MCM) be the problem of finding δ-approximate maximum weight (cardinality) matching, as well as the matching itself. It is important to note that the reductions between MWPM to MWM do not work for the approximate versions of these problems since the approximation may compromise perfection. In this article, we only consider approximations of the MWM problem. Consequently, our algorithms cannot be used in applications that call for perfect matchings such as Lawler [1976], Orlova and Dorfman [1972], Hadlock [1975], Edmonds and Johnson [1973], Kwan [1962], and Christofides [1976].

Tables I, II, and III give an at-a-glance history of exact matching algorithms. Algorithms are dated according to their initial publication, and are included either because they establish a new time bound, or employ a noteworthy technique, or are of historical interest. Table IV gives a history of approximate MCM and MWM algorithms.

1.1. Algorithms for Bipartite Graphs

The MWM problem is expressible as the following integer linear program, where x represents the incidence vector of the matching.

$$\begin{aligned} \text{maximize} \quad & \sum_{e \in E} w(e)x(e) & & \\ \text{subject to} \quad & 0 \le x(e) \le 1 & \forall e \in E & \quad (1) \\ & \sum_{e=(u,u') \in E} x(e) \le 1 & \forall u \in V & \\ & x(e) \text{ is an integer} & \forall e \in E & \quad (2) \end{aligned}$$

It is well known that in bipartite graphs the integrality requirement (2) is redundant, that is, the basic feasible solutions of the LP (1) are nonetheless integral. See Birkhoff [1946] and Dantzig [1951]. The dual of (1) is

$$\begin{aligned} \text{minimize} \quad & \sum_{u \in V} y(u) & & \\ \text{subject to} \quad & y(e) \ge w(e) & \forall e \in E & \quad (3) \\ & y(u) \ge 0 & \forall u \in V & \\ \text{where, by definition,} \quad & y(u,v) \stackrel{\text{def}}{=} y(u) + y(v). & & \end{aligned}$$

Table I. Cardinality Matching

Year	Authors	Time Bound & Notes
	folklore/trivial	mn BIPARTITE
1965 1965	Edmonds Witzgall & Zahn	mn^2
1969 1976 1976 1976	Balinski Gabow Lawler Karzanov	mn or n^3
1971 1973	Hopcroft & Karp Dinic & Karzanov	$m\sqrt{n}$ BIPARTITE
1980 1991	Micali & Vazirani Gabow & Tarjan	$m\sqrt{n}$
1981	Ibarra & Moran	n^ω CARDINALITY ONLY,RANDOMIZED,BIPARTITE
1989	Rabin & Vazirani	n^ω CARDINALITY ONLY,RANDOMIZED $n^{\omega+1}$ RANDOMIZED
1991	Alt, Blum, Mehlhorn & Paul	$n\sqrt{nm/\log n}$ BIPARTITE
1991 1997	Feder & Motwani Goldberg & Kennedy	$m\sqrt{n}/\kappa$ BIPARTITE, $\kappa = \frac{\log n}{\log(n^2/m)}$
1996	Cheriyan & Mehlhorn	$n^2 + n^{5/2}/w$ BIPARTITE, w = machine word size
2004	Goldberg & Karzanov	$m\sqrt{n}/\kappa$
2004 2006	Mucha & Sankowski Harvey	n^ω RANDOMIZED

Note: Here $\omega < 2.373$ is the exponent of $n \times n$ matrix multiplication [Williams 2012]. The mn running time on general graphs depends on a special union-find data structure [Gabow and Tarjan 1985] developed later. Without it, the running time would be $mn\alpha(m, n)$, where α is the inverse-Ackermann function.

In the MWPM problem $\sum_{e=(u,u')} x(e) = 1$ holds with equality in the primal and $y(u)$ is unconstrained in the dual. Kuhn's [1955a, 1956] publication of the Hungarian method stimulated research on this problem from an algorithmic perspective, but it was not without precedent. Kuhn noted that the algorithm was latent in the work of Hungarian mathematicians König and Egerváry.[2] However, the history goes back even further. A recently rediscovered article of Jacobi from 1865 describes a variant of the Hungarian algorithm; see Ollivier [2009]. Although Kuhn's algorithm self-evidently runs in polynomial time, this mark of efficiency was noted later: Munkres [1957] showed that $O(n^4)$ time is sufficient. Independent of Kuhn's work, Gleyzal [1955] discovered a polynomial-time cycle-canceling algorithm for the assignment problem and von Neumann [1953] gave a reduction from the assignment problem to finding the optimum strategy in a zero-sum bimatrix game, which can be solved in polynomial time [Brown and von Neumann 1950].

The search for faster assignment algorithms began in earnest in the 1960s. Dinic and Kronrod [1969] gave an $O(n^3)$-time algorithm and Edmonds and Karp [1972] and Tomizawa [1971] observed that assignment is reducible to n single-source shortest

[2]A translation of Egerváry's work appears in Kuhn [1955b].

Table II. Weighted Matching: Bipartite Graphs

Year	Authors	Time Bound & Notes
1946	Easterfield	$2^n \text{poly}(n)$
1953	von Neumann	$\text{poly}(n)$
1955	Kuhn	
1955	Gleyzal	
1957	Munkres	
1964	Balinski & Gomory	
1969	Dinic & Kronrod	n^3
1970	Edmonds & Karp	$n \cdot \text{SP}^+$ $\text{SP}^+ =$ time for one SSSP computation on a non-negatively weighted graph
1971	Tomizawa	
1975	Johnson	$mn \log_d n$ $d = 2 + m/n$
1983	Gabow	$mn^{3/4} \log N$ INTEGER WEIGHTS
		$Nm\sqrt{n}$ MWM ONLY, INTEGER WEIGHTS
1984	Fredman & Tarjan	$mn + n^2 \log n$
1988	Gabow & Tarjan	$m\sqrt{n} \log(nN)$ INTEGER WEIGHTS
1992	Orlin & Ahuja	
1997	Goldberg & Kennedy	
2012	Duan & Su	
1996	Cheriyan & Mehlhorn	$n^{5/2} \log(nN)(\frac{\log\log n}{\log n})^{1/4}$ INTEGER WEIGHTS
1999	Kao, Lam, Sung & Ting	$Nm\sqrt{n}/\kappa$ MWM ONLY, INTEGER WEIGHTS
		$N(n^2 + n^{5/2}/w)$ MWM ONLY, INTEGER WEIGHTS
		Nn^ω MWM ONLY, RANDOMIZED, INTEGER WEIGHTS
2006	Sankowski	Nn^ω RANDOMIZED, INTEGER WEIGHTS
2012	Duan & Su	$m\sqrt{n} \log N$ MWM ONLY, INTEGER WEIGHTS

Note: N is the maximum integer edge weight, w is the machine word size, and $\kappa = \log n / \log(n^2/m)$. The time bounds of Johnson [1975] and Fredman and Tarjan [1987] reflect faster priority queues. The time bounds of Kao et al. [2001] reflect a reduction from MWM to N instances of MCM.

path computations on a nonnegatively weighted directed graph.[3] Using Fibonacci heaps, n executions of Dijkstra's [1959] shortest path algorithm take $O(mn + n^2 \log n)$ time. On integer weighted graphs this algorithm can be implemented slightly faster, in $O(mn + n^2 \log\log n)$ time [Han 2002; Thorup 2003] or $O(mn)$ time (randomized) [Andersson et al. 1998; Thorup 2007], independent of the maximum edge weight. Gabow and Tarjan [1989], improving an earlier algorithm of Gabow [1985b], gave a *scaling* algorithm for the assignment problem running in $O(m\sqrt{n} \log(nN))$ time, which is just a $\log(nN)$ factor slower than the fastest MCM algorithm [Hopcroft and Karp 1973; Karzanov 1973].[4] For reasonably sparse graphs Gabow and Tarjan's [1989] assignment algorithm remains unimproved. However, faster algorithms have been developed when N is small or the graph is dense [Cheriyan and Mehlhorn 1996; Kao

[3]It was known that the assignment problem is reducible to n shortest path computations on arbitrarily weighted graphs. See Ford and Fulkerson [1962], Hoffman and Markowitz [1963], and Desler and Hakimi [1969] for different reductions.

[4]Gabow and Tarjan's algorithm takes a Hungarian-type approach. The same time bound has been achieved by Orlin and Ahuja [1992] using the *auction* approach of Bertsekas [1981], by Goldberg and Kennedy [1997] using a preflow-push approach, and by Duan and Su [2012] using a primal cycle-canceling approach.

Table III. Weighted Matching: General Graphs

Year	Authors	Time Bound & Notes
1965	Edmonds	mn^2
1974 1976	Gabow Lawler	n^3
1976	Karzanov	$n^3 + mn \log n$
1978	Cunningham & Marsh	poly(n)
1982	Galil, Micali & Gabow	$mn \log n$
1985	Gabow	$mn^{3/4} \log N$ INTEGER WEIGHTS $Nm\sqrt{n}$ MWM ONLY, INTEGER WEIGHTS
1989	Gabow, Galil & Spencer	$mn \log\log\log_d n + n^2 \log n$ $d = 2 + m/n$
1990	Gabow	$mn + n^2 \log n$
1991	Gabow & Tarjan	$m\sqrt{n \log n} \log(nN)$ INTEGER WEIGHTS
2012	Huang & Kavitha	$Nm\sqrt{n}/\kappa$ MWM ONLY, INTEGER WEIGHTS
2012	Pettie	Nn^ω MWM ONLY, RANDOMIZED, INTEGER WEIGHTS
2012	Cygan, Gabow & Sankowski	Nn^ω RANDOMIZED, INTEGER WEIGHTS

Note: N is the maximum integer edge weight, ω is the exponent of $n \times n$ matrix multiplication, and $\kappa = \log n / \log(n^2/m)$. The bounds of Huang & Kavitha and Pettie reflect a reduction from MWM to N instances of MCM.

Table IV. Approximate Maximum Cardinality/Weight Matching

Year	Authors	Approx. Problem	Time Bound & Notes
1971 1973	Hopcroft & Karp Dinic & Karzanov	$(1-\epsilon)$-MCM	$m\epsilon^{-1}$ BIPARTITE
1980 1991	Micali & Vazirani Gabow & Tarjan	$(1-\epsilon)$-MCM	$m\epsilon^{-1}$

	folklore/trivial	$\frac{1}{2}$-MWM	$m \log n$
1988	Gabow & Tarjan	$(1-\epsilon)$-MWM	$m\sqrt{n} \log(n/\epsilon)$ BIPARTITE
1991	Gabow & Tarjan	$(1-\epsilon)$-MWM	$m\sqrt{n \log n} \log(n/\epsilon)$
1999 2003	Preis Drake & Hougardy	$\frac{1}{2}$-MWM	m
2003	Drake & Hougardy	$(\frac{2}{3}-\epsilon)$-MWM	$m\epsilon^{-1}$
2004	Pettie & Sanders	$(\frac{2}{3}-\epsilon)$-MWM	$m \log \epsilon^{-1}$
2010 2010	Duan & Pettie Hanke & Hougardy	$(\frac{3}{4}-\epsilon)$-MWM	$m \log n \log \epsilon^{-1}$
new		$(1-\epsilon)$-MWM	$m\epsilon^{-1} \log \epsilon^{-1}$ ARBITRARY WEIGHTS $m\epsilon^{-1} \log N$ INTEGER WEIGHTS

Note: N is the maximum integer edge weight and $\epsilon > 0$ is arbitrary.

et al. 2001; Sankowski 2009], or when an MWM is sought [Duan and Su 2012], or when a matching of a specified size is sought [Ramshaw and Tarjan 2012]. Of particular interest is Sankowski's algorithm [2009], which solves MWPM in $O(Nn^\omega)$ time, where ω is the exponent of square matrix multiplication.

1.2. Algorithms for General Graphs

Whereas the basic solutions to (1,3) are integral on bipartite graphs, the same is not true for general graphs. For example, if the graph is a unit-weighted cycle with length $2k + 1$ the MWM has weight k but (1) achieves its maximum of $k + 1/2$ by setting $x(e) = 1/2$ for all $e \in E$. Let $\mathcal{V}_{odd}$ be the set of all odd-size subsets of V. Clearly every feasible solution to the integer linear program (1,2) also satisfies the following odd-set constraints.

$$\sum_{e \in E(B)} x(e) \leq (|B| - 1)/2 \qquad \forall B \in \mathcal{V}_{odd} \tag{4}$$

Edmonds [1965a, 1965b] proved that if we replace the integrality constraint (2) with (4), the basic solutions to the resulting LP are integral.[5] Edmonds' algorithm mimics the structure of the Hungarian algorithm but the search for augmenting paths is complicated by the presence of odd-length alternating cycles and the fact that matched edges must be searched in both directions. Edmonds' solution is to contract *blossoms* as they are encountered. A blossom is defined inductively as an odd-length cycle alternating between matched and unmatched edges, whose components are either single vertices or blossoms in their own right. Blossoms are discussed in detail in Section 2.1.

The fastest implementation of Edmonds' algorithm, due to Gabow [1990], runs in $O(mn + n^2 \log n)$ time, which matches the running time of the best bipartite MWPM algorithm [Fredman and Tarjan 1987]. Gabow and Tarjan [1991] extended their scaling algorithm for MWPM to general graphs, achieving a running time of $O(m\sqrt{n \log n} \log(nN))$, which is the fastest known algorithm for integer-weighted graphs and nearly matches the $O(m\sqrt{n})$ time bound of the best MCM algorithms [Micali and Vazirani 1980; Vazirani 1994].[6] As in the bipartite case, faster algorithms for MWM and MWPM are known when the graph is dense or N is small. Huang and Kavitha [2012] and Pettie [2012] gave reductions from MWM to N instances of MCM, which can be used in conjunction with the MCM algorithms of Mucha and Sankowski [2004], Harvey [2009], and Goldberg and Karzanov [2004]. Cygan et al. [2012] recently showed that MWPM on general graphs is solvable in $O(Nn^{\omega})$ time, matching the time bound of Sankowski [2009] for bipartite graphs.

1.3. Approximating Weighted Matching

The approximate MWM problem is remarkable in that it has been studied for decades, has practical applications, and yet as late as 1999, essentially nothing better than the greedy algorithm was known.[7] Moreover, the $(1 - \epsilon)$-MCM problem had been solved satisfactorily in the early 1970s. Although not stated as such, the $O(m\sqrt{n})$-time exact MCM algorithms [Dinic 1970; Hopcroft and Karp 1973; Karzanov 1973; Micali and Vazirani 1980] are actually $(1 - \epsilon)$-MCM algorithms running in $O(m\epsilon^{-1})$ time. These

[5] In the MWPM problem $\sum_{e=(u,u')} x(e) = 1$, for all $u \in V$, and we have the freedom to use an alternative variety of odd-set constraints, namely, $\sum_{e=(u,v) \in E\,:\, u \in B, v \notin B} x(e) \geq 1$, $\forall B \in \mathcal{V}_{odd}$.

[6] Gabow and Tarjan [1991] claim a running time of $O(m\sqrt{n \log n \alpha(m,n)} \log(nN))$, where the $\alpha(m,n)$ factor comes from an $O(m\alpha(m,n))$ implementation of the *split-findmin* data structure [Gabow 1985a]. This can be reduced to $O(m \log \alpha(m,n))$ [Pettie 2005]. Thorup [1999] noted that on integer-weighted inputs, split-findmin can be implemented in $O(m)$ time on a word-RAM.

[7] The greedy algorithm repeatedly includes the heaviest edge in the matching and removes all incident edges. Gabow and Tarjan [1989, 1991] observed that by retaining the $O(\log(n/\epsilon))$ high-order bits of the edge weights, their exact scaling algorithms become $\tilde{O}(m\sqrt{n})$-time $(1 - \epsilon)$-MWM algorithms for bipartite and general graphs.

algorithms are based on three observations (i) a maximal set of shortest augmenting paths can be found in linear time, (ii) augmenting along such a set increases the length of the shortest augmenting path, and (iii) that after k rounds of such augmentations the resulting matching is a $(1 - \frac{1}{k+1})$-MCM.

Preis [1999] gave a linear-time $\frac{1}{2}$-MWM algorithm, which improves on the greedy algorithm's $O(m \log n)$ running time but not its approximation guarantee. Drake and Hougardy (see Vinkemeier and Hougardy [2005]) presented the first linear time algorithm with an approximation guarantee greater than 1/2. Specifically, they gave a $(\frac{2}{3} - \epsilon)$-MWM algorithm running in $O(m\epsilon^{-1})$ time, for any $\epsilon > 0$. The dependence on ϵ was later improved by Pettie and Sanders [2004]. These algorithms are based on a weighted version of Hopcroft and Karp's [1973] argument, namely that any matching whose weight-augmenting paths *and cycles* have at least k unmatched edges is necessarily a $(1 - \frac{1}{k})$-MWM. Duan and Pettie [2010] and Hanke and Hougardy [2010] presented $(\frac{3}{4} - \epsilon)$-MWM algorithms running in time $O(m \log n \log \epsilon^{-1})$.[8]

1.4. New Results

We present the first $(1 - \epsilon)$-MWM algorithm that significantly improves on the $\tilde{O}(m\sqrt{n})$ running times of Gabow and Tarjan [1989, 1991]. Our algorithm runs in $O(m\epsilon^{-1} \log \epsilon^{-1})$ time on general graphs and $O(m\epsilon^{-1} \cdot \min\{\log \epsilon^{-1}, \log N\})$ time on integer-weighted general graphs. This is optimal for any fixed ϵ and near-optimal as a function of ϵ, given the state-of-the-art in MCM algorithms.[9] Moreover, our algorithm is as *simple* as one could reasonably hope for. Its search for augmenting paths uses depth first search [Gabow and Tarjan 1991, Sect. 8] rather than the double depth first search of Micali and Vazirani [1980]. It uses no priority queues, split-findmin structures [Gabow 1985a], or the blossom "shells" that arise from Gabow and Tarjan's [1991] scaling technique.

1.5. Remarks on Approximate Weighted Matching and Its Applications

Our focus is on algorithms that accept *arbitrary* input graphs and that give provably good *worst-case* approximations. These twin objectives are self-evidently attractive, yet nearly all work on approximate weighted matching (prior to Preis [1999]) focused on specialized cases or weaker approximation guarantees. Early work on the problem usually considered complete bipartite graphs, and confirmed the efficiency of heuristics either experimentally or analytically with respect to inputs over some natural distribution [Avis 1978; Brogden 1946; Kuhn and Baumol 1962; Kurtzberg 1962; Motzkin 1956; Thorndike 1950]. See Avis [1983] for a more detailed discussion of heuristics.

Most work in the area considers graphs defined by metrics, often Euclidean metrics. Reingold and Tarjan [1981] proved that the greedy algorithm for metric MWPM[10] has an approximation ratio of $\approx n^{\log \frac{3}{2}} > n^{0.58}$. Goemans and Williamson [1995] gave a 2-approximation for metric MWPM that can be implemented in $O(n^2)$ time [Gabow and Pettie 2002], or $O(m \log^2 n)$ time [Cole et al. 2001] in metrics defined by m-edge

[8]Hanke and Hougardy [2010] also claimed a $(\frac{4}{5} - \epsilon)$-MWM algorithm running in $O(m \log^2 n \log \epsilon^{-1})$ time, though the details were not substantiated.

[9]Note that any $(1 - \epsilon)$-MWM algorithm running in $O(f(\epsilon)m)$ time yields an *exact* MCM algorithm running in $O(m \cdot (f(\epsilon) + \epsilon n))$ time, for any ϵ. Thus, any $(1 - \epsilon)$-MWM algorithm running in $o(m\epsilon^{-1})$ time would improve the $O(m\sqrt{n})$ MCM algorithms [Dinic 1970; Hopcroft and Karp 1973; Karzanov 1973; Micali and Vazirani 1980].

[10]For metric inputs let MWPM be the minimum-weight perfect matching problem.

graphs. The Euclidean MWPM comes in two flavors: the monochromatic version is given $2n$ points and the bichromatic version is given $2n$ points, n of which are colored blue, the rest red, where the matching cannot include monochromatic edges.[11] Both the mono- and bichromatic variants of 2D Euclidean MWPM can be $(1+\epsilon)$-approximated in $O(n\,\mathrm{poly}(\epsilon^{-1}, \log n))$ time [Sharathkumar and Agarwal 2012; Varadarajan and Agarwal 1999]. Some work considers the even more specialized case of Euclidean matching in the unit square, which allows for algorithms that guarantee absolute upper bounds on the weight of the matching; see Iri et al. [1983], Reingold and Supowit [1983], Avis [1983] and the references therein.

There are several applications of MWM (on general or bipartite graphs) in which one would gladly sacrifice matching quality for speed. 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 [McKeown 1999] and PIM [Anderson et al. 1993], guarantee $\frac{1}{2}$-MCMs and it has been shown [Giaccone et al. 2005; McKeown et al. 1996] that (approximate) MWMs have good throughput guarantees, where edge weights are based on queue-length. See also Leonardi et al. [2003], Shah and Kopikare [2002], and Shah et al. [2002]. Approximate MWM algorithms are a component in several multilevel graph clustering libraries.[12] (PARTY, for example, builds a hierarchical clustering by iteratively finding and contracting approximate MWMs; see Preis and Diekmann [1997].) Approximate MWM algorithms are used as a heuristic preprocessing step in several sparse linear system solvers [Duff and Gilbert 2002; Hagemann and Schenk 2006; Olschowka and Neumaier 1996; Schenk et al. 2007]. The goal is to permute the rows/columns to maximize the weight on or near the main diagonal.

1.6. Organization

Section 2 introduces some notation, states well-known properties of augmenting paths and blossoms, and reviews Edmonds' optimality conditions for weighted matching. In Section 3, we present our $(1-\epsilon)$-MWM algorithms.

2. PRELIMINARIES

We use $E(H)$ and $V(H)$ to refer to the edge and vertex sets of H or the graph induced by H, that is, $V(E')$ is the set of endpoints of $E' \subseteq E$ and $E(V')$ is the edge set of the graph induced by $V' \subseteq V$. A *matching* M is a set of vertex-disjoint edges. Vertices not incident to an M edge are *free*. An alternating path (or cycle) is one whose edges alternate between M and $E \backslash M$. An alternating path P is *augmenting* if P begins and ends at free vertices, that is, $M \oplus P \stackrel{\text{def}}{=} (M \backslash P) \cup (P \backslash M)$ is a matching with cardinality $|M \oplus P| = |M| + 1$.

Because we are only seeking a $(1-\epsilon)$-approximate solution, we can afford to scale and round edge weights to small integers. To see this, observe that the weight of the MWM is at least $w_{\max} = \max\{w(e) \mid e \in E(G)\}$. It suffices to find a $(1-\epsilon/2)$-MWM M with respect to the weight function $\tilde{w}(e) = \lfloor w(e)/\gamma \rfloor$ where $\gamma = \epsilon \cdot w_{\max}/n$. Note that

[11] The weight of the bichromatic MWPM is also known as the *earth mover distance* between the red and blue points.

[12] For example, METIS [Karypis and Kumar 1998], PARTY [Preis and Diekmann 1997], PT-SCOTCH [Pellegrini 2008] CHACO [Hendrickson and Leland 1995], JOSTLE [Walshaw and Cross 2007], and KaFFPa/KaFFPaE [Holtgrewe et al. 2010; Sanders and Schulz 2012].

$w(e) - \gamma < \gamma \cdot \tilde{w}(e) \leq w(e)$ for any e. Define M^* and $\tilde{M}^*$ to be the MWMs with respect to w and $\tilde{w}$. It follows that:

$$\begin{aligned}
w(M) &\geq \gamma \cdot \tilde{w}(M) && \text{Defn. of } \tilde{w} \\
&\geq \gamma \cdot (1-\epsilon/2) \cdot \tilde{w}(\tilde{M}^*) && \text{Defn. of } M \\
&\geq \gamma \cdot (1-\epsilon/2) \cdot \tilde{w}(M^*) && \text{Defn. of } \tilde{M}^* \\
&> \gamma \cdot (1-\epsilon/2) \cdot (\gamma^{-1} \cdot w(M^*) - |M^*|) && \text{Defn. of } \tilde{w} \\
&\geq (1-\epsilon/2) \cdot (w(M^*) - \gamma n/2) && |M^*| \leq n/2 \\
&= (1-\epsilon/2) \cdot (w(M^*) - \epsilon \cdot w_{\max}/2) && \text{Defn. of } \gamma \\
&> (1-\epsilon) \cdot w(M^*) && w(M^*) \geq w_{\max}\,.
\end{aligned}$$

Since it is better to use an exact MWM algorithm when $\epsilon < 1/n$ [Gabow 1990; Gabow and Tarjan 1991], we assume, henceforth, that $w : E \to \{1, 2, \ldots, N\}$, where $N \leq n^2$ is the maximum integer edge weight. For notational convenience, we also assume that N is a power of 2.

2.1. Blossoms and the LP Formulation of MWM

The dual LP of (1,4) is

$$\begin{aligned}
\text{minimize} \quad & \sum_{u \in V(G)} y(u) + \sum_{B \in \mathcal{V}_{odd}} \frac{|B|-1}{2} \cdot z(B) && \\
\text{subject to} \quad & yz(e) \geq w(e) && \forall e \in E(G) \\
& y(u) \geq 0, z(B) \geq 0 && \forall u \in V(G), \forall B \in \mathcal{V}_{odd}
\end{aligned}$$

$$\text{where, by definition,} \quad yz(u,v) \stackrel{\text{def}}{=} y(u) + y(v) + \sum_{\substack{B \in \mathcal{V}_{odd}, \\ (u,v) \in E(B)}} z(B)\,.$$

Note that y and z map vertices and odd sets to their dual values. It is convenient to create a dual $yz(e)$ for each edge e, defined to be the sum of the y-values of its endpoints and the z-values of all odd sets that contain e.

Despite the exponential number of primal constraints and dual z-variables, Edmonds [1965a] demonstrated that an optimum matching could be found in polynomial time by maintaining information (z-values) on no more than $n/2$ elements of $\mathcal{V}_{odd}$ at any given time. At intermediate stages of Edmonds' algorithm [1965a], there is a matching M and a laminar (nested) subset $\Omega \subseteq \mathcal{V}_{odd}$ of *blossoms*. A blossom is identified with a vertex set $B \in \mathcal{V}_{odd}$ and an edge set E_B on B. If $v \in V$, then $B = \{v\}$ is a trivial blossom with $E_B = \emptyset$. Suppose there is an odd-length sequence of blossoms $A_0, A_1, \ldots, A_\ell$ with associated edge sets $E_{A_0}, \ldots, E_{A_\ell}$. If the $\{A_i\}$ are connected in a cycle by edges $e_0, \ldots, e_\ell$, where $e_i \in A_i \times A_{i+1}$ (modulo $\ell+1$), then $B = \bigcup_i A_i$ is also a blossom associated with edge set $E_B = \bigcup_i E_{A_i} \cup \{e_0, \ldots, e_\ell\}$. A short proof by induction shows that $|B|$ is odd. A blossom B is *full* with respect to a matching M if $|M \cap E_B| = (|B|-1)/2$, that is, $M \cap E_B$ matches all vertices except one, which is called the *base* of B. Only full blossoms are included in Ω. Note that $E(B) = E \cap (B \times B)$ may contain many edges outside of E_B.

A key property of blossoms is that any vertex $b \in B$ can be made the base by choosing a suitable matching M on E_B. Suppose, without loss of generality, that $b \in A_0$ and let $e_i = (u_i, v_{i+1})$ (modulo $\ell+1$), where $u_i, v_i \in A_i$. Recursively find matchings in $\{E_{A_i}\}$ using b as the base of A_0, u_i as the base of A_i when i is odd, and v_i as the base of A_i when $i > 0$ is even. At this point the only unmatched vertices are the bases

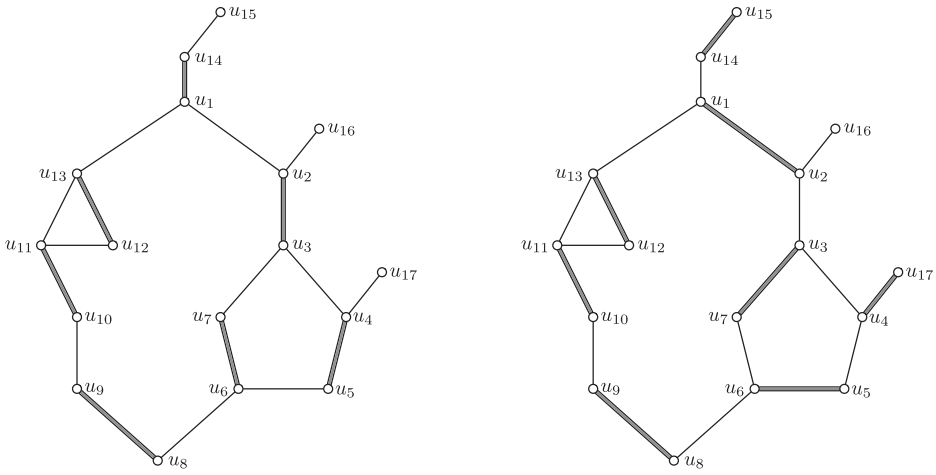

Fig. 1. Thick edges are matched, thin unmatched. (a) A blossom $B_1 = (u_1, u_2, B_2, u_8, u_9, u_{10}, B_3)$ with base u_1 containing nontrivial sub-blossoms $B_2 = (u_3, u_4, u_5, u_6, u_7)$ with base u_3 and $B_3 = (u_{11}, u_{12}, u_{13})$ with base u_{11}. Vertices u_{15}, u_{16}, and u_{17} are free. The path $(u_{16}, u_2, u_3, u_7, u_6, u_5, u_4, u_{17})$ is an example of an augmenting path that exists in G but not G/B_1, the graph obtained by contracting B_1. (b) The situation after augmenting along $(u_{15}, u_{14}, B_1, u_{17})$ in G/B_1, which corresponds to augmenting along $(u_{15}, u_{14}, u_1, u_2, u_3, u_7, u_6, u_5, u_4, u_{17})$ in G. After augmentation B_1 and B_2 have their base at u_4. If, instead, we augmented along $(u_{15}, u_{14}, B_1, u_{16})$, corresponding to the path $(u_{15}, u_{14}, u_1, u_{13}, u_{12}, u_{11}, u_{10}, u_9, u_8, u_6, u_7, u_3, u_2, u_{16})$, the base of B_1 would be relocated to u_2 and the bases of B_2 and B_3 would be relocated to u_6 and u_{13}, respectively.

$\{b\} \cup \{u_i\}_{i\,\text{odd}} \cup \{v_i\}_{i\,\text{even}}$. Include in the matching the edges $\{(u_i, v_{i+1})\}_{i\,\text{odd}}$, which leaves only b unmatched. (See Figure 1.) The blossom structure guarantees that there is an even-length alternating path from b to every other vertex c in B. Inductively assume this claim holds for $A_0, \ldots, A_\ell$. Suppose $c \in A_i$. If i is even, take a path from b to c via edges $(u_0, v_1), (u_1, v_2), \ldots, (u_{i-1}, v_i)$. The inductive hypothesis gives us even-length alternating paths from b to u_0 in E_{A_0}, from v_1 to u_1 in E_{A_1}, and so on. If i is odd, we go around the cycle in the other direction, taking a path from b to c via edges $(v_0, u_\ell), (v_\ell, u_{\ell-1}), \ldots, (v_{i+1}, u_i)$. (See Figure 1.)

Matching algorithms represent a nested set Ω of full, *active* blossoms by rooted trees, where leaves represent vertices and internal nodes represent nontrivial blossoms. A *root blossom* is one not contained in any other blossom. The children of an internal node representing B are ordered according to the odd cycle that formed B, where one child is distinguished as containing the base of B. As we will see, it is often possible to treat blossoms as if they were single vertices. Let the *contracted graph* G/Ω be obtained by contracting all root blossoms and removing spurious edges. 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 $A_0, \ldots, A_\ell$. Lemma 2.1 summarizes some well-known properties of blossoms and the contracted graph.

LEMMA 2.1. *Let Ω be a set of full blossoms with respect to a matching M and let $B \in \Omega$ be a root blossom.*

(1) *If M is a matching in G, then M/Ω is a matching in G/Ω.*
(2) *Every augmenting path P' relative to M/Ω in G/Ω extends to an augmenting path P relative to M in G. (That is, P is obtained from P' by substituting for each nontrivial blossom vertex B in P' a path through E_B. See Figure* 1*.)*

(3) *Let P' and P be as mentioned in* (2). Then, Ω remains a valid set of full blossoms (possibly with different bases) with respect to the augmented matching $M \oplus P$. (See Figure 1.)

PROOF. Part (1) follows from the fact that $M \cap E_B$ leaves only one vertex in B unmatched, namely its base, implying B is incident to at most one edge in G/Ω. For Part (2) consider an augmenting path $P' = (B_0, B_1, \ldots, B_k)$ in G/Ω. The matched edge in G corresponding to (B_i, B_{i+1}), where i is odd, necessarily connects the bases of B_i and B_{i+1}. We can therefore extend P' to an augmenting path P in G by substituting for each B_i a suitable even-length alternating path through E_{B_i}, one endpoint of which is the base of B_i. Turning to Part (3), suppose P enters a blossom B at its base b and leaves at a vertex c. It follows that b is either free or matched to a vertex outside of B. After augmentation b will be matched and c will be matched to a vertex outside of B, that is, augmenting along P shifts the base from b to c. Augmenting along P may also shift the bases of other blossoms contained in B. For example, suppose $b \in A_0$, $c \in A_i$, and i is even. (As defined earlier, B is composed of sub-blossoms $\{A_i\}$ connected by cycle edges $\{(u_i, v_{i+1})\}$.) The portion of P in E_B traverses cycle edges $(u_0, v_1), \ldots, (u_{i-1}, v_i)$ and even-length alternating paths in each of $A_0, A_1, \ldots, A_i$; augmenting along P relocates all of their bases. For example, the base of A_0 is relocated from b to u_0 and the base of A_1 from u_1 to v_1, since $(u_0, v_1) \in M \oplus P$. □

Implementations of Edmonds' algorithm grow a matching M while maintaining Property 2.2, which controls the relationship between M, Ω and the dual variables.

Property 2.2. (*Feasibility and Complementary Slackness Conditions*)

(1) *Nonnegativity*. $z(B) \geq 0$ for all $B \in \mathcal{V}_{odd}$ and $y(u) \geq 0$ for all $u \in V(G)$. If $y(u) > 0$, then u is matched.
(2) *Active Blossoms*. Ω contains all B with $z(B) > 0$ and all root blossoms B have $z(B) > 0$. (Non-root blossoms may have zero z-values.)
(3) *Domination*. $yz(e) \geq w(e)$ for all $e \in E$.
(4) *Tightness*. $yz(e) = w(e)$ when $e \in M$ or $e \in E_B$ for some $B \in \Omega$.

If the y-values of free vertices become zero, it follows from domination and tightness that M is a maximum weight matching, as the following short proof attests. Here M^* is any maximum weight matching.

$$
\begin{aligned}
w(M) = \sum_{e \in M} w(e) \; &= \sum_{e \in M} yz(e) && \text{tightness} \\
&= \sum_{u \in V(G)} y(u) + \sum_{B \in \Omega} \frac{|B| - 1}{2} \cdot z(B) && \text{free } y\text{-values, defn. of } yz \\
&\geq \sum_{u \in V(M^*)} y(u) + \sum_{B \in \Omega} |E(B) \cap M^*| \cdot z(B) && y, z \text{ nonnegative} \\
&= \sum_{e \in M^*} yz(e) \; \geq \; w(M^*) && \text{domination}
\end{aligned}
$$

Our approximate MWM algorithms are based on the observation that if domination and tightness are satisfied up to a $1 \pm \epsilon$ factor, then the given matching is a $(1 - O(\epsilon))$-MWM.

LEMMA 2.3. *Let M and Ω be a matching and set of nested blossoms, and let y and z assign their dual values. Suppose y, z satisfy Property* 2.2(1,2) *and satisfy Property* 2.2(3,4) *in the following approximate sense. For all e, $yz(e) \geq (1 - \epsilon_0) \cdot w(e)$, and for*

all $e \in M \cup \bigcup_{B\in\Omega} E_B$, $yz(e) \le (1+\epsilon_1)\cdot w(e)$. *If the y-values of free vertices are zero,* M *is a* $\left((1+\epsilon_1)^{-1}(1-\epsilon_0)\right)$-MWM.

PROOF. From the definitions of approximate tightness and approximate domination, we have the following.

$$\begin{aligned}
w(M) &= \sum_{e\in M} w(e) \ \ge (1+\epsilon_1)^{-1} \sum_{e\in M} yz(e) && \text{approx. tightness}\\
&= (1+\epsilon_1)^{-1}\left(\sum_{u\in V(G)} y(u) + \sum_{B\in\Omega} \frac{|B|-1}{2}\cdot z(B)\right) && \text{free } y\text{-values, defn. of } yz\\
&\ge (1+\epsilon_1)^{-1}\left(\sum_{u\in V(M^*)} y(u) + \sum_{B\in\Omega} |E(B)\cap M^*|\cdot z(B)\right) && y,z \text{ nonnegative}\\
&= (1+\epsilon_1)^{-1}\cdot \sum_{e\in M^*} yz(e) && \text{defn. of } yz\\
&\ge (1+\epsilon_1)^{-1}(1-\epsilon_0)\cdot w(M^*) && \text{approx. domination}
\end{aligned}$$

□

3. A SCALING ALGORITHM FOR APPROXIMATE MWM

Our algorithm maintains a *dynamic* relaxation of the feasibility and complementary slackness conditions. In the beginning, *domination* is weak but becomes progressively tighter at each scale whereas *tightness* is weakened at each scale, though not uniformly. The degree to which a matched edge or blossom edge may violate tightness depends on the scale when it last entered the blossom or matching.

Recall that N is the maximum integer edge weight. The parameter $\epsilon' = \Theta(\epsilon)$ will be selected later to guarantee that the final matching is a $(1-\epsilon)$-MWM. Henceforth, assume that $N \ge 1$ and $\epsilon' \le 1/4$ are powers of two. Define $\delta_0 = \epsilon' N$ and $\delta_i = \delta_0/2^i$. At scale i we use the truncated weight function $w_i(e) = \delta_i \lfloor w(e)/\delta_i \rfloor$. Note that $w_{i+1}(e) = w_i(e)$ or $w_i(e) + \delta_{i+1}$.

Property 3.1. (*Relaxed Feasibility and Complementary Slackness*). There are $L+1$ scales numbered $0,\ldots,L$, where $L = \log N$. Let $i \in [0,L]$ be the current scale.

(1) *Granularity*. $z(B)$ is a nonnegative multiple of δ_i, for all $B \in \mathcal{V}_{odd}$, and $y(u)$ is a nonnegative multiple of $\delta_i/2$, for all $u \in V(G)$.
(2) *Active Blossoms*. Ω contains all B with $z(B) > 0$ and all root blossoms B have $z(B) > 0$. (Non-root blossoms may have zero z-values.)
(3) *Near Domination*. $yz(e) \ge w_i(e) - \delta_i$ for all $e \in E$.
(4) *Near Tightness*. Call a matched or blossom edge *type j* if it was last made a matched or blossom edge in scale $j \le i$. (That is, it entered the set $M \cup \bigcup_{B\in\Omega} E_B$ in scale j and has remained in that set, even as M and Ω evolve as augmenting paths are found and blossoms are formed and dissolved.) If e is such a type j edge, then $yz(e) \le w_i(e) + 2(\delta_j - \delta_i)$.
(5) *Free Vertex Duals*. The y-values of free vertices are equal and strictly less than the y-values of matched vertices.

Property 3.1's definitions of near domination and near tightness differ from their analogues in previous (exact) scaling algorithms. The differences stem from the halting conditions for a scale and how dual variables are adjusted between scales. In Gabow and Tarjan's [1991] algorithm, for example, scale i begins by *forgetting* the matching

and blossom edges from the previous scale (though not the nested structure of the old blossoms), replacing the weight function w_{i-1} by w_i, and adjusting the duals to guarantee domination with respect to w_i.[13] Since the matching and blossom edges are forgotten, near tightness holds trivially. The scale ends once the algorithm finds a *perfect* matching, blossoms, and duals y, z that satisfy tightness to within δ_i.

For subsequent scales of the Gabow-Tarjan algorithm to be efficient, it is critical that each scale ends with a perfect matching and nearly optimum duals. Our algorithm, however, is obliged to run in linear time (for fixed ϵ) and therefore cannot afford to find a perfect matching, nor can it afford to discard the matching computed by one scale before proceeding to the next. The goal of a scale must be different. Consider the difficulty of preserving Property 3.1(3,4) (near tightness and near domination) across multiple scales. At the end of scale $i-1$, we have $yz(e) \geq w_{i-1}(e) - \delta_{i-1}$ whereas at the beginning of scale i we require $yz(e) \geq w_i(e) - \delta_i$. We force near domination to be satisfied by simply having each vertex add δ_i to its y-value. However, doing this effectively *weakens* near-tightness on matched and blossom edges by $2\delta_i$. If an edge e enters the matching at scale j and witnesses $i-j$ scale changes, in scale i it may violate tightness by as much as $2\delta_{j+1} + \cdots + 2\delta_i = 2(\delta_j - \delta_i)$. The question is whether violations of tightness of this magnitude are tolerable.

In order to invoke Lemma 2.3, we need to write Property 3.1(3,4) as multiplicative $(1-\epsilon_0)$- and $(1+\epsilon_1)$-approximations, for some $\epsilon_0, \epsilon_1 = O(\epsilon)$. The algorithm halts after scale L. At this time $w_L = w$, $\delta_L = \epsilon'$ and the y-values of free vertices are zero. Since weights are positive integers the near dominance condition $yz(e) \geq w_L(e) - \delta_L = w(e) - \epsilon'$ implies $yz(e) \geq (1-\epsilon')w(e)$. We will show later that the near-tightness condition $yz(e) \leq w_i(e) + 2(\delta_j - \delta_i)$ implies $yz(e) \leq (1+\epsilon_1)w(e)$ for an $\epsilon_1 = O(\epsilon')$.

3.1. The Scaling Algorithm

Initially, $M = \emptyset, \Omega = \emptyset$, and $y(u) = N/2 - \delta_0/2$ for all $u \in V$, which clearly satisfies Property 3.1 for scale $i = 0$, since $yz(e) = 2(N/2 - \delta_0/2) \geq w_0(e) - \delta_0$. The algorithm, given in Figure 2, consists of scales $0, \ldots, L = \log N$, where the purpose of scale i is to halve the y-values of free vertices while maintaining Property 3.1. In each iteration of scale i, the algorithm (1) augments a maximal set of vertex-disjoint augmenting paths of *eligible* edges, (2) finds and contracts blossoms of *eligible* edges, (3) performs dual adjustments on y- and z-values, and (4) dissolves previously contracted root blossoms if their z-values become zero. Each dual adjustment step decrements by $\delta_i/2$ the y-values of free vertices. Thus, there are roughly $(N/2^{i+2})/(\delta_i/2) = O(\epsilon^{-1})$ iterations per scale, independent of i. The efficiency and correctness of the algorithm depend on *eligibility* being defined properly.

Definition 3.2. At scale i, an edge e is *eligible* if at least one of the following hold.

(i) $e \in E_B$ for some $B \in \Omega$.
(ii) $e \notin M$ and $yz(e) = w_i(e) - \delta_i$.
(iii) $e \in M$ and $yz(e) - w_i(e)$ is a nonnegative integer multiple of δ_i.

Let E_{elig} be the set of eligible edges and let $G_{elig} = (V, E_{elig})/\Omega$ be the unweighted graph obtained by discarding ineligible edges and contracting root blossoms.

Criterion (i) for eligibility simply ensures that an augmenting path in G_{elig} extends to an augmenting path of eligible edges in G. A key implication of Criteria (ii) and (iii) is that if P is an augmenting path in G_{elig}, every edge in P becomes ineligible

[13] By "w_i" and "δ_i", we mean the weight function and granularity used in the ith scale of their algorithm. Their initial granularity δ_0 differs from ours.

Initialization:

$M \leftarrow \emptyset$ — no matched edges

$\Omega \leftarrow \emptyset$ — no blossoms

$\delta_0 \leftarrow \epsilon' N$ — $\epsilon' = \Theta(\epsilon)$ is a parameter

$y(u) \leftarrow \frac{N}{2} - \frac{\delta_0}{2}$, for all $u \in V(G)$ — satisfies Property 3.1(1,3)

Execute scales $i = 0, \ldots, L = \log N$ and return the matching M.

Scale i:

— Repeat the following four steps until y-values of free vertices reach $N/2^{i+2} - \delta_i/2$, if $i \in [0, L)$, or until they reach zero, if $i = L$.

(1) **Augmentation:**
Find a maximal set Ψ of vertex-disjoint augmenting paths in G_{elig} and set $M \leftarrow M \oplus (\bigcup_{P \in \Psi} P)$. Update G_{elig}.

(2) **Blossom Shrinking:**
Let $V_{out} \subseteq V(G_{elig})$ 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}. (That is, if $(u, v) \in E(G_{elig}) \backslash M$ and $u, v \in V_{out}$, then u and v must be in a common blossom.) Let $V_{in} \subseteq V(G_{elig}) \backslash V_{out}$ be those non-V_{out}-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_{elig}.

(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:

$y(u) \leftarrow y(u) - \delta_i/2$, if $u \in \hat{V}_{out}$.

$y(u) \leftarrow y(u) + \delta_i/2$, if $u \in \hat{V}_{in}$.

$z(B) \leftarrow z(B) + \delta_i$, if $B \in \Omega$ is a root blossom with $B \subseteq \hat{V}_{out}$.

$z(B) \leftarrow z(B) - \delta_i$, if $B \in \Omega$ is a root blossom with $B \subseteq \hat{V}_{in}$.

(4) **Blossom Dissolution:**
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_{elig} by the new Ω.

— Prepare for the next scale, if $i \in [0, L)$:

$\delta_{i+1} \leftarrow \delta_i/2$

$y(u) \leftarrow y(u) + \delta_{i+1}$, for all $u \in V(G)$ — free vertices' y-values are $N/2^{i+2}$.

Fig. 2. The scaling algorithm.

after augmentation. This follows from the fact that eligible unmatched edges must have $yz(e) - w_i(e) < 0$ whereas eligible matched edges must have $yz(e) - w_i(e) \geq 0$. Regarding Criterion (iii), note that Property 3.1 (granularity and near domination) implies that $yz(e) - w_i(e)$ is at least $-\delta_i$ and an integer multiple of $\delta_i/2$.

3.2. Analysis and Correctness

The aim of this section is to prove Lemma 3.5, which states that the algorithm maintains Property 3.1 after each of the $O(\epsilon^{-1})$ Dual Adjustment steps in each scale. Lemmas 3.3 and 3.4 establish some facts used in the proof of Lemma 3.5.

LEMMA 3.3. *After the Augmentation and Blossom Shrinking steps, G_{elig} contains no augmenting path, nor is there an even-length alternating path from a free vertex to a blossom.*

PROOF. Suppose there is an augmenting path P in G_{elig} after augmenting along paths in Ψ. Since Ψ is maximal, P must intersect some $P' \in \Psi$ at a vertex v. However, after the Augmentation step every edge in P' will become ineligible, so the matching edge $(v, v') \in M$ is no longer in G_{elig}, contradicting the fact that P consists of eligible edges. Since Ω' is maximal there can be no blossom reachable from a free vertex in G_{elig} after the Blossom Shrinking step. □

Lemma 3.4 guarantees that all y-values updated in a Dual Adjustment step have the same parity as a multiple of $\delta_i/2$, that is, they are either both even or both odd multiples of $\delta_i/2$. In the proof of Lemma 3.5, this fact is used to argue that if both endpoints of an edge e have their y-values decremented, then $yz(e)$ is a multiple of δ_i.

LEMMA 3.4. *Let $R \subseteq V(G_{elig})$ be the set of vertices reachable from free vertices by eligible alternating paths, at any point in scale i. Let $\hat{R} \subseteq V(G)$ be the set of original vertices represented by those in R. Then, the y-values of $\hat{R}$-vertices have the same parity, as a multiple of $\delta_i/2$.*

PROOF. The claim clearly holds at initialization ($i = 0$) since all vertices have identical y-values. Assume, inductively, that before the Blossom Shrinking step, all vertices in a common blossom have the same parity, as a multiple of $\delta_i/2$. (This property is clearly preserved when transitioning from scale $i-1$ to scale i.) Consider an eligible path $P = (B_0, B_1, \ldots, B_k)$ in G_{elig}, where the $\{B_j\}$ are either vertices or blossoms in Ω and B_0 is unmatched in G_{elig}. Let $(u_0, v_1), (u_1, v_2), \ldots, (u_{k-1}, v_k)$ be the G-edges corresponding to P, where $u_j, v_j \in B_j$. By the inductive hypothesis, u_j and v_j have the same parity, and whether (u_j, v_{j+1}) is matched or unmatched, Definition 3.2 stipulates that $yz(u_j, v_{j+1})/\delta_i$ is an integer, which implies $y(u_j)$ and $y(v_{j+1})$ have the same parity as a multiple of $\delta_i/2$. Thus, the y-values of all vertices in $B_0 \cup \cdots \cup B_k$ have the same parity as a free vertex in B_0, whose y-value is equal to every other free vertex, by Property 3.1(5). Since new blossoms are formed by eligible edges, the inductive hypothesis is preserved after the Blossom Shrinking step. It is also preserved after the Dual Adjustment step since the y-values of vertices in a common blossom are incremented or decremented in lockstep. This concludes the induction. □

LEMMA 3.5. *The algorithm preserves Property* 3.1.

PROOF. Property 3.1(5) (free vertex duals) is maintained as *only* free vertices have their y-values decremented in each Dual Adjustment step. Any newly matched edge e is ineligible so neither endpoint of e can be in $\hat{V}_{out}$. Property 3.1(2) (active blossoms) is also maintained since all the new root blossoms discovered in the Blossom Shrinking step are contained in V_{out} and will have positive z-values after adjustment. Furthermore, each root blossom whose z-value drops to zero is dissolved, after Dual Adjustment. At the beginning of scale i, all y- and z-values are integer multiples of $\delta_i/2$ and δ_i, respectively, satisfying Property 3.1(1) (granularity). This property is clearly

maintained in each Dual Adjustment step. If $e \notin M$ is placed in M during an Augmentation step or placed in $\bigcup_{B\in\Omega} E_B$ during a Blossom Shrinking step, then e is type i and $yz(e) = w_i(e) - \delta_i < w_i(e)$, which satisfies Property 3.1(4).

It remains to show that the algorithm maintains Property 3.1(3,4) (near domination and near tightness). It is clear that these properties are preserved during Augmentation and Blossom Shrinking. First consider the dual adjustments made at the *end* of scale i (the last line of pseudocode in Figure 2). Let $e = (u, v)$ be an arbitrary edge and let yz and yz' be the function before and after dual adjustment. It follows that

$$\begin{aligned} yz'(e) &= yz(e) + 2\delta_{i+1} && y(u), y(v) \text{ incremented by } \delta_{i+1} \\ &\geq w_i(e) - \delta_i + 2\delta_{i+1} && \text{near domination at scale } i \\ &\geq w_{i+1}(e) - \delta_{i+1} && w_i(e) \geq w_{i+1}(e) - \delta_{i+1}. \end{aligned}$$

That is, Property 3.1(3) is preserved. If $e \in M \cup \bigcup_{B\in\Omega} E_B$ is a type j edge, then at the end of scale i Property 3.1(4) is also preserved since

$$yz'(e) = yz(e) + 2\delta_{i+1} \leq w_i(e) + 2(\delta_j - \delta_i) + 2\delta_{i+1} \leq w_{i+1}(e) + 2(\delta_j - \delta_{i+1}).$$

The first inequality follows from Property 3.1(4) at scale i and the second inequality from the fact that $w_i(e) \leq w_{i+1}(e)$ and $\delta_i = 2\delta_{i+1}$.

Now consider a Dual Adjustment step. If neither u nor v is in $\hat{V}_{in} \cup \hat{V}_{out}$ or if u, v are in the same root blossom in Ω, then $yz(e)$ is unchanged, preserving Property 3.1. The remaining cases depend on whether (u, v) is in M or not, whether (u, v) is eligible or not, and whether both $u, v \in \hat{V}_{in} \cup \hat{V}_{out}$ or not.

Case 1. $e \notin M$, $u, v \in \hat{V}_{in} \cup \hat{V}_{out}$. If e is ineligible, then $yz(e) > w_i(e) - \delta_i$. However, by Lemma 3.4 (parity of y-values), we know $(yz(e) - w_i(e))/\delta_i$ is an integer, so $yz(e) \geq w_i(e)$ before adjustment and $yz(e) \geq w_i(e) - \delta_i$ afterward (which could occur if both $u, v \in \hat{V}_{out}$), thereby preserving Property 3.1(3). If e 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, which also preserves Property 3.1(3).

Case 2. $e \in M$, $u, v \in \hat{V}_{in} \cup \hat{V}_{out}$. Since $u, v \in \hat{V}_{in} \cup \hat{V}_{out}$, Lemma 3.4 (parity of y-values) guarantees that $(yz(e) - w_i(e))/\delta_i$ is an integer. If $yz(e) - w_i(e) = -\delta_i$, then e is ineligible. Thus, both u and v are in $\hat{V}_{in}$ and $yz(e) = w_i(e)$ after dual adjustment, which preserves Property 3.1(3,4). If $yz(e) - w_i(e) \geq 0$, then e is eligible, $u \in \hat{V}_{in}$, and $v \in \hat{V}_{out}$. It cannot be that $u, v \in \hat{V}_{out}$ as otherwise e would have been included in an augmenting path or root blossom. In this case, $yz(e)$ is unchanged by dual adjustment, preserving Property 3.1(3,4).

Case 3. $e \notin M$, *only* $v \notin \hat{V}_{in} \cup \hat{V}_{out}$. If e is eligible, then $u \in \hat{V}_{in}$ and $yz(e)$ will increase. If it is ineligible, then $yz(e) \geq w_i(e) - \delta_i/2$ before adjustment and $yz(e) \geq w_i(e) - \delta_i$ afterward. In both cases, Property 3.1(3) is preserved.

Case 4. $e \in M$, *only* $v \notin \hat{V}_{in} \cup \hat{V}_{out}$. It must be that e is ineligible, so $u \in \hat{V}_{in}$ and $yz(e) - w_i(e)$ is either negative or an odd multiple of $\delta_i/2$. If e is type j, then, by Property 3.1(1,4) (granularity and near tightness), $yz(e) \leq w_i(e) + 2(\delta_j - \delta_i) - \delta_i/2$ before adjustment and $yz(e) \leq w_i(e) + 2(\delta_j - \delta_i)$ afterward, preserving Property 3.1(4). □

Recall that Lemma 2.3 stated that the final matching will be a $(1 - O(\epsilon))$-MWM if free vertices have zero y-values, and $|yz(e) - w(e)| = O(\epsilon) \cdot w(e)$. Lemmas 3.6 and 3.7 establish these bounds.

LEMMA 3.6. *Let $i \le L$ be the scale index. Then*

(1) *for $i < L$, all edges eligible at any time in scales 0 through i have weight at least $N/2^{i+1} + \delta_i$;*
(2) *for any i, if $e \in M$, then $yz(e) \le (1 + 4\epsilon')w(e)$.*

PROOF.

Part 1. The last search for augmenting paths in scale i begins when the y-values of free vertices are $N/2^{i+2}$, and strictly less than y-values of other vertices, by Property 3.1(5). An unmatched edge $e = (u, v)$ can only be eligible at this scale if $yz(e) = w_i(e) - \delta_i \le w(e) - \delta_i$. Hence, $w(e) \ge w_i(e) \ge y(u) + y(v) + \delta_i \ge N/2^{i+1} + \delta_i$.

Part 2. Let e be a type j edge in M during scale i. Property 3.1(4) states that $yz(e) - w_i(e) \le 2(\delta_j - \delta_i)$. Since $w_i(e) \le w(e)$ it also follows that $yz(e) - w(e) \le 2\delta_j - 2\delta_i < 2\delta_j = \epsilon' N/2^{j-1}$. By part 1, a type j edge must have weight at least $N/2^{j+1} + \delta_j$, hence $yz(e) - w(e) < 4\epsilon' \cdot w(e)$. □

LEMMA 3.7. *After scale $L = \log N$, M is a $(1 - 5\epsilon')$-*MWM.

PROOF. The final scale ends when free vertices have zero y-values and $\delta_L = \epsilon'$. According to Lemmas 3.6 and 2.3, $w(M) \ge (1-\epsilon')(1+4\epsilon')^{-1} \cdot w(M^*) > (1-5\epsilon') \cdot w(M^*)$. □

THEOREM 3.8. *A $(1 - \epsilon)$-*MWM *can be computed in time $O(m\epsilon^{-1} \log N)$.*

PROOF. Each Augmentation and Blossom Shrinking step takes $O(m)$ time using a modified depth-first search [Gabow and Tarjan 1991, Sect. 8]. (Finding a maximal set of augmenting paths is significantly simpler, conceptually, than finding a maximal set of minimum-length augmenting paths, as is done in Micali and Vazirani [1980] and Vazirani [1994].) Each Dual Adjustment step clearly takes linear time, as does the dual adjustment at the end of each scale. Scale $i < L = \log N$ begins with free vertices' y-values at $N/2^{i+1}$ (or $N/2 - \delta_0/2$, if $i = 0$) and performs Dual Adjustments until they are $N/2^{i+2} - \delta_i/2$. Since y-values of free vertices are decremented by $\delta_i/2$ in each Dual Adjustment step, there are exactly $(N/2^{i+2} + \delta_i/2)/(\delta_i/2) = N/(2\delta_0) + 1 = \epsilon'^{-1}/2 + 1$ such steps. The final scale begins with free vertices' y-values at $N/2^{L+1}$ and ends with them at zero, so there are fewer than $(N/2^{L+1})/(\delta_L/2) = \epsilon'^{-1}$ Dual Adjustment steps. Lemma 3.7 guarantees that the final matching is a $(1 - \epsilon)$-MWM for $\epsilon' \le \epsilon/5$. Note that M is not represented explicitly. After scale L, we have computed a nested set of blossoms Ω and a matching, call it M', in the contracted graph G/Ω. Extending M' to the corresponding matching M in G is easily accomplished in $O(n)$ time by dissolving the blossoms in a top-down fashion. Hence, the total running time is $O(m\epsilon^{-1} \log N)$. □

3.3. A Linear-Time Algorithm

Our $O(m\epsilon^{-1} \log N)$-time algorithm requires few modifications to run in linear time, independent of N. In fact, the algorithm as it appears in Figure 2 requires no modifications at all: we only need to change the definition of *eligibility* and, in each scale, refrain from scanning edges that cannot possibly be eligible. In light of Lemma 3.6(1), it is helpful to index edges according to the first scale in which they may be eligible.

Definition 3.9. Define $\mu_i = N/2^{i+1} + \delta_i$, for $i < L$, and $\mu_L = 0$. Define scale(e) to be the i such that $w(e) \in [\mu_i, \mu_{i-1})$.

In other words, once we compute scale(e) we can ignore e in all scales $i <$ scale(e), thereby saving time. The idea of our linear time algorithm is to forcibly ignore e in

scales $i > \text{scale}(e) + \log \epsilon'^{-1}$. Ignoring an otherwise eligible edge can cause violations of near tightness and near dominance that increase with each dual adjustment. We shall prove that beyond scale $\text{scale}(e) + \log \epsilon'^{-1}$, the magnitude of these violations is an $O(\epsilon')$ fraction of the weight of e, which is small enough to obtain a $(1 - O(\epsilon'))$-MWM. Definition 3.10 redefines eligibility. The differences with Definition 3.2 are underlined.

Definition 3.10. Let $\gamma = \log \epsilon'^{-1}$. At scale i, an edge e is *eligible* if at least one of the following holds.

(1) $e \in E_B$ for some $B \in \Omega$.
(2) $e \notin M$ and $yz(e) = w_i(e) - \delta_i$ and $\text{scale}(e) \geq i - \gamma$.
(3) $e \in M$ and $yz(e) - w_i(e)$ is a nonnegative integer multiple of δ_i and $\text{scale}(e) \geq i - \gamma$.

Let E_{elig} be the set of eligible edges and let $G_{elig} = (V, E_{elig})/\Omega$ be the unweighted graph obtained by discarding ineligible edges and contracting root blossoms.

LEMMA 3.11. *Consider an execution of the algorithm using Definition* 3.10 *of eligibility rather than Definition* 3.2. *Property* 3.1(1,2,5) *holds throughout the execution. For an edge e and $k = \text{scale}(e)$, Property* 3.1(3,4) *(near domination and near tightness) hold in all scales $i \leq k + \gamma$ and thereafter in the following form. For each e, $yz(e) > (1 - \epsilon')w_i(e)$ and for each matching or blossom edge $yz(e) < (1 + 6\epsilon')w_i(e)$.*

PROOF. The definition of eligibility has no bearing on Property 3.1(1,2,5) so they are maintained correctly. In scales k through $k + \gamma$, Property 3.1(3,4) is maintained as the two definitions of eligibility are the same. At the beginning of scale $t = k + \gamma + 1$, e may still be eligible if it is in a blossom. The moment the blossom (if any) is dissolved, e will become ineligible and remain so for the remainder of the computation. At the beginning of scale t, the y-values of free vertices are $N/2^{t+1}$. From this moment on, the y-values of free vertices are incremented by a total of $\sum_{l \geq t+1} \delta_l$ (the dual adjustments following scales t through $L - 1$) and decremented a total of $N/2^{t+1} + \sum_{l \geq t+1} \delta_l$ (in the Dual Adjustment steps following searches for augmenting paths and blossoms). We consider the effects of these adjustments on near tightness and near domination separately.

Near Tightness. Let $e = (u, v)$ be a type j edge (matched or in a blossom) at the beginning of scale $t \leq L$. Each adjustment to y-values by some quantity Δ may cause $yz(e)$ to increase by 2Δ. This clearly occurs in the dual adjustments following each scale as $y(u)$ and $y(v)$ are incremented by Δ. Following a search for blossoms, it may be that $u, v \in \hat{V}_{in}$, which would also cause $y(u)$ and $y(v)$ to each be incremented by Δ. Putting this all together, it follows that at any scale $i \geq t = k + \gamma + 1 = k + \log \epsilon'^{-1} + 1$,

$$
\begin{aligned}
yz(e) &\leq w_i(e) + 2(\delta_j - \delta_t) + 2 \cdot \left(N/2^{t+1} + 2 \cdot \sum_{l \geq t+1} \delta_l \right) \\
&< w_i(e) + 2(\delta_k - \delta_t) + 2\left(\delta_{k+2} + 2\delta_t\right) && j \geq k, \text{ defn. of } t \\
&= w_i(e) + (2 + 1/2 + \epsilon')\delta_k && \text{defn. of } \delta_t = \epsilon' \delta_k / 2 \\
&< w_i(e) + 3\delta_k \\
&< (1 + 6\epsilon') \cdot w_i(e) && w(e) \geq w_i(e) > N/2^{k+1} = (2\epsilon')^{-1} \cdot \delta_k \,.
\end{aligned}
$$

The inequality $yz(e) < (1 + 6\epsilon') \cdot w_i(e)$ also holds if $i < t$ since, in this case, $yz(e) \leq w_i(e) + 2(\delta_j - \delta_i) < w_i(e) + 2\delta_k$.

Near Domination. At the beginning of scale $t \le L$, the y-values of free vertices are $N/2^{t+1}$ and we have $yz(e) \ge w_t(e) - \delta_t$. In the remaining dual adjustments, $yz(e)$ will be incremented by $2(\sum_{l \ge t+1} \delta_l)$ (following each scale) and decremented by as much as $2(N/2^{t+1} + \sum_{l \ge t+1} \delta_l)$. (This could occur if in each dual adjustment step, $u, v \in \hat{V}_{out}$.) Thus, at any scale $i \ge t = k + \gamma + 1$,

$$\begin{aligned} yz(e) &\ge w_t(e) - \delta_t - N/2^t \\ &> w_i(e) - 2\delta_t - N/2^t \\ &= w_i(e) - (\epsilon' + \tfrac{1}{2})\delta_k && t = k + \gamma + 1;\ \delta_k = \epsilon' N/2^k \\ &\ge w_i(e) \cdot (1 - (\epsilon' + \tfrac{1}{2})/(\tfrac{1}{2\epsilon'} + 1)) && w_i(e) \ge N/2^{k+1} + \delta_k = (\tfrac{1}{2\epsilon'} + 1)\delta_k \\ &= w_i(e) \cdot (1 - \epsilon') && . \end{aligned}$$

If $t > L$, that is, Property 3.1(3) holds at the end of the computation, then $yz(e) \ge w_L(e) - \delta_L = w(e) - \epsilon' \ge (1 - \epsilon') \cdot w(e)$. □

THEOREM 3.12. *A* $(1 - \epsilon)$-MWM *can be computed in time* $O(m\epsilon^{-1} \log \epsilon^{-1})$.

PROOF. We execute the algorithm from Figure 2 where G_{elig} refers to the eligible subgraph as defined in Definition 3.10. According to Lemmas 2.3 and 3.11, the algorithm returns a $((1 - \epsilon')(1 + 6\epsilon')^{-1})$-MWM, which, for $\epsilon' \le \epsilon/7$, is a $(1 - \epsilon)$-MWM. It remains to prove that the running time is $O(m\epsilon^{-1} \log \epsilon^{-1})$.

Consider an edge e with scale$(e) = k$. By Lemma 3.6(1) e can be ignored in scales 0 through $k - 1$. It can only appear in G_{elig} (that is, as an eligible edge not in any blossom) in scales k through $k + \gamma$. In scales $k + \gamma + 1$ through $L = \log N$, e may remain eligible, but only as a blossom edge. As soon as the blossoms containing e are dissolved, e can be ignored as it will never become eligible again. Since the algorithm need not spend any time examining edges in contracted blossoms, each edge actively participates in only $\gamma + 1 = \log \epsilon^{-1} + 1$ scales, with $O(\epsilon^{-1})$ iterations per scale. The total running time is therefore $O(m\epsilon^{-1} \log \epsilon^{-1})$, provided that the scale$(\cdot)$ function can be efficiently computed.

Computing scale(e) is tantamount to computing the most significant bit of $w(e)$. Once MSB$(w(e)) = \lfloor \log_2 w(e) \rfloor$ is known, scale(e) can be just one of two values. Recall from Section 2 that we can assume, without loss of generality, that $N \le n^2$. Using $O(n)$ space and preprocessing time we can tabulate the MSB function on $\log n$-bit integers. We can then determine MSB$(w(e))$ in $O(1)$ time with two table lookups. □

REFERENCES

Alt, H., Blum, N., Mehlhorn, K., and Paul, M. 1991. Computing a maximum cardinality matching in a bipartite graph in time $O(n^{1.5}\sqrt{m/\log n})$. *Inf. Proc. Lett. 37*, 4, 237–240.

Anderson, T., Owicki, S., Saxe, J., and Thacker, C. 1993. High speed switch scheduling for local area networks. *ACM Trans. Comput. Syst. 11*, 4, 319–352.

Andersson, A., Hagerup, T., Nilsson, S., and Raman, R. 1998. Sorting in linear time? *J. Comput. Syst. Sci. 57*, 1, 74–93.

Avis, D. 1978. Two greedy heuristics for the weighted matching problems. In *Proceedings of the 9th Southeast Conference on Combinatorics, Graph Theory, and Computing (CongrNumer XXI)*. 65–76.

Avis, D. 1983. A survey of heuristics for the weighted matching problem. *Networks 13*, 475–493.

Balinski, M. L. 1969. Labelling to obtain a maximum matching. In *Combinatorial Mathematics and Its Applications*, R. C. Bose and T. A. Downing Eds., University of North Carolina Press, 585–602.

Bertsekas, D. P. 1981. A new algorithm for the assignment problem. *Math. Prog. 21*, 152–171.

Birkhoff, G. 1946. Tres observaciones sobre el elgebra lineal. *Universidad Nacional de Tucuman, Revista A 5*, 1–2, 147–151.

Brogden, H. E. 1946. An approach to the problem of differential prediction. *Psychometrika 11*, 3, 139–154.

Brown, G. and von Neumann, J. 1950. Solutions of games by differential equations. In *Contributions to the Theory of Games*, H. Kuhn and A. Tucker Eds., Annals of Mathematical Studies Series, vol. 24. Princeton University Press, 73–79.

Cheriyan, J. and Mehlhorn, K. 1996. Algorithms for dense graphs and networks on the random access computer. *Algorithmica 15*, 6, 521–549.

Christofides, N. 1976. Worst case analysis of a new heuristic for the travelling salesman problem. Tech. rep., Graduate School of Industrial Administration, Carnegie Mellon University.

Cole, R., Hariharan, R., Lewenstein, M., and Porat, E. 2001. A faster implementation of the Goemans-Williamson clustering algorithm. In *Proceedings of the 12th ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 17–25.

Cunningham, W. H. and Marsh, III, A. B. 1978. A primal algorithm for optimum matching. *Math. Prog. Study 8*, 50–72.

Cygan, M., Gabow, H. N., and Sankowski, P. 2012. Algorithmic applications of Baur-Strassen's theorem: Shortest cycles, diameter and matchings. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 531–540.

Dantzig, G. B. 1951. Application of the simplex method to the transportation problem. In *Activity Analysis of Production and Allocation*, Cowles Commission Monograph 13, T. C. Koopmans Ed., Wiley, New York, 359–373.

Desler, J. F. and Hakimi, S. L. 1969. A graph-theoretic approach to a class of integer-programming problems. *Oper. Res. 17*, 6, 1017–1033.

Dijkstra, E. W. 1959. A note on two problems in connexion with graphs. *Numer. Math. 1*, 269–271.

Dinic, E. A. 1970. Algorithm for solution of a problem of maximum flow in networks with power estimation. *Soviet Math. Dokl. 11*, 1277–1280.

Dinic, E. A. and Kronrod, M. A. 1969. An algorithm for the solution of the assignment problem. *Soviet Math. Dokl. 10*, 6, 1324–1326.

Drake, D. and Hougardy, S. 2003a. Improved linear time approximation algorithms for weighted matchings. In *Proceedings of the 7th International Workshop on Randomization and Approximation Techniques in Computer Science (APPROX)*. Lecture Notes in Computer Science, vol. 2764, Springer. 14–23.

Drake, D. and Hougardy, S. 2003b. A simple approximation algorithm for the weighted matching problem. *Inf. Proc. Lett. 85*, 211–213.

Duan, R. and Pettie, S. 2010. Approximating maximum weight matching in near-linear time. In *Proceedings of the 51st IEEE Symposium on Foundations of Computer Science (FOCS)*. 673–682.

Duan, R. and Su, H.-H. 2012. A scaling algorithm for maximum weight matching in bipartite graphs. In *Proceedings of the 23rd ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 1413–1424.

Duff, I. S. and Gilbert, J. R. 2002. Maximum-weighted matching and block pivoting for symmetric indefinite systems. In *Householder Symposium XV Book of Abstracts*, 73–75.

Easterfield, T. E. 1946. A combinatorial algorithm. *J. London Math. Soc. 21*, 219–226. (Republished as: An algorithm for the allocation problem, *Oper. Res. 11*, 3, 123–129, 1960.)

Edmonds, J. 1965a. Maximum matching and a polyhedron with 0, 1-vertices. *J. Res. Nat. Bur. Stand. Sect. B 69B*, 125–130.

Edmonds, J. 1965b. Paths, trees, and flowers. *Canad. J. Math. 17*, 449–467.

Edmonds, J. and Johnson, E. L. 1973. Matching, Euler tours, and the Chinese postman. *Math. Prog. 5*, 88–124.

Edmonds, J. and Karp, R. M. 1972. Theoretical improvements in algorithmic efficiency for network flow problems. *J. ACM 19*, 2, 248–264.

Feder, T. and Motwani, R. 1995. Clique partitions, graph compression and speeding-up algorithms. *J. Comput. Syst. Sci. 51*, 2, 261–272.

Ford, L. R. and Fulkerson, D. R. 1962. *Flows in Networks*. Princeton University Press.

Fredman, M. L. and Tarjan, R. E. 1987. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM 34*, 3, 596–615.

Gabow, H. N. 1974. Implementation of algorithms for maximum matching on nonbipartite graphs. Ph.D. thesis, Stanford University.

Gabow, H. N. 1985a. A scaling algorithm for weighted matching on general graphs. In *Proceedings of the 26th IEEE Symposium on Foundations of Computer Science (FOCS)*. 90–100.

Gabow, H. N. 1985b. Scaling algorithms for network problems. *J. Comput. Syst. Sci. 31*, 2, 148–168.

Gabow, H. N. 1990. Data structures for weighted matching and nearest common ancestors with linking. In *Proceedings of the 1st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 434–443.

Gabow, H. N. and Pettie, S. 2002. The dynamic vertex minimum problem and its application to clustering-type approximation algorithms. In *Proceedings of the 8th Scandinavian Workshop on Algorithm Theory (SWAT)*. 190–199.

Gabow, H. N. and Tarjan, R. E. 1985. A linear-time algorithm for a special case of disjoint set union. *J. Comput. Syst. Sci. 30*, 2, 209–221.

Gabow, H. N. and Tarjan, R. E. 1989. Faster scaling algorithms for network problems. *SIAM J. Comput. 18*, 5, 1013–1036.

Gabow, H. N. and Tarjan, R. E. 1991. Faster scaling algorithms for general graph-matching problems. *J. ACM 38*, 4, 815–853.

Gabow, H. N., Galil, Z., and Spencer, T. H. 1989. Efficient implementation of graph algorithms using contraction. *J. ACM 36*, 3, 540–572.

Galil, Z., Micali, S., and Gabow, H. N. 1986. An $O(EV \log V)$ algorithm for finding a maximal weighted matching in general graphs. *SIAM J. Comput. 15*, 1, 120–130.

Giaccone, P., Leonardi, E., and Shah, D. 2005. On the maximal throughput of networks with finite buffers and its application to buffered crossbars. In *Proceedings of the 24th INFOCOM*. Vol. 2, 971–980.

Gleyzal, A. 1955. An algorithm for solving the transportation problem. *J. Res. Nat. Bur. Standards 54*, 4, 213–216.

Goemans, M. X. and Williamson, D. P. 1995. A general approximation technique for constrained forest problems. *SIAM J. Comput. 24*, 2, 296–317.

Goldberg, A. V. and Karzanov, A. V. 2004. Maximum skew-symmetric flows and matchings. *Math. Program., Ser. A 100*, 537–568.

Goldberg, A. V. and Kennedy, R. 1997. Global price updates help. *SIAM J. Disc. Math. 10*, 4, 551–572.

Hadlock, F. 1975. Finding a maximum cut of a planar graph in polynomial time. *SIAM J. Comput. 4*, 3, 221–225.

Hagemann, M. and Schenk, O. 2006. Weighted matchings for preconditioning symmetric indefinite linear systems. *SIAM J. Sci. Comput. 28*, 2, 403–420.

Han, Y. 2002. Deterministic sorting in $O(n \log \log n)$ time and linear space. In *Proceedings of the 34th ACM Symposium on Theory of Computing (STOC)*. 602–608.

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

Harvey, N. 2009. Algebraic algorithms for matching and matroid problems. *SIAM J. Comput. 39*, 2, 679–702.

Hendrickson, B. and Leland, R. 1995. The Chaco user's guide: Version 2.0. Tech. rep. SAND94–2344, Sandia National Laboratories.

Hitchcock, F. L. 1941. The distribution of a product from several sources to numerous localities. *J. Math. Phys. 20*, 224–230.

Hoffman, A. J. and Markowitz, H. M. 1963. A note on shortest path, assignment, and transportation problems. *Naval Res. Logistics Quart. 10*, 1, 375–379.

Holtgrewe, M., Sanders, P., and Schulz, C. 2010. Engineering a scalable high quality graph partitioner. In *Proceedings of the 24th IEEE Symposium on Parallel and Distributed Processing (IPDPS)*. 1–12.

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

Huang, C.-C. and Kavitha, T. 2012. Efficient algorithms for maximum weight matchings in general graphs with small edge weights. In *Proceedings of the 23rd ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 1400–1412.

Ibarra, O. H. and Moran, S. 1981. Deterministic and probabilistic algorithms for maximum bipartite matching via fast matrix multiplication. *Inf. Proc. Lett. 13*, 1, 12–15.

Iri, M., Murota, K., and Matsui, S. 1983. Heuristics for planar minimum-weight perfect matchings. *Networks 13*, 1, 67–92.

Johnson, D. B. 1975. Priority queues with update and finding minimum spanning trees. *Inf. Proc. Lett. 4*, 3, 53–57.

Kantorovitch, L. 1942. On the translocation of masses. *Doklady Akad. Nauk SSSR 37*, 199–201.

Kao, M.-Y., Lam, T.-K., Sung, W.-K., and Ting, H.-F. 2001. A decomposition theorem for maximum weight bipartite matchings. *SIAM J. Comput. 31*, 1, 18–26.

Karypis, G. and Kumar, V. 1998. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput. 20*, 1, 359–392.

Karzanov, A. V. 1973. An exact estimate of an algorithm for finding a maximum flow, applied to the problem "on representatives" [in Russian]. *Probl. Cybernetics 5*, 66–70. (English translation available at the author's website.)

Karzanov, A. V. 1976. Efficient implementations of Edmonds' algorithms for finding matchings with maximum cardinality and maximum weight. In *Studies in Discrete Optimization*, A. A. Fridman Ed., Nauka, Moscow, 306–327.

Kuhn, H. W. 1955a. The Hungarian method for the assignment problem. *Naval Res. Log. Quart. 2*, 83–97.

Kuhn, H. W. 1955b. On combinatorial properties of matrices. *George Washington University Logistics Papers 11*, 1–11. (English translation of J. Egerváry, Matrixok kombinatorius tulajdonságairól, *Matematikai és Fizikai Lapok 38*, 16–28, 1931.)

Kuhn, H. W. 1956. Variants of the Hungarian method for assignment problems. *Naval Res. Log. Quart. 3*, 253–258.

Kuhn, H. W. and Baumol, W. J. 1962. An approximate algorithm for the fixed-charges transportation problem. *Naval Res. Log. Quart. 9*, 1–15.

Kurtzberg, J. M. 1962. On approximation methods for the assignment problem. *J. ACM 9*, 4, 419–439.

Kwan, M. K. 1962. Graphic programming using odd or even points. *Chinese Math. 1*, 273–277.

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

Leonardi, E., Mellia, M., Neri, F., and Marsan, M. A. 2003. Bounds on delays and queue lengths in input-queued cell switches. *J. ACM 50*, 4, 520–550.

McKeown, N. 1999. The iSLIP scheduling algorithm for input-queued switches. *IEEE/ACM Trans. Netw. 7*, 2, 188–201.

McKeown, N., Anantharam, V., and Walrand, J. C. 1996. Achieving 100% throughput in an input-queued switch. In *Proceedings of the 15th INFOCOM*. 296–302.

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

Motzkin, T. S. 1956. The assignment problem. In *Proceedings of the Symposia in Applied Mathematics VI,* Numerical Analysis, 109–125.

Mucha, M. and Sankowski, P. 2004. Maximum matchings via Gaussian elimination. In *Proceedings of the 45th IEEE Symposium on Foundations of Computer Science (FOCS)*. 248–255.

Munkres, J. 1957. Algorithms for the assignment and transportation problems. *J. Soc. Indust. Appl. Math. 5*, 32–38.

Ollivier, F. 2009. Looking for the order of a system of arbitrary ordinary differential equations. *Appl. Algebra Eng. Commun. Comput. 20*, 1, 7–32. (English translation of: C. G. J. Jacobi, De investigando ordine systematis aequationum differentialum vulgarium cujuscunque, *Borchardt Journal für die reine und angewandte Mathematik 65*, 4, pp. 297–320, 1865.)

Olschowka, M. and Neumaier, A. 1996. A new pivoting strategy for Gaussian elimination. *Linear Algebra Appl. 240*, 131–151.

Orlin, J. B. and Ahuja, R. K. 1992. New scaling algorithms for the assignment and minimum mean cycle problems. *Math. Program. 54*, 41–56.

Orlova, G. I. and Dorfman, Y. G. 1972. Finding the maximum cut in a planar graph. *Eng. Cybernet. 10*, 502–506.

Pellegrini, F. 2008. SCOTCH 5.1 user's guide. Technical report, LaBRI.

Pettie, S. 2005. Sensitivity analysis of minimum spanning trees in sub-inverse-Ackermann time. In *Proceedings of the 16th International Symposium on Algorithms and Computation (ISAAC)*. 964–973.

Pettie, S. 2012. A simple reduction from maximum weight matching to maximum cardinality matching. *Inf. Proc. Lett. 112*, 23, 893–898.

Pettie, S. and Sanders, P. 2004. A simpler linear time $2/3 - \epsilon$ approximation to maximum weight matching. *Inf. Proc. Lett. 91*, 6, 271–276.

Preis, R. 1999. Linear time 1/2-approximation algorithm for maximum weighted matching in general graphs. In *Proceedings of the 16th Symposium on Theoretical Aspects of Computer Science (STACS)*. Lecture Notes in Computer Science, vol. 1563, 259–269.

Preis, R. and Diekmann, R. 1997. 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, 63–71.

Ramshaw, L. and Tarjan, R. E. 2012. A weight-scaling algorithm for min-cost imperfect matchings in bipartite graphs. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science, (FOCS)*. 581–590.

Reingold, E. M. and Supowit, K. J. 1983. Probabilistic analysis of divide-and-conquer heuristics for minimum weighted Euclidean matching. *Networks 13*, 49–66.

Reingold, E. M. and Tarjan, R. E. 1981. On a greedy heuristic for complete matching. *SIAM J. Comput. 10*, 4, 676–681.

Sanders, P. and Schulz, C. 2012. Distributed evolutionary graph partitioning. In *Proceedings of the 14th Workshop on Algorithm Engineering and Experiments (ALENEX)*. 43–54.

Sankowski, P. 2009. Maximum weight bipartite matching in matrix multiplication time. *Theoret. Comput. Sci. 410*, 44, 4480–4488.

Schenk, O., Wächter, A., and Hagemann, M. 2007. Matching-based preprocessing algorithms to the solution of saddle-point problems in large-scale nonconvex interior-point optimization. *Comput. Optim. Appl. 36*, 2–3, 321–341.

Shah, D., Giaccone, P., and Prabhakar, B. 2002. Efficient randomized algorithms for input-queued switch scheduling. *IEEE Micro 22*, 1, 10–18.

Shah, D. and Kopikare, M. 2002. Delay bounds for the approximate maximum weight matching algorithm for input queued switches. In *Proceedings of the 21st INFOCOM*. 1024–1031.

Sharathkumar, R. and Agarwal, P. K. 2012. A near-linear time ϵ-approximation algorithm for geometric bipartite matching. In *Proceedings of the 44th Symposium on Theory of Computing Conference (STOC)*. 385–394.

Thorndike, R. L. 1950. The problem of classification of personnel. *Psychometrika 15*, 215–235.

Thorup, M. 1999. Undirected single-source shortest paths with positive integer weights in linear time. *J. ACM 46*, 3, 362–394.

Thorup, M. 2003. Integer priority queues with decrease key in constant time and the single source shortest paths problem. In *Proceedings of the 35th ACM Symposium on Theory of Computing (STOC)*. 149–158.

Thorup, M. 2007. Equivalence between priority queues and sorting. *J. ACM 54*, 6.

Tomizawa, N. 1971. On some techniques useful for solution of transportation network problems. *Networks 1*, 2, 173–194.

Varadarajan, K. R. and Agarwal, P. K. 1999. Approximation algorithms for bipartite and non-bipartite matching in the plane. In *Proceedings of the 10th ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 805–814.

Vazirani, V. V. 1994. A theory of alternating paths and blossoms for proving correctness of the $O(\sqrt{V}E)$ general graph maximum matching algorithm. *Combinatorica 14*, 1, 71–109.

Vinkemeier, D. E. D. and Hougardy, S. 2005. A linear-time approximation algorithm for weighted matchings in graphs. *ACM Trans. Algor. 1*, 1, 107–122.

von Neumann, J. 1953. A certain zero-sum two-person game equivalent to the optimal assignment problem. In *Contributions to the Theory of Games*, H. W. Kuhn and A. W. Tucker Eds., Vol. II, Princeton University Press, 5–12.

Walshaw, C. and Cross, M. 2007. JOSTLE: Parallel multilevel graph-partitioning software – an overview. In *Mesh Partitioning Techniques and Domain Decomposition Techniques*, 27–58.

Williams, V. V. 2012. Multiplying matrices faster than Coppersmith-Winograd. In *Proceedings of the 44th Symposium on Theory of Computing Conference (STOC)*. 887–898.

Witzgall, C. and Zahn, Jr., C. T. 1965. Modification of Edmonds' maximum matching algorithm. *J. Res. Nat. Bur. Standards Sect. B 69B*, 91–98.

Received December 2011; revised March 2013, September 2013; accepted September 2013