Mathematics of Operations Research

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

New Algorithms for Maximum Weight Matching and a Decomposition Theorem

Chien-Chung Huang, Telikepalli Kavitha

To cite this article:
Chien-Chung Huang, Telikepalli Kavitha (2016) New Algorithms for Maximum Weight Matching and a Decomposition Theorem. Mathematics of Operations Research

Published online in Articles in Advance 21 Oct 2016

. http://dx.doi.org/10.1287/moor.2016.0806

MATHEMATICS OF OPERATIONS RESEARCH
Articles in Advance, pp. 1–16
ISSN 0364-765X (print) | ISSN 1526-5471 (online)

https://doi.org/10.1287/moor.2016.0806

New Algorithms for Maximum Weight Matching and a Decomposition Theorem

Chien-Chung Huang
Chalmers University of Technology, villars@gmail.com

Telikepalli Kavitha
Tata Institute of Fundamental Research, India, kavitha@tcs.tifr.res.in

We revisit the classical maximum weight matching problem in general graphs with nonnegative integral edge weights. We present an algorithm that operates by decomposing the problem into W unweighted versions of the problem, where W is the largest edge weight. Our algorithm has running time as good as the current fastest algorithms for the maximum weight matching problem when W is small. One of the highlights of our algorithm is that it also produces an integral optimal dual solution; thus our algorithm also returns an integral certificate corresponding to the maximum weight matching that was computed.

Our algorithm yields a new proof to the total dual integrality of Edmonds' matching polytope and it also gives rise to a decomposition theorem for the maximum weight of a matching in terms of the maximum size of a matching in certain subgraphs. We also consider the maximum weight capacitated b-matching problem in bipartite graphs with nonnegative integral edge weights and show that it can also be decomposed into W unweighted versions of the problem, where W is the largest edge weight. Our second algorithm is competitive with known algorithms when W is small.

Keywords: maximum weight matching; exact algorithms; total dual integrality
MSC2000 subject classification: Primary: 05C70
OR/MS subject classification: Primary: Analysis of algorithms
History: Received November 13, 2014; revised September 25, 2015, and May 10, 2016. Published online in *Articles in Advance* October 21, 2016.

1. Introduction. The input here is a graph $G = (V, E)$ with edge weights given by the function w: $E \to \{1, 2, \ldots, W\}$. A *matching* M is a subset of E such that no two edges in M share an endpoint. The weight of a matching M is the sum of the weights of the edges in M. Our objective is to find a *maximum weight matching* in G. We note that a maximum weight matching need not be a maximum cardinality matching; a maximum weight matching has many applications. For instance, suppose V is a set of players, E is the set of possible pairings of players, and the weight of each edge is the utility of pairing its endpoints together. We seek a set of disjoint pairings so that the sum of utilities is maximized; in other words, what we seek is a maximum weight matching in G.

A closely related problem is that of computing a maximum weight perfect matching, where the goal is to find a perfect matching, and subject to that, one with the largest weight. Although the maximum weight matching problem and the maximum weight perfect matching problem can be reduced to each other in polynomial time, the two problems are different and an algorithm designed for one problem may not achieve the same running time in the other problem. See Duan and Pettie [9] for a more detailed discussion on this issue. The algorithms presented in this paper have running time guarantees only for the maximum weight matching problem.

Matching problems lie at the core of graph theory, polyhedral combinatorics, and linear optimization. Because of their fundamental nature and vast application, in the past decades, intense investigations have been made for the problem. Edmonds' pioneering work back in the 1960s especially highlights the intricate correlation between algorithm design and polyhedral characterization. We refer the interested reader to Duan and Pettie [9] and Schrijver [41] for a history of the various matching algorithms and their performance. For the polyhedral characterization aspect of matchings, see Cook et al. [4] and Schrijver [41].

The most common approach to solving the maximum weight matching problem is the primal-dual schema—often called the *Hungarian method* (Kuhn [28]) in the special case of bipartite graphs. For general graphs, this approach was initiated by Edmonds [12], and various later algorithms, such as Gabow [17], can be regarded as refinements of Edmonds' algorithm. The idea is to build up feasible primal and dual solutions simultaneously and show that in the end, both solutions satisfy complementary slackness conditions and hence by the duality theorem, the primal solution is a maximum weight matching. Another approach in dealing with the maximum weight matching problem is to maintain a feasible matching and successively augment it to increase its weight, until no more augmentation is possible. The work of Cunningham and Marsh [6] (and also Derigs [8]) can be regarded as representative of this approach.

A different approach, though still primal-dual in nature, was proposed by Kao et al. [27]. In the special case of bipartite graphs, they showed that the problem can be decomposed into W maximum cardinality matchings. Using the fastest maximum cardinality matching algorithms as a subroutine, their algorithm was faster than other

TABLE 1. A summary of the current fastest maximum weight matching algorithms in $G = (V, E)$, where $|V| = n$ and $|E| = m$. The $\tilde{O}$ notation of the Cyganet et al. [7] algorithm hides some factors of $\log nW$.

Running time	Algorithm
$O(n(m + n\log n))$	Gabow [17]
$O(m\sqrt{n\log n}\log nW)$, $O(m\sqrt{n}\log nW)$	Gabow and Tarjan [20], Duan et al. [10]
$\tilde{O}(Wn^{\omega})$	Cygan et al. [7]
$O(Wn^{\omega})$, $O(W\sqrt{nm}\log_n n^2/m)$	Gabow [15], Pettie [38], this paper

algorithms when W is small. Their algorithm can be roughly described as follows. In the i-th iteration, only edges of weights higher than $W - i$ are considered and in particular, only edges whose weight minus the vertex potentials (i.e., the dual solution) equals one are retained. Next, a maximum cardinality matching is computed in the subgraph and this matching is subsequently used to update the vertex potentials. The final matching and the vertex potentials can be shown to satisfy the complementary slackness conditions.

In a preliminary version of the paper (Huang and Kavitha [26]), we showed that the same approach can be generalized to the case of general graphs.[1] The difficulty mostly lies in how to manipulate the "blossoms" of Edmonds' algorithm and update the more complicated dual variables. Throughout the paper, let $n = |V|$ and $m = |E|$. We show that by using one of the fastest maximum cardinality matchings as a subroutine (Goldberg and Karzanov [22], Mucha and Sankowski [33]), we can solve the maximum weight matching problem in $O(W\sqrt{nm}\log_n(n^2/m))$ time or in $O(Wn^{\omega})$ time with high probability, where $\omega \approx 2.3728$ is the exponent of matrix multiplication (Le Gall [30]).

Table 1 has the running times of the various fastest maximum weight matching algorithms in general graphs. Subsequent to the preliminary version of this paper, Pettie [38] pointed out that the earlier algorithm of Gabow [15] can be shown to achieve the same running time with a much simpler proof; in fact, he showed that Gabow's algorithm also can be regarded as decomposing the problem into W maximum cardinality matching problems. Compared to the previous algorithms (Duan et al. [10], Gabow [17], Gabow and Tarjan [20]), Gabow's algorithm (Gabow [15]) and our algorithm (without using the algebraic algorithm of Mucha and Sankowski [33] as a subroutine) are faster when $W = o(\log n)$, and if the graph is very dense, i.e., $m = \Theta(n^2)$, then the two latter algorithms are faster when $W = o(\log^2 n)$. Compared to the algebraic algorithm (Cygan et al. [7]), Gabow's algorithm and ours are always faster by a poly-log factor.

In the present work, we give a new algorithm for solving the maximum weight matching problem in general graphs. This algorithm is significantly different from our previous one and its proof and presentation are much simpler. Additionally, our algorithm produces an integral optimal dual solution. This in turn gives a new proof to the fact that the linear program proposed by Edmonds to describe the matching polytope is totally dual integral. It is well known that if a matrix A is *totally unimodular* and b, c are integral vectors, then $\max\{c^T x\colon Ax \le b, x \ge 0\}$ and $\min\{y^T b\colon y^T A \ge c^T, y \ge 0\}$ are attained by integral vectors x and y whenever the optima exist and are finite (Schrijver [40]). However, total unimodularity is a very strong condition and, in fact, the integrality of vectors x and y holds even under the weaker condition that the system of inequalities $Ax \le b$, $x \ge 0$ is totally dual integral. Since TDI is a weaker sufficient condition for the polytopes $\{Ax \le b, x \ge 0\}$ and $\{y^T A \ge c^T, y \ge 0\}$ to be integral (where b and c are integral vectors), it is interesting to know if a given system of inequalities is TDI.

Although the total dual integrality of the constraints describing Edmonds' matching polytope is well known and there are several proofs for it (Cook [5], Cunningham and Marsh [6], Hoffman and Oppenheim [25], and Schrijver [41]), to the best of our knowledge, ours is the fastest algorithm to compute an integral optimal dual solution when W is small. Note that this dual solution is a witness or certificate to the optimality of our matching.

Our algorithm also gives rise to a decomposition theorem. In a graph $G = (V, E)$ with edge weights in $\{1, \ldots, W\}$, we show that the maximum weight of a matching is exactly $\sum_{i=1}^{W} |M_i|$, where for each $i \in \{1, \ldots, W\}$, M_i is a maximum weight matching in a subgraph G_i of G with the edge set $E_i' = \{e \in E\colon w(e) \ge W - (i-1)\}$ and the weight function $w_i\colon E_i' \to \{1, \ldots, i\}$ defined as $w_i(e) = w(e) - (W - i)$. In particular, we show the following equation:

$$|M_i| = \sum_{u \in V} y_u^i + \sum_{B \in \Omega} z_B^i \left(\frac{|B| - 1}{2}\right) - \sum_{u \in V} y_u^{i-1} - \sum_{B \in \Omega} z_B^{i-1} \left(\frac{|B| - 1}{2}\right),$$

where $\langle y_u^i, z_B^i \rangle_{(u \in V,\, B \in \Omega)}$ is an integral optimal solution to the dual program for the maximum weight matching program in the graph G_i (note that y_u^0 and z_B^0 are just 0 for all $u \in V$ and $B \in \Omega$). Since the graph $G_W = G$, the above equation yields the decomposition theorem.

[1] In Kao et al. [27], a decomposition theorem of the maximum weight matching in bipartite graphs is given and their algorithm is then derived from it. However, in general graphs, the same theorem fails and the correctness of our algorithm rather follows from the complementary slackness conditions. See Section 2.2 for details.

TABLE 2. A summary of the current fastest maximum weight bipartite capacitated b-matching. Here $\beta = \sum_{v\in A\cup B} b(e)$, $B = \max_{v\in A\cup B} B(e)$, $C = \max_{e\in E} c(e)$, $n_1 = \min(|A|, |B|)$, and $SP_+(n, m, W)$ is the time needed to solve a shortest path problem in a digraph with n vertices, m arcs, and nonnegative edge length function l and $\sum_{e\in E} l(e) \le W$. Furthermore, the $\tilde{O}$ notation of Gabow and Sankowski's algorithm hides some factors of $\log \beta W$; the $\tilde{O}$ notation of Lee and Sidford's algorithm hides some factors of $\log m$.

Running time	Algorithm	Note
$O(n \log \beta \cdot SP_+(n, m, W))$	Lawler [29]	
$O(nm \log n^2/m \log nW)$	Goldberg and Tarjan [23, 24]	
$O(m \log n \cdot SP_+(n, m, W))$	Orlin [34, 35]	
$O((\sqrt{\beta}m + \beta \log \beta) \log nW)$	Gabow and Tarjan [19]	
$O((nm + \beta \log \beta) \log nW)$	Gabow and Tarjan [19]	
$O(n_1 m + n_1^3 \log n_1 W)$	Ahuja et al. [2]	
$O(n_1 m \log(2 + n_1^2/m) \log n_1 W)$	Ahuja et al. [2]	
$\tilde{O}(\sqrt{n}m \cdot \log^{O(1)} C)$	Lee and Sidford [31]	
$\tilde{O}(W\beta^{\omega})$	Gabow and Sankowski [18]	Only for simple graphs
$O(Wnm)$	This paper	
$O(W\sqrt{\beta}m)$	This paper	Only for simple b-matching ($c \equiv 1$)
$O(W(n_1 m + n_1^3))$	This paper	
$O(W(n_1 m + n_1^2\sqrt{m}))$	This paper	
$O(W(n_1 m + n_1^2\sqrt{\log C}))$	This paper	
$O(Wn_1 m \log(2 + n_1^2/m))$	This paper	

Maximum weight bipartite capacitated b-matching. Let $G = (A \cup B, E)$ be a bipartite graph with weight function $w\colon E \to \{1, \ldots, W\}$. Note that G need not be simple; i.e., multiple edges are allowed. Additionally, there is a *quota* $b\colon A \cup B \to Z_{>0}$ on the vertices and a *capacity* $c\colon E \to Z_{>0}$ on the edges. A function $M\colon E \to Z_{\ge 0}$ is a feasible solution if (1) $M(e) \le c(e)$ for every $e \in E$ and (2) $\sum_{e\in\delta(v)} M(e) \le b(v)$ for every $v \in A \cup B$. The goal is to find a feasible solution M so that $\sum_{e\in E} w(e)M(e)$ is maximized.

We show that this problem can also be decomposed into W unweighted (and capacitated) versions of the same problem. Using Orlin's new maximum flow algorithm (Orlin [36]) as a subroutine, we can solve the problem in $O(Wnm)$ time; in the case of simple b-matching (where $c \equiv 1$), we can use Gabow's algorithm (Gabow [14]) to solve the problem in $O(W\sqrt{\beta}m)$ time, where $\beta = \sum_{v\in A\cup B} b(e)$. Moreover, in the case that the graph G is very "unbalanced," i.e., the number of vertices on one side is much larger than the number on the other side, then we can use the algorithms of Ahuja et al. [2] as a subroutine, to solve the problem, in $O(W(n_1 m + n_1^3))$, $O(W(n_1 m + n_1^2\sqrt{m}))$, $O(W(n_1 m + n_1^2\sqrt{\log C}))$, or $O(Wn_1 m \log(2 + n_1^2/m))$ time, where $n_1 = \min\{|A|, |B|\}$ and $C = \max_{e\in E} c(e)$. See Table 2 for a summary of the fastest algorithms for this problem. As there are many parameters, it is difficult to compare the running time of these algorithms without resorting to case analysis. But *in general*, unless the largest capacity $C = \max_{e\in E} c(e)$ is really large, the recent algorithm of Lee and Sidford [31] is the fastest algorithm. Compared to theirs, our algorithms are faster only when the graph is very unbalanced, i.e., when $\min\{|A|, |B|\} = o(n)$ and W is $O(\text{poly-log}(m, C))$.

Our second algorithm differs from the first in that it does not seek to find the feasible dual solution in each iteration. A final adjustment step is performed to show that the produced dual solution and the primal matching satisfy complementary slackness conditions.

2. Maximum weight matching in general graphs.

In this section we present our algorithm for computing a maximum weight matching in $G = (V, E)$ with edge weights in $\{1, \ldots, W\}$. As mentioned in Section 1, our algorithm also computes an integral optimal dual solution along with the optimal matching M_W. We recall the linear program describing the matching polytope and its dual program below. Let Ω be the set of all odd sized subsets of V of size at least three, also referred to as *odd sets* of vertices. For any set $B \subseteq V$, let $E[B]$ be the set of edges (x, y) both of whose endpoints x and y belong to B. For any vertex v, let $E(v)$ be the set of edges incident on v.

$$\max \sum_{e\in E} w_e x_e \qquad\qquad \min \left\{ \sum_{v\in V} y_v + \sum_{B\in\Omega} z_B \frac{|B|-1}{2} \right\}$$

$$\sum_{e\in E(v)} x_e \le 1 \quad \forall v \in V \qquad\qquad y_u + y_v + \sum_{B:\, e\in E[B]} z_B \ge w_e \quad \forall e = (u, v) \in E$$

$$\sum_{e\in E[B]} x_e \le \frac{|B|-1}{2} \quad \forall B \in \Omega \qquad\qquad y_v \ge 0 \quad \forall v \in V$$

$$x_e \ge 0 \quad \forall e \in E. \qquad\qquad z_B \ge 0 \quad \forall B \in \Omega$$

We first briefly review some classical concepts on which our algorithm is built: Edmonds' blossom algorithm and Gallai-Edmonds decomposition. We highlight the main features of the blossom algorithm. More details can be found in Lovász and Plummer [32] and Schrijver [41]. In this and the next section, $M(u)$ refers to the vertex that is matched to u under the matching M.

Petersen [37] observed as early as 1891 that a matching M is of maximum cardinality if and only if there is no augmenting path with respect to M. It is not difficult to detect an augmenting path with respect to a given matching in bipartite graphs. But finding such a path in general graphs turns out to be more challenging. To overcome this difficulty, Edmonds introduced the idea of opening/closing blossoms.

Definition 1. Let $G = (V, E)$ be the original graph. Let $G(V_1)$ be a *shrunken* graph of G, defined as follows.
(i) $V_1 \subset V \cup \Omega$; i.e., each vertex $v \in V$ is either in V_1 or it belongs to an odd set in $V_1 \cap \Omega$,
(ii) Edges in $G(V_1)$ are induced by V_1. More precisely, (a, b) is an edge in $G(V_1)$ if and only if there exists an edge $(u, v) \in E$ so that (1) $u = a$ or $u \in a$ (where $a \in V_1 \cap \Omega$) and (2) $v = b$ or $v \in b$ (where $b \in V_1 \cap \Omega$).

Suppose $G(V_1)$ is a shrunken graph of G and M_1 is a matching in it. A set of vertices $B = (a_0, a_1, \ldots, a_{2t})$ is a *blossom* if (1) there exists a circuit traversing the vertices in B, i.e., $(a_i, a_{(i+1) \bmod 2t+1}) \in E_1$ for $0 \le i \le 2t$, and (2) $M(a_{2i-1}) = a_{2i}$, for $1 \le i \le t$. The first vertex a_0 is called the *base* of the blossom B. (Note that a_0 can be matched to some vertex in $V_1 \backslash B$, or it can be left unmatched.)

Closing a blossom. Suppose $B = (a_0, a_1, \ldots, a_{2t})$ is a blossom; then closing the blossom B means we form a new shrunken graph $G(V_2)$, where $V_2 = (V_1 \backslash \{a_i\}_{i=0}^{2t}) \cup \{B\}$, and a new matching M_2 in $G(V_2)$ as follows:

$$M_2(a) = M_1(a) \quad \text{if } a \notin B \cup \{M_1(a_0)\}; \qquad M_2(B) = M_1(a_0).$$

Notice that once the blossom B is closed in $G_1(V_1)$ to form a new shrunken graph $G(V_2)$, there can be another blossom B' in $G(V_2)$. It can happen that B in $G(V_2)$ (now a vertex in V_2) forms part of B'. In this case, B is said to be *embedded* in B'. A blossom not embedded in any other blossom is an *outermost* blossom. (Note that by definition, an outermost blossom must be a vertex in the last shrunken graph).

Definition 2. *Opening a blossom.* Let $G(V_1)$ be an shrunken graph of G derived from G by a sequence of closing blossoms. Let M_1 a matching in $G(V_1)$. Let B be an outermost blossom in V_1 and assume that $B = (a_0, a_1, \ldots, a_{2t})$, where a_is are the nodes corresponding to B when B is closed (note that a_i can be a vertex or a blossom). Opening the blossom B means that we form a new shrunken graph $G(V_2)$, where $V_2 = (V_1 \backslash \{B\}) \cup \{a_i\}_{i=0}^{2t}$ and create a new matching M_2 in $G(V_2)$ as follows:

- $M_2(a) = M_1(a)$ if $a \notin B \cup \{M_1(B)\}$.
- If B is unmatched in M_1, then a_0 is unmatched in M_2 as well and $M_2(a_{2i-1}) = a_{2i}, \forall 1 \le i \le t$.
- If $M_1(B) = a' \in V_1$, then choose $a_k \in B$ so that there is an edge in E connecting a vertex in a' and a vertex in a_k; furthermore, let

$$M_2(a_{(k+2i-1) \bmod (2t+1)}) = a_{(k+2i) \bmod (2t+1)}, \quad \forall\, 1 \le i \le t; \qquad M_2(a_k) = a'.$$

We now describe how Edmonds' blossom algorithm works. In each round, it seeks to find an augmenting path by building a Hungarian forest. A Hungarian forest is a disjoint set of trees, whose roots are unmatched nodes; every vertex in such a tree is connected to the root by an alternating path using the edges in the tree. In the process of building the Hungarian forest, a blossom may be detected. A detected blossom is then closed and the building of the Hungarian forest restarts with respect to the updated graph. After the closing of blossoms, if an augmenting path is found, then the matching is augmented along it. Furthermore, all blossoms are reopened (thus restoring the graph completely) and this round is terminated.

In the last round of the blossom algorithm, no augmenting path will be detected even after the closing of some blossoms. Let $\tilde{G} = (\tilde{V}, \tilde{E})$ denote the final (updated) graph, $\tilde{M}$ the current matching in it, and $\tilde{T}$ the final Hungarian forest that was constructed. Some of the vertices in $\tilde{G}$ can indeed be outermost blossoms. To avoid confusion, we refer to the vertices $\tilde{V}$ in $\tilde{G}$ as *nodes*. Note that $\tilde{M}$ is a maximum cardinality matching in $\tilde{G}$.

By Tutte-Berge formula (Berge [3], Tutte [42]), it can be shown that if we reopen all blossoms, then we have a maximum cardinality matching in the original graph. At the end of the blossom algorithm, we are left with a Hungarian forest and a set of matched edges in $\tilde{M}$ (and the latter cannot be reached by any alternating path starting from an unmatched node). Together they encode the Gallai-Edmonds decomposition (Edmonds [13], Gallai [21]):

- In $\tilde{T}$, we have a set of pairwise disjoint trees, whose roots are left unmatched in $\tilde{M}$. Each tree is composed of a set of alternating paths starting from the root. A node is *odd* (similarly, *even*) if there is an alternating path of odd (respectively, even) length starting from the root to this node.

- We have also the remaining matched edges in $\tilde{M}$ (that are not part of $\tilde{T}$). The endpoint nodes of these matched edges are *unreachable*.

Furthermore,

- All blossoms are (or are contained in) even nodes.
- There is no edge in the original graph between an even node and an even/unreachable node in $\tilde{V}$ (but notice that if an even node is a blossom, then G has some edges between its members).
- No edge between an odd node and an odd/unreachable node is present in $\tilde{M}$.

For convenience of presentation, in the following, when we say the Hungarian forest and use the notation $\tilde{T}$, we mean both the disjoint trees and those remaining matched edges not included in them.

We now show a simple proof that Edmonds' algorithm produces a maximum cardinality matching. Note that this proof is different from most found in the textbooks; it highlights the new ingredient in our approach.

PROPOSITION 1. *Let $\tilde{T}$ be the Hungarian forest at the end of Edmonds' blossom algorithm, with the node set $\tilde{V} \subset V \cup \Omega$ and $\tilde{M}$ the corresponding matching. Then the matching M obtained by opening all blossoms in $\tilde{V}$ is a maximum cardinality matching in G.*

PROOF. We show the optimality of M by constructing a dual solution $\langle y_u, z_B \rangle_{(u \in V,\, B \in \Omega)}$ such that this dual solution and M together satisfy complementary slackness conditions. For each vertex v, the value y_v is defined as follows:

- If v is an odd node in $\tilde{T}$, then $y_v = 1$;
- If v is an even node in $\tilde{T}$ or is contained in an even node, then $y_v = 0$;
- Let $U \subseteq V$ be the set of remaining vertices—these are the unreachable vertices in $\tilde{T}$. Choose any one of them $u^* \in U$; set $y_{u^*} = 1$ and $y_u = 0$ for the remaining vertices in $u \in U \backslash \{u^*\}$.

For each odd set $B \in \Omega$, the value z_B is defined as follows:

- If $B \in \Omega$ is an outermost blossom in $\tilde{G}$, then $z_B = 1$.
- If $|U \backslash \{u^*\}| \geq 2$, then set $z_{U \backslash \{u^*\}} = 1$; for the remaining odd sets $B \in \Omega$, let $z_B = 0$.

We now verify that $\langle y_u, z_B \rangle_{(u \in V,\, B \in \Omega)}$ is a dual feasible solution in the original graph G: the nonnegativity conditions obviously hold. We need to check that all edges are properly covered.

- Edges incident on an odd vertex v are clearly covered since $y_v = 1$.
- It follows from Gallai-Edmonds decomposition that there is no edge between an even node and an even/unreachable node. If an edge $e = (u, v)$ has both endpoints in the same even node B in $\tilde{T}$, then $z_B = 1$.
- If an edge $e = (u, v)$ is incident on an unreachable node u, then the other endpoint v is either an odd or an unreachable node. In the former case, $y_v = 1$; in the latter case, either $v = u^*$ (then $y_{u^*} = 1$) or $e \in E[U \backslash \{u^*\}]$ (then $z_{U \backslash \{u^*\}} = 1$).

We now show that M and the above dual solution $\langle y_u, z_B \rangle_{(u \in V,\, B \in \Omega)}$ satisfy complementary slackness conditions:

- All vertices $v \in V$ with $y_v > 0$ are actually matched in M.
- All odd sets B with $z_B > 0$ have exactly $(|B| - 1)/2$ edges of $E[B]$ matched in M; this also includes the odd set $B = U \backslash \{u^*\}$.
- Every edge in M is between an odd node and an even node, or between 2 unreachable nodes, or within a blossom. Thus. all edges $e = (u, v) \in M$ are tight; i.e., $y_u + y_v + \sum_{B:\, e \in E[B]} z_B = 1$.

Hence by complementary slackness, $\langle y_u, z_B \rangle_{(u \in V,\, B \in \Omega)}$ is dual optimal and M is primal optimal, i.e., M is a maximum cardinality matching in G. □

A more well-known optimal dual (e.g., see Cook et al. [4, Exercise 5.30]) is the following: all dual variables remain the same as defined in the proof above except for the following two changes: (1) all unreachable nodes $v \in U$ have $y_u = 1/2$ and (2) the odd set $U \backslash \{u^*\}$ has $z_{U \backslash \{u^*\}} = 0$. In fact, this dual was used by the earlier version of our algorithm and also by the algorithm of Gabow. We choose not to use it because the final dual solution will only be half-integral here.

Observe that we use the variable $z_{U \backslash \{u^*\}}$ to cover all edges in $E[U \backslash \{u^*\}]$. Roughly speaking, in our algorithm, we will regard the odd set $U \backslash \{u^*\}$ as a "pseudo-blossom." This pseudo-blossom is *shrunk* or contracted into a single node and gets matched to u^* in M—we will show in the next section that this contraction lasts for exactly one iteration: this pseudo-blossom will be reopened during the next iteration. The details are described in the next section.

2.1. The algorithm. Our algorithm runs for W iterations: in the i-th iteration, the algorithm computes a maximum weight matching M_i in a graph $G_i = (V, E_i')$, where $E_i' = E_W \cup \cdots \cup E_{W+1-i}$ and $E_t = \{e \mid w(e) = t\}$ for $W + 1 - i \leq t \leq W$. The edge weights in G_i are given by the function w_i: $E_i' \to \{1, \ldots, i\}$ defined as $w_i(e) = w(e) - (W - i)$ for $e \in E_i'$. For simplicity of presentation, in case $e \notin E_i'$, we write $w_i(e) = 0$.

The optimality of the matching M_i in G_i will be established via the dual variables y_u^i, for each $u \in V$, and z_B^i, for each $B \in \Omega$. In the i-th iteration we compute integral values for these dual variables and ensure that the dual feasibility conditions ((1) and (5)) and complementary slackness conditions ((2)–(4)) are satisfied by these dual values and the matching M_i. This will establish the optimality of M_i and these dual values in G_i.

We will ensure that conditions (1)–(5) hold for all $i \in \{1, \ldots, W\}$ in our algorithm. In the last iteration, i.e., when $i = W$, the set E'_W of edges in G_W is the same as the original edge set E and its weight function w_W coincides with the original weight function w. This will guarantee that M_W is a maximum weight matching in the original graph G.

$$y_u^i + y_v^i + \sum_{B:\, e \in E[B]} z_B^i \geq w_i(e) \qquad \forall \text{ edges } e = (u, v) \text{ in } E'_i \tag{1}$$

$$y_u^i + y_v^i + \sum_{B:\, e \in E[B]} z_B^i = w_i(e) \qquad \forall \text{ edges } e = (u, v) \in M_i \tag{2}$$

$$y_u^i > 0 \Rightarrow u \text{ is matched in } M_i \qquad \forall\, u \in V \tag{3}$$

$$z_B^i > 0 \Rightarrow \frac{|B| - 1}{2} \text{ edges of } E[B] \text{ are in } M_i \qquad \forall\, B \in \Omega \tag{4}$$

$$y_u^i \geq 0 \text{ and } z_B^i \geq 0 \qquad \forall\, u \in V \text{ and } B \in \Omega. \tag{5}$$

The graph G_i and its function w_i are introduced mainly to facilitate the proof. In implementation, our algorithm does not work with $G_i = (V, E'_i)$. The actual *working graph* is an *unweighted* graph $H_i = (V_i, F_i)$. The vertex set $V_i \subset V \cup \Omega$ is decided by the previous iterations; the edge set F_i consists of edges $e \in E'_i$ that satisfy $w_i(e) - y_u^{i-1} - y_v^{i-1} - \sum_{B:\, e \in E(B)} z_B^{i-1} = 1$.

We now describe the i-th iteration: the graph is $H_i = (V_i, F_i)$ and the starting matching is $\tilde{M}_{i-1}$ (inherited from the previous iteration) and it will be the case that $\tilde{M}_{i-1} \subseteq F_i$. Edmonds' blossom algorithm is called to augment $\tilde{M}_{i-1}$ into $\tilde{M}_i$. Let $\tilde{V}_i$ be the resulting node set after the execution of Edmonds' algorithm.[2] Recall that $\tilde{V}_i$ can be partitioned into $\mathcal{O}_i \dot{\cup} \mathcal{U}_i \dot{\cup} \mathcal{E}_i$, where $\mathcal{O}_i$ is the set of odd nodes, $\mathcal{U}_i$ is the set of unreachable nodes, and $\mathcal{E}_i$ is the set of even nodes.

We then create a pseudo-blossom B_i^* to replace a subset of unreachable nodes, as discussed in the previous section, and a unreachable node u_i^* not in the subset is chosen to be B_i^*'s partner in $\tilde{M}_i$. (Note that the pseudo-blossom B_i^* is also closed as a regular blossom, i.e., the set of unreachable nodes in B_i^* are now replaced by a single node B_i^* and its incident edges are induced by the nodes contained in B_i^* and the nodes not in B_i^*.) We define the next round of dual variables $\langle y_u^i, z_B^i \rangle_{(u \in V,\, B \in \Omega)}$ based on $\langle y_u^{i-1}, z_B^{i-1} \rangle_{(u \in V,\, B \in \Omega)}$ and the decomposition $\mathcal{O}_i \dot{\cup} \mathcal{U}_i \dot{\cup} \mathcal{E}_i$.

We will guarantee that in the i-th iteration, in H_i, u_{i-1}^* has no other incident edge except (u_{i-1}^*, B_{i-1}^*) (see inequality (7) in Lemma 1 and the discussion immediately before that lemma). This guarantees that the edge (u_{i-1}^*, B_{i-1}^*) is never part of an augmenting path and the edge is never shrunk into other blossoms in the i-th iteration. Moreover, at the end of i-th iteration, either both B_{i-1}^* and u_{i-1}^* belong to $\mathcal{U}_i$ or the former is in $\mathcal{O}_i$ while the latter is in $\mathcal{E}_i$. In both cases, we will reopen the pseudo-blossom B_{i-1}^*; thus there will be at most one pseudo-blossom at the end of the i-th iteration. Note that opening a pseudo-blossom simply means to match the node originally matched with u_{i-1}^* and restore the other originally matched edges contained in B_{i-1}^*. We now give the algorithm formally below. Recall that for any $e \in E$, $w_i(e) = w(e) - (W - i)$.

1. *Initialization*: $y_v^0 = 0\ \forall v \in V$; $z_B^0 = 0\ \forall B \in \Omega$; $\tilde{M}_0 = \varnothing$; $V_1 = V$.
2. For $i = 1$ to W do
   (a) Let $H_i = (V_i, F_i)$, where $F_i = \{e = (u, v) \in E'_i \colon w_i(e) - y_u^{i-1} - y_v^{i-1} - \sum_{B:e \in E[B]} z_B^{i-1} = 1\}$.
   (b) Let $\tilde{M}_{i-1}$ be the initial matching in H_i.
   (c) Call Edmonds' blossom algorithm to augment $\tilde{M}_{i-1}$ so as to find a maximum cardinality matching $\tilde{M}_i$ in H_i.
   (d) Let $\tilde{V}_i$ be the resultant node set and let $\tilde{V}_i = \mathcal{O}_i \dot{\cup} \mathcal{U}_i \dot{\cup} \mathcal{E}_i$, where $\mathcal{O}_i$ (similarly, $\mathcal{U}_i$ or $\mathcal{E}_i$) is the set of odd (respectively, unreachable or even) nodes.
   (e) If $i = 1$ then choose an arbitrary node $u_1^* \in \mathcal{U}_1$. If $|\mathcal{U}_1 \backslash \{u_1^*\}| \geq 2$, then replace $\mathcal{U}_1 \backslash \{u_1^*\}$ by a pseudo-blossom B_1^* and let $(B_1^*, u_1^*) \in \tilde{M}_1$.

[2] Here in the description and also in step 2(c) of the algorithm, we use Edmonds' algorithm for ease of presentation. In actual implementation, we can use any other maximum cardinality algorithm to find a maximum cardinality $\tilde{M}_i$ in H_i, as long as $\tilde{M}_i$ is an augmentation of $\tilde{M}_{i-1}$ (that is, a node matched in $\tilde{M}_{i-1}$ must be matched in $\tilde{M}_i$ as well). By building the Hungarian forest according to $\tilde{M}_i$, we can shrink the blossoms and the resultant matching and nodes sets will be $\tilde{M}_i$ and $\tilde{V}_i$.

Else (i.e., $i > 1$) then choose an arbitrary node $u_i^* \in \mathcal{U}_i \backslash \{B_{i-1}^*, u_{i-1}^*\}$.

• If $|\mathcal{U}_i \backslash \{B_{i-1}^*, u_{i-1}^*, u_i^*\}| \geq 2$, then replace this set by a pseudo-blossom B_i^*; in case $|\mathcal{U}_i \backslash \{B_{i-1}^*, u_{i-1}^*, u_i^*\}| = 1$ and $\mathcal{U}_i \backslash \{B_{i-1}^*, u_{i-1}^*, u_i^*\}$ is a blossom B, then let $B_i^* = B$.

• Let $(B_i^*, u_i^*) \in \tilde{M}_i$.

[*We now update the dual values: if no explicit update happens for vertex u, then* $y_u^i = y_u^{i-1}$*; similarly* $z_B^i = z_B^{i-1}$ *for any odd set B unless otherwise stated.*]

(f) For each $v \in V$: if $v \in \mathcal{O}_i$ or $v \in B$ where $B \in \mathcal{O}_i$, then set $y_v^i = y_v^{i-1} + 1$.

(g) For each outermost blossom or pseudo-blossom $B \in \mathcal{O}_i$: set $z_B^i = z_B^{i-1} - 1$.

(h) If u_i^* is a vertex $v \in V$, then $y_v^i = y_v^{i-1} + 1$.
Else (u_i^* is a blossom $B \in \Omega$) set $y_v^i = y_v^{i-1} + 1$ for all $v \in B$ and $z_B^i = z_B^{i-1} - 1$.

(i) If $B_{i-1}^* \in \mathcal{U}_i$, then set $y_v^i = y_v^{i-1} + 1$ for each $v \in B_{i-1}^*$ and set $z_{B_{i-1}^*}^i = z_{B_{i-1}^*}^{i-1} - 1$.

(j) For each outermost blossom $B \in \mathcal{E}_i$ and for $B = B_i^*$: set $z_B^i = z_B^{i-1} + 1$.

(k) If u_{i-1}^* is a blossom and is in $\mathcal{U}_i$, then set $z_{u_{i-1}^*}^i = z_{u_{i-1}^*}^{i-1} + 1$.

(l) Suppose that $B \in \Omega \backslash \{u_i^*\}$ is a pseudo-blossom or an outermost blossom. If $z_B^i = 0$, then open the blossom B and recursively so any other blossom B' in B with $z_{B'}^i = 0$.

(m) Let V_{i+1} be the current node set and let $\tilde{M}_i$ be the corresponding matching.

3. Return the matching M_W by opening all blossoms in $\tilde{M}_W$.

Step 2 is the heart of the above algorithm and this step essentially consists of two parts: steps (a)–(e) compute the matching $\tilde{M}_i$ while steps (f)–(k) set the dual values. The matching $\tilde{M}_i$ gets further updated in step (l) by opening out all outermost blossoms whose z-value becomes 0 and now if there are new "outermost" blossoms (these blossoms were earlier embedded but due to the outermost blossom getting opened, these embedded ones become outermost) with z-value 0, these get opened and so on. The last step defines the node set V_{i+1} to be used in the next iteration.

The steps that update the dual values here are analogous to how dual values are set in Proposition 1. Here we make $y_u^i = y_u^{i-1} + 1$ for every vertex u in $\mathcal{O}_i$ and this includes vertices that belong to blossoms in $\mathcal{O}_i$. For outermost blossoms B in $\mathcal{O}_i$, we decrease z_B^i by 1. The y-value for vertices in $\mathcal{E}_i$ is unchanged and if B is an outermost blossom in $\mathcal{E}_i$, then we increase z_B^i by 1.

The node u_i^* is treated as an *odd* node while B_i^* is treated as an even node. Thus we make $y_u^i = y_u^{i-1} + 1$ for every vertex u in u_i^* and decrease the z-value of u_i^* by 1 (if u_i^* is a blossom) while the y-value for vertices in B_i^* is unchanged and we increase the z-value of B_i^* by 1. To compensate for this, in the $(i+1)$-st iteration, u_i^* will get treated as an *even* node while B_i^* will get treated as an odd node—this is seen here in steps (j) and (k) where the dual variable values for u_{i-1}^* and B_{i-1}^* are updated: u_{i-1}^* is treated as an even node while B_{i-1}^* is treated as an odd node.

Note that after we shrink the pseudo-blossom B_i^* and match it to u_i^*, the dual variables are set in such a way that in the next iteration, u_i^* has only one incident edge (u_i^*, B_i^*) in H_{i+1}. This can be seen by inequality (7).

LEMMA 1. *For each* $1 \leq i \leq W$, *conditions* (1)–(5) *stated earlier hold, where* M_i *is the matching derived from* $\tilde{M}_i$ *by opening all blossoms*; *conditions* (6) *and* (7) *stated below hold as well.*

$$\text{There is at most one pseudo-blossom } B_i^* \text{ in } V_{i+1} \text{ and if such a } B_i^* \text{ exists, then } z_{B_i^*}^i = 1. \tag{6}$$

$$\text{Given } e = (u, v) \in E_i',\ u = u_i^* \text{ (if } u_i^* \in V) \text{ or } u \in u_i^* \text{ (if } u_i^* \in \Omega), \text{ and } v \notin B_i^* \cup u_i^*, \text{ then } y_u^i + y_v^i + \sum_{B:\, e \in E[B]} z_B^i > w_i(e). \tag{7}$$

PROOF. We show by induction on i that conditions (1)–(7) hold for all $1 \leq i \leq W$. The base case is $i = 1$. The matching M_1 is the maximum cardinality matching obtained by running Edmonds' algorithm in the graph H_1 and the setting of dual variables in the first iteration of our algorithm corresponds exactly to the setting of dual variables in Proposition 1. Thus by the same arguments as in the proof of Proposition 1, conditions (1)–(5) hold.

It is also easy to see that condition (6) holds as there is at most one pseudo-blossom $B_1^* \in \mathcal{U}_1$ at the end of the first iteration. We now show condition (7). Since at the end of Edmonds' algorithm, all blossoms are in even nodes, it follows that $u_1^* \in \mathcal{U}_1$ is a vertex in V and we have $y_{u_1^*}^1 = 1$. Let $e = (u_1^*, v) \in E_1'$ where $v \notin B_1^*$; then v has to be in $\mathcal{O}_1$ (as there are no edges in $\mathcal{U}_1 \times \mathcal{E}_1$ in F_1 and $E_1' = F_1$). Thus we have $y_v^1 = 1$ and so $y_{u_1^*}^1 + y_v^1 + \sum_{B:e \in E[B]} z_B^i \geq 2 > w_1(e)$ since $w_1(e) = 1$.

We now assume conditions (1)–(7) hold for $i = k - 1$ and show that conditions (1)–(7) corresponding to $i = k$ hold as well.

Condition (1). We know by conditions (1) and (5) for $i = k-1$ that for any edge $e = (u, v)$ in E'_k, we have $y_u^{k-1} + y_v^{k-1} + \sum_{B: e \in E[B]} z_B^{k-1} \geq w_{k-1}(e)$. We now need to show the following inequality:

$$y_u^k + y_v^k + \sum_{B: e \in E[B]} z_B^k \geq w_{k-1}(e) + 1 = w_k(e). \tag{8}$$

Consider first the case when $e \in E[B]$, where $B \in \tilde{V}_k \cap \Omega$. If $B \in \mathscr{O}_k$, then $y_u^k = y_u^{k-1} + 1$, $y_v^k = y_v^{k-1} + 1$, while $z_B^k = z_B^{k-1} - 1$. Thus inequality (8) clearly holds. If $B \in \mathscr{C}_k$, then there is some blossom $B' \supseteq B$ in $\tilde{V}_k$ and we set $z_{B'}^k = z_{B'}^{k-1} + 1$, thereby ensuring inequality (8). If $B \in \mathscr{U}_k$, then we have the following cases: (i) $B = u_k^*$, (ii) $B = B_k^*$ or $B \subset B_k^*$, (iii) $B = u_{k-1}^*$, (iv) $B = B_{k-1}^*$. Note that cases (i) and (iv) are analogous to $B \in \mathscr{O}_k$ and cases (ii) and (iii) are analogous to $B \in \mathscr{C}_k$. Thus inequality (8) holds when $e \in E[B]$.

We now consider the case when u and v are not inside the same blossom in $\tilde{V}_k$. If one of u, v is in $\mathscr{O}_k$, then $y_u^k + y_v^k \geq y_u^{k-1} + y_v^{k-1} + 1$. Thus inequality (8) holds. If both u and v are in $\mathscr{C}_k$ and the edge (u, v) is in E'_k, then it has to be the case that $y_u^{k-1} + y_v^{k-1} + \sum_{B: e \in E[B]} z_B^{k-1} > w_{k-1}(e)$ (as there are no edges of F_k in $\mathscr{C}_k \times \mathscr{C}_k$ other than edges inside blossoms). Since $y_u^k \geq y_u^{k-1}$, $y_v^k \geq y_v^{k-1}$, $z_B^k = z_B^{k-1} = 0$ for every odd set B containing e, and $w_{k-1}(e) + 1 = w_k(e)$, inequality (8) again holds. The only case left is when one of u, v is in $\mathscr{U}_k$ and the other is in $\mathscr{C}_k \cup \mathscr{U}_k$.

Since F_k has no edges in $\mathscr{C}_k \times \mathscr{U}_k$, this means that $y_u^{k-1} + y_v^{k-1} + \sum_{B: e \in E[B]} z_B^{k-1} > w_{k-1}(e)$ and we again have $y_u^{k-1} + y_v^{k-1} + \sum_{B: e \in E[B]} z_B^{k-1} \geq w_{k-1}(e) + 1 = w_k(e)$. Thus, inequality (8) holds. Finally, if both u and v are in $\mathscr{U}_k$ and by listing all subcases here, it is easy to see that inequality (8) holds. We show one particular subcase, which is when $u = u_{k-1}^*$ (or $u \in u_{k-1}^*$). Here we use condition (7) for $i = k-1$ which states that if $v \notin B_{k-1}^*$, then $y_u^{k-1} + y_v^{k-1} + \sum_{B: e \in E[B]} z_B^{k-1} > w_{k-1}(e)$. Thus inequality (8) holds when $v \notin B_{k-1}^*$ and when $v \in B_{k-1}^*$, inequality (8) holds since $y_v^k = y_v^{k-1} + 1$.

Condition (2). Let $e = (u, v)$ be an edge in M_k. It follows from the definition of the edge set F_k that $y_u^{k-1} + y_v^{k-1} + \sum_{B: e \in E[B]} z_B^{k-1} = w_{k-1}(e)$. By going through the same case analysis as in the proof of inequality (8), we can show that $y_u^k + y_v^k + \sum_{B: e \in E[B]} z_B^k = w_k(e)$. For instance, when $u, v \in V_k$ (i.e., u and v do not belong to any odd set $B \in V_k$) and say $u \in \mathscr{O}_k$, then since an odd node can only be matched to an even node, it follows that $v \in \mathscr{C}_k$ and thus $y_u^k = y_u^{k-1} + 1$ while $y_v^k = y_v^{k-1}$ and $z_B^k = z_B^{k-1} = 0$ for any odd set B containing (u, v). Thus we have

$$y_u^k + y_v^k + \sum_{B: e \in E[B]} z_B^k = (y_u^{k-1} + 1) + y_v^{k-1} + \sum_{B: e \in E[B]} z_B^{k-1} = w_{k-1}(e) + 1 = w_k(e).$$

Condition (3). We are given that $y_u^k > 0$ and there are two cases here: $y_u^{k-1} > 0$ or $y_u^{k-1} = 0$. Consider the former case—here it follows from condition (3) for $i = k-1$ that u was matched in M_{k-1} and we now show that u continues to be matched in M_k. We first claim that all edges of $\tilde{M}_{k-1}$ are present in the edge set F_k, i.e., step 2(b) is correct. This follows from condition (2) for $i = k-1$: for each edge $e = (u, v) \in M_{k-1}$, we have

$$w_k(e) - y_u^{k-1} - y_v^{k-1} - \sum_{B: e \in E[B]} z_B^{k-1} = (w_{k-1}(e) + 1) - y_u^{k-1} - y_v^{k-1} - \sum_{B: e \in E[B]} z_B^{i-1} = 1.$$

Thus edge $e \in F_k$. Now condition (3) follows because $\tilde{M}_k$ is augmented from $\tilde{M}_{k-1}$ and because (pseudo-)blossom opening guarantees that vertices matched in M_{k-1} remain matched in M_k. Now consider the case $y_u^{k-1} = 0$. For y_u^k to become positive, it must be the case that $u \in \mathscr{O}_k \cup B_{k-1}^* \cup u_k^*$ (or $u = u_k^*$). In all three cases, u is matched in M_k.

Condition (4). This is similar to the above proof that condition (3) holds. We are given that $z_B^k > 0$ and if z_B^{k-1} is also positive, then it means that the odd set $B \in V_k$ and B remains a blossom throughout the k-th iteration. By the definition of opening a blossom, it follows that when M_k is derived from $\tilde{M}_k$, there are $(|B| - 1)/2$ edges of $E[B]$ in M_k. If $z_B^{k-1} = 0$, then it means that the (pseudo-)blossom B was newly formed in the k-th iteration. Thus, $B \in \tilde{V}_k \cap \mathscr{C}_k$ or $B = B_k^*$ or $B = u_{k-1}^*$ and it is easy to see that in all three cases, $(|B| - 1)/2$ edges in $E[B]$ are present in M_k.

Condition (5). The nonnegativity of y_u^k for all vertices u is clear because $y_u^k \geq y_u^{k-1}$ for each $u \in V$ and by condition (5) for $i = k-1$, we have $y_u^{k-1} \geq 0$ for all $u \in V$. Regarding z_B^k, we claim that at the end of the $(k-1)$-st iteration, all outermost blossoms $B \in V_k$, along with $B = B_{k-1}^*$, have $z_B^k > 0$, with the possible exception of u_{k-1}^*. This is because of Step 2(l) where we recursively opened up all outermost blossoms and pseudo-blossoms B that satisfy $z_B^{k-1} = 0$. Thus at the beginning of the k-th iteration, all outermost blossoms (other than possibly u_{k-1}^*) and

pseudo-blossoms B satisfy $z_B^k > 0$. Hence in spite of decreasing by 1 the z^k-value of some outermost blossoms and pseudo-blossom B_{k-1}^*, at the end of the k-th iteration we maintain the condition that $z_B^k \geq 0$ for all $B \in \Omega$.

Condition (6). By condition (7) for $i = k-1$, in the edge set E'_{k-1}, the only edge incident on the node u_{k-1}^* is (u_{k-1}^*, B_{k-1}^*) in H_i. Suppose there is some other edge $e = (u, v)$, where $u = u_{k-1}^*$ (or $u \in u_{k-1}^*$) and $v \notin B_{k-1}^* \cup u_{k-1}^*$, in the edge set F_k. Then it has to be the case that

$$y_u^{k-1} + y_v^{k-1} + \sum_{B:\, e \in E[B]} z_B^{k-1} = w_{k-1}(e).$$

Since $e \in E'_k \backslash E'_{k-1}$, we have $w_{k-1}(e) = 0$ while we know that $y_u^{k-1} \geq 1$ since $u = u_{k-1}^*$ (or $u \in u_{k-1}^*$). This contradicts the above equation as y_v^{k-1} and z_B^{k-1} are nonnegative for all $v \in V$ and $B \in \Omega$. Thus the only edge incident on u_{k-1}^* in the graph H_k is the edge $(u_{k-1}^*, B_{k-1}^*) \in \tilde{M}_{k-1}$. Hence in the k-th iteration of the algorithm, neither u_{k-1}^* nor B_{k-1}^* belongs to any augmenting path and so these nodes also cannot become a part of a newly formed blossom. Thus they remain matched to each other during the k-th iteration.

If B_{k-1}^* is a pseudo-blossom, then by condition (6) for $i = k-1$, we have $z_{B_{k-1}^*}^{k-1} = 1$. Thus, $z_{B_{k-1}^*}^k$ becomes zero because either $B_{k-1}^* \in \mathscr{O}_k$ or $B_{k-1}^* \in \mathscr{U}_k$; hence we open up B_{k-1}^* at the end of the k-th iteration. By condition (6) for $i = k-1$, we also know that there is no other pseudo-blossom at the end of the $(k-1)$-st iteration. Thus there is at most one pseudo-blossom B_k^* at the end of the k-th iteration. Finally, if B_k^* is a pseudo-blossom, then $z_{B_k^*}^{k-1} = 0$, so $z_{B_k^*}^k = 1$.

Condition (7). Suppose $e = (u, v)$ is an edge in E'_k with one endpoint $u \in u_k^*$ and the other endpoint $v \notin B_k^* \cup u_k^*$ (for simplicity of exposition, we assume u_k^* is a blossom here; the same proof holds when u_k^* is a vertex also). We consider two cases here: (i) $v \in \mathscr{E}_k \cup u_{k-1}^*$ and (ii) $v \in \mathscr{O}_k \cup B_{k-1}^*$.

We know that F_k has no edges in $\mathscr{U}_k \times \mathscr{E}_k$, also we have seen in the proof of condition (6) that the only edge incident on u_{k-1}^* in the graph H_k is the edge (u_{k-1}^*, B_{k-1}^*). Thus in case (i), i.e., when $v \in \mathscr{E}_k \cup u_{k-1}^*$, there is no edge (u, v) in F_k; hence we have $y_u^{k-1} + y_v^{k-1} + \sum_{B:\, e \in E[B]} z_B^{k-1} > w_{k-1}(e)$. Since $y_u^k = y_u^{k-1} + 1$, $y_v^k \geq y_v^{k-1}$, $\sum_{B:\, e \in E[B]} z_B^k = \sum_{B:\, e \in E[B]} z_B^{k-1} = 0$ (as there is no blossom $B \supseteq \{u, v\}$ either in V_k or in V_{k+1}) and $w_k(e) = w_{k-1}(e) + 1$, we have $y_u^k + y_v^k + \sum_{B:\, e \in E[B]} z_B^k > w_k(e)$.

We now consider case (ii). In this case $v \in \mathscr{O}_k \cup B_{k-1}^*$, so $y_v^k = y_v^{k-1} + 1 \geq 1$. We know that $y_u^k = y_u^{k-1} + 1 \geq 1$. If $e \notin E'_{k-1}$, then $w_k(e) = 1$. Thus we have $y_u^k + y_v^k + \sum_{B:\, e \in E[B]} z_B^k \geq 2 > w_k(e)$. Suppose $e \in E'_{k-1}$. We have $y_u^k + y_v^k + \sum_{B:\, e \in E[B]} z_B^k = (y_u^{k-1} + 1) + (y_v^{k-1} + 1) + \sum_{B:\, e \in E[B]} z_B^{k-1}$. Since $y_u^{k-1} + y_v^{k-1} + \sum_{B:\, e \in E[B]} z_B^{k-1} \geq w_{k-1}(e)$ for all edges in E'_{k-1}, we have $y_u^k + y_v^k + \sum_{B:\, e \in E[B]} z_B^k \geq w_{k-1}(e) + 2 \geq w_k(e) + 1$. This finishes the proof that conditions (1)–(7) hold for $i = k$ as well. □

Thus, for each i, M_i is a matching in G_i and $\langle y_u^i, z_B^i \rangle_{(u \in V,\, B \in \Omega)}$ is a setting of dual variables such that conditions (1)–(5) are satisfied. This immediately proves that M_i is an optimal solution for the primal program of the i-th iteration; in other words, M_i is a maximum weight matching in the graph G_i. Similarly, $\langle y_u^i, z_B^i \rangle_{(u \in V,\, B \in \Omega)}$ is an (integral) optimal solution for the dual program of the i-th iteration. Hence we can conclude Theorem 1.

THEOREM 1. *The matching M_W returned by the algorithm is a maximum weight matching. Furthermore, the variables $\langle y_u^W, z_B^W \rangle_{(u \in V,\, B \in \Omega)}$ are an integral optimal solution for the dual program. Thus the linear program describing the matching polytope is totally dual integral.*

Since the maximum weight matching problem in a graph $G = (V, E)$ with edge weights in $\{1, \ldots, W\}$ can be solved as W maximum cardinality matching problems, we can draw the following computational conclusion. Recall that m and n are the number of edges and number of vertices in G, respectively.

THEOREM 2. *The maximum weight matching problem in G can be solved in $O(W\sqrt{n}m \log_n(n^2/m))$ time, or in $O(Wn^\omega)$ time with high probability, using the algorithms of Goldberg and Karzanov* [22] *and Mucha and Sankowski* [33] *as a subroutine, where $\omega \approx 2.3728$ is the exponent of matrix multiplication.*

PROOF The bottleneck in each iteration is in step 2(c), where the maximum cardinality matching in H_i gets computed, and we need to spend $O(\sqrt{n}m \log_n(n^2/m))$ or $O(n^\omega)$ time, using the algorithms of Goldberg and Karzanov [22] and Mucha and Sankowski [33]. Each of the other parts of step 2 can be done in $O(m)$ time.

In case we do not use Edmonds' blossom algorithm but use Mucha and Sankowski's algebraic algorithm in step 2(c), then to ensure condition (3), where we claim that the new maximum cardinality matching in H_i is augmented from $\tilde{M}_{i-1}$, we can do the following: first find *any* maximum cardinality matching in H_i and let its cardinality be t. Create $|V_i| - 2t$ dummy vertices and connect each of them to all nodes in V_i that are left unmatched by $\tilde{M}_{i-1}$. It is easy to see that there is now a perfect matching and we can find it by running the maximum cardinality

matching algorithm again. Moreover, the perfect matching so found must guarantee that only the nodes in V_i left unmatched by $\tilde{M}_{i-1}$ can be matched to the dummy vertices and the rest of the matching is the desired maximum cardinality matching $\tilde{M}_i$ in H_i. In step 2(d) we can build the Hungarian forest according to $\tilde{M}_i$ to define the resulting vertex set $\tilde{V}_i = \mathscr{O}_i \dot{\cup} \mathscr{U}_i \dot{\cup} \mathscr{E}_i$ in $O(m)$ time. □

2.2. Consequences of the above algorithm: A decomposition theorem. Our algorithm gives rise to the following decomposition theorem in a graph $G = (V, E)$ with edge weights in $\{1, \ldots, W\}$. Define graphs $G_1, \ldots, G_W$ as follows: $G_i = (V, E_i')$ where $E_i' = \{e \in E : w(e) \geq W - (i-1)\}$ with edge weight function $w_i(e) = w(e) - (W - i)$ for each $e \in E_i'$.

THEOREM 3. *There exist matchings $M_1, \ldots, M_W$ and dual solutions $\langle y_u^i, z_B^i \rangle_{(u \in V,\, B \in \Omega)}$ for $i = 1, \ldots, W$ such that the following properties hold*:

1. *For $1 \leq i \leq W$, M_i is a maximum weight matching in G_i and $\langle y_u^i, z_B^i \rangle_{(u \in V,\, B \in \Omega)}$ is an optimal dual solution.*
2. *For $1 \leq i \leq W$, $\langle y_u^i, z_B^i \rangle_{(u \in V,\, B \in \Omega)}$ is an integral solution; furthermore, the set of odd sets B with $z_B^i > 0$ forms a laminar family.*
3. *For $1 \leq i \leq W$, $|M_i| = \sum_{u \in V} y_u^i + \sum_{B \in \Omega} z_B^i((|B| - 1)/2) - \sum_{u \in V} y_u^{i-1} - \sum_{B \in \Omega} z_B^{i-1}((|B| - 1)/2)$, where $y_u^0 = 0 \; \forall \, u \in V$ and $z_B^0 = 0 \; \forall \, B \in \Omega$.*
4. *The maximum weight of a matching in G is equal to $\sum_{i=1}^{W} |M_i|$.*

PROOF. For $1 \leq i \leq W$, we will show that the matching M_i and the dual solution $\langle y_u^i, z_B^i \rangle_{(u \in V,\, B \in \Omega)}$ computed in the i-th iteration of our algorithm satisfy all the four parts of the above theorem. Part 1 is a corollary of Lemma 1.

Part 2 follows in a straightforward manner from our algorithm—it is easy to see that y_u^i and z_B^i are integral for all vertices u and odd sets B. The fact that the sets B with $z_B^i > 0$ form a laminar family follows from how the node set $\tilde{V}_i$ is formed and how the z-values are assigned.

We now show part 3 of the above theorem. It follows from conditions (1)–(5) that

$$\sum_{e \in M_i} w_i(e) = \sum_{u \in V} y_u^i + \sum_{B \in \Omega} z_B^i \left(\frac{|B| - 1}{2} \right).$$

We know that for each edge e, $w_i(e) = w_{i-1}(e) + 1$. Thus $\sum_{e \in M_i} w_i(e) = \sum_{e \in M_i} w_{i-1}(e) + |M_i|$. Since every edge used in M_i belongs to the edge set F_i, we have $w_{i-1}(e) = y_u^{i-1} + y_v^{i-1} + \sum_{B:\, e \in E[B]} z_B^{i-1}$ for edge $e = (u, v) \in M_i$. Also, the vertices that are not matched in M_i are unmatched in M_{i-1} as well. Thus $y_u^{i-1} = 0$ for all vertices unmatched in M_i (by condition (3)). Hence we have

$$\sum_{u \in V} y_u^i + \sum_{B \in \Omega} z_B^i \left(\frac{|B| - 1}{2} \right) = |M_i| + \sum_{u \in V} y_u^{i-1} + \sum_{e \in M_i} \sum_{B:\, e \in E[B]} z_B^{i-1}.$$

What is left to show is that $\sum_{e \in M_i} \sum_{B:\, e \in E[B]} z_B^{i-1} = \sum_{B \in \Omega} z_B^{i-1}((|B| - 1)/2)$. By rearranging the terms in the sum, we have $\sum_{e \in M_i} \sum_{B:\, e \in E[B]} z_B^{i-1} = \sum_{B \in \Omega} z_B^{i-1} \cdot |E[B] \cap M_i|$. Thus we just need to show that for each $B \in \Omega$ with $z_B^{i-1} > 0$, exactly $(|B| - 1)/2$ edges of $E[B]$ are present in M_i.

Consider any such B. Since $z_B^{i-1} > 0$, at the beginning of the i-th iteration of our algorithm, the odd set B is shrunk in the node set $\tilde{V}_i$. At the end of the i-th iteration, either $z_B^i > 0$, in which case $(|B| - 1)/2$ edges of B are present in M_i (by condition (4)), or z_B^i becomes zero, in which case by the process of opening a blossom, $(|B| - 1)/2$ edges of $E[B]$ are present in M_i. This completes the proof of part 3.

Part 4 follows from adding the equations of part 3 for all $i \in \{1, \ldots, W\}$. Since the right-hand side consists of a cascading sum and $y_u^0 = 0$ for all u and $z_B^0 = 0$ for all B, this results in

$$\sum_{i=1}^{M} |M_i| = \sum_{u \in V} y_u^W + \sum_{B \in \Omega} z_B^W \left(\frac{|B| - 1}{2} \right).$$

We also know that the above right side is the optimum value for the dual program in the W-th iteration. This equals the value of the optimal primal solution, which is the maximum weight of a matching in G_W. Since $G_W = G$, the maximum weight of a matching in G equals $\sum_{i=1}^{W} |M_i|$. □

It may be tempting to try to generalize the following decomposition theorem of Kao et al. [27] to the context of general graphs.

THEOREM 4 (FROM KAO ET AL. [27]). *Let G be bipartite. Let G' be G_i with the weight function w_i for some $i \in \{1, \ldots, W\}$. Let $\langle y_u^i \rangle_{u \in V}$ be any minimum weight cover in G_i and G'' be the subgraph of G with the edge set $\{e = (a, b) : w(e) - y_a^i - y_b^i > 0\}$ with weight of edge e given by $\tilde{w}(e) = w(e) - y_a^i - y_b^i$. The maximum weight of a matching in G is then equal to the sum of the maximum weights of a matching in G' and in G''.*

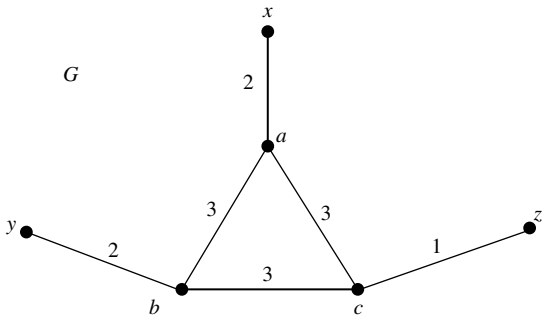

FIGURE 1. The numbers on edges denote their weights and the bold edges (x, a) and (b, c) are the ones in the maximum weight matching. The weight of a maximum weight matching in G is 5.

Unfortunately, the above theorem with $\tilde{w}(e)$ updated to $w(e) - y_u^i - y_u^i - \sum_{B: e \in E[B]} z_B^i$ for $e = (u, v)$ need not hold in nonbipartite graphs. Consider the example in Figure 1.

The maximum weight of a matching in G is 5. Let $G' = G_1$, so the weight function is $w_1(e) = w(e) - 2$; thus we have $w_1(e) = 1$ for $e \in \{(a, b), (b, c), (c, a)\}$ while $w_1(e) \leq 0$ for $e \in \{(a, x), (b, y), (c, z)\}$. The maximum weight of a matching in G' is 1 and there is a unique optimal dual solution: $z^1_{\{a,b,c\}} = 1$; all other z-values and y-values are 0. In G'', the weight of edge $e = (u, v)$ is $\tilde{w}(e) = w(e) - y_u^1 - y_v^1 - \sum_{B: e \in E[B]} z_B^1$ (see Figure 2). The maximum weight of a matching in G'' is 5. However the maximum weight of a matching in G is *not* $1 + 5 = 6$.

3. Maximum weight bipartite capacitated b-matching. The input here is a bipartite graph $G = (A \cup B, E)$ where as before, there is a weight function w: $E \to \{1, \ldots, W\}$. Associated with each vertex is a quota given by b: $A \cup B \to Z^+$ and the goal is to compute a *b-matching* of maximum weight. A b-matching is a subset $M \subseteq E$ such that for any vertex u, at most $b(u)$ edges incident on u are present in M. When $b(u) = 1$ for each vertex u, the resulting b-matching is the standard matching studied in the previous section.

In fact, here we consider an even more generalized concept called "capacitated b-matchings": several copies of an edge e can be present in M. That is, there is an edge capacity function c: $E \to Z^+$ and up to $c(e)$ copies of edge e are allowed in M. Recall that $E(v)$ is the set of edges incident on vertex v. For any vertex v, we assume that $b(v) \leq \sum_{e \in E(v)} c(e)$.

Associated with every capacitated b-matching M is an m-tuple $(M(e_1), \ldots, M(e_m))$ where $0 \leq M(e_i) \leq c(e_i)$ for every edge e_i. For simplicity, we refer to capacitated b-matchings as b-matchings here. When $w(e) = 1$ for all edges e, a maximum weight b-matching M maximizes $\sum_{e \in E} M(e)$, so we will call such a matching a *maximum cardinality* b-matching.

We will solve the maximum weight b-matching problem in bipartite graphs by reducing it to several instances of the maximum cardinality b-matching problem. We will use the following terminology here.

- An edge e is *saturated* in M if $M(e) = c(e)$; otherwise, e is *unsaturated*. Similarly, a vertex v is *saturated* if $\sum_{e \in E(v)} M(e) = b(v)$; otherwise, v is *unsaturated*.
- An edge e with $M(e) > 0$ is a *positive* edge.

DEFINITION 3. An *alternating* path with respect to a b-matching M is a path $p = \langle e_0, e_1, \ldots, e_k \rangle$ in G such that $e_0, e_2, \ldots$ are unsaturated edges while $e_1, e_3, \ldots$ are positive edges. That is, unsaturated edges and positive edges alternate in p.

An alternating path p of *odd* length with respect to a b-matching M such that both the endpoints of p are unsaturated in M is also called an "augmenting path" with respect to M. It is easy to see that a maximum cardinality

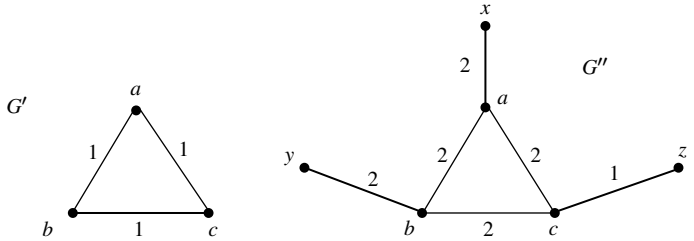

FIGURE 2. In the graph G', the edges with nonpositive weight have not been shown above. The weight of a maximum weight matching in G' is 1 and that in G'' is 5.

b-matching admits no augmenting path with respect to it. We present below a generalization of the coarse version of Dulmage-Mendelsohn decomposition of bipartite graphs (Dulmage and Mendelsohn [11]). We first need the following definition.

DEFINITION 4. Let M be a maximum cardinality b-matching in $G = (A \cup B, E)$.

- Let $\mathcal{E}_A \subseteq A$ (similarly, $\mathcal{O}_B \subseteq B$) be the set of vertices reachable from an unsaturated vertex in A via an alternating path of even (respectively, odd) length with respect to M.
- Let $\mathcal{E}_B \subseteq B$ (similarly, $\mathcal{O}_A \subseteq A$) be the set of vertices reachable from an unsaturated vertex in B via an alternating path of even (respectively, odd) length with respect to M.
- Let $\mathcal{U}_A = A \backslash (\mathcal{E}_A \cup \mathcal{O}_A)$ and let $\mathcal{U}_B = B \backslash (\mathcal{E}_B \cup \mathcal{O}_B)$.

PROPOSITION 2. *Let M be a maximum cardinality b-matching in $G = (A \cup B, E)$ and $\mathcal{E}_A$, $\mathcal{E}_B$, $\mathcal{O}_A$, $\mathcal{O}_B$, $\mathcal{U}_A$, $\mathcal{U}_B$ be the sets of vertices as defined in Definition 4. Furthermore, let M' be an arbitrary maximum cardinality b-matching in G; then the following holds.*

(1) *The three sets $\mathcal{O}_A$, $\mathcal{E}_A$, and $\mathcal{U}_A$ are pairwise disjoint; so are $\mathcal{O}_B$, $\mathcal{E}_B$, and $\mathcal{U}_B$.*
(2) *All vertices in $\mathcal{O}_A \cup \mathcal{O}_B \cup \mathcal{U}_A \cup \mathcal{U}_B$ are saturated in M'.*
(3) *There is no positive edge in $(\mathcal{U}_A \cup \mathcal{O}_A) \times \mathcal{O}_B$ or in $\mathcal{O}_A \times \mathcal{U}_B$ in M'.*
(4) *Every edge in $\mathcal{E}_A \times (\mathcal{E}_B \cup \mathcal{U}_B)$ is saturated in M'; so is every edge in $\mathcal{U}_A \times \mathcal{E}_B$ in M'.*

PROOF. To show (1), we first show that $\mathcal{O}_A$, $\mathcal{E}_A$, and $\mathcal{U}_A$ are disjoint from one another. It follows from the definition of $\mathcal{U}_A$ that $\mathcal{U}_A \cap (\mathcal{E}_A \cup \mathcal{O}_A) = \varnothing$. Thus, what is left to show is that $\mathcal{E}_A \cap \mathcal{O}_A = \varnothing$. Suppose $v \in \mathcal{E}_A \cap \mathcal{O}_A$—then gluing p_1 and p_2 till their first common vertex yields an augmenting path p with respect to M, where p_1 is the even length alternating path between some unsaturated vertex $a \in A$ and v (such a path p_1 exists because $v \in \mathcal{E}_A$) and p_2 is the odd length alternating path between some unsaturated vertex $b \in B$ and v (such a path p_2 exists because $v \in \mathcal{O}_A$). The augmenting path p contradicts that M is a maximum cardinality b-matching. A similar argument shows that $\mathcal{O}_B$, $\mathcal{E}_B$, and $\mathcal{U}_B$ are disjoint from one another. This finishes the proof of (1).

In the following, we first prove (2)–(4) assuming that $M' = M$; we will later remove this assumption.

It follows from the definition of $\mathcal{U}_A$ and $\mathcal{U}_B$ that every vertex u in $\mathcal{U}_A \cup \mathcal{U}_B$ has to be saturated in M (otherwise there is a length zero alternating path from an unsaturated vertex to u, contradicting that $u \in \mathcal{U}_A \cup \mathcal{U}_B$). If a vertex $a \in \mathcal{O}_A$ is unsaturated, then there is an alternating path of odd length from some unsaturated vertex in B to the unsaturated vertex a, i.e., M admits an augmenting path, a contradiction. Similarly every vertex in $\mathcal{O}_B$ is saturated. This finishes the proof of (2) when $M' = M$.

Suppose there is an edge $e \in \mathcal{O}_A \times \mathcal{O}_B$ such that $M(e) > 0$. Let $e = (u, v)$. We can again show an augmenting path with respect to M between an unsaturated vertex $a \in A$ and an unsaturated vertex $b \in B$ using the paths p_1, p_2 and the edge (u, v), where p_1 is the (odd length) a-u alternating path and p_2 is the (odd length) b-v alternating path. Similarly, if $e = (u, v)$ in $\mathcal{U}_A \times \mathcal{O}_B$ satisfies $M(e) > 0$, then there is an alternating path from some unsaturated vertex in A to u; this contradicts the fact that $u \in \mathcal{U}_A$. Similarly, there is no positive edge in $\mathcal{O}_A \times \mathcal{U}_B$. This finishes the proof of (3) when $M' = M$.

Suppose there is an edge $e \in \mathcal{E}_A \times \mathcal{E}_B$ such that $M(e) < c(e)$. Let $e = (u, v)$. We can again show an augmenting path between some unsaturated vertex $a \in A$ and some unsaturated vertex $b \in B$ using p_1, p_2, and (u, v), where p_1 is the (even length) a-u alternating path and p_2 is the (even length) b-v alternating path. Similarly, if $e = (u, v)$ in $\mathcal{U}_A \times \mathcal{E}_B$ satisfies $M(e) < c(e)$, then there is an alternating path between some unsaturated $b \in B$ and u; this contradicts the fact that $u \in \mathcal{U}_A$. Similarly, there is no unsaturated edge in $\mathcal{E}_A \times \mathcal{U}_B$. This finishes the proof of (4) when $M' = M$.

We now remove the assumption that $M' = M$. Let M' be an arbitrary maximum cardinality b-matching in G. Let us define a bipartite directed flow f as follows: $f(e) = |M'(e) - M(e)|$. The direction of the edge $e = (u, v)$ under f is from u to v if $M'(e) > M(e)$ and from v to u if $M'(e) < M(e)$. It is clear that M' is the sum of M and the flow f. Let us summarize what we know about f using what we have proved so far.

(i) Under f, there is no edge going from $\mathcal{E}_A$ to $\mathcal{U}_B \cup \mathcal{E}_B$ and no edge from $\mathcal{U}_A$ to $\mathcal{E}_B$,
(ii) Under f, there is no edge going from $\mathcal{O}_B$ to $\mathcal{U}_A \cup \mathcal{O}_A$, and no edge from $\mathcal{U}_B$ to $\mathcal{O}_A$,
(iii) For all vertices in $\mathcal{O}_A \cup \mathcal{U}_A \cup \mathcal{O}_B \cup \mathcal{U}_B$, the amounts of incoming flow and outgoing flow are equivalent.

It is well known, e.g., Ahuja et al. [1], that f can be decomposed into a set of cycle flows C_i and a set of path flows P_j by a greedy algorithm. By (i) and (ii), the cycle C_i consists entirely of vertices in $\mathcal{E}_A \cup \mathcal{O}_A$ or $\mathcal{E}_B \cup \mathcal{O}_B$. Furthermore, by (iii), the starting and ending vertices of P_j can only be in $\mathcal{E}_A \cup \mathcal{E}_B$. By (i) and (ii), P_j cannot start from $\mathcal{E}_A$ and end in $\mathcal{E}_B$. Moreover, we cannot have P_j start from $\mathcal{E}_B$ and end in $\mathcal{E}_A$ as it would imply an augmenting path in M' from $\mathcal{E}_A$ to $\mathcal{E}_B$. We can thus conclude that the cycle C_i and P_j consist entirely of vertices in $\mathcal{E}_A \cup \mathcal{O}_A$ or in $\mathcal{E}_B \cup \mathcal{O}_B$ and the proof follows. □

Note that if a maximum cardinality b-matching M is given, the decomposition $A = \mathcal{O}_A \cup \mathcal{E}_A \cup \mathcal{U}_A$ and $B = \mathcal{O}_B \cup \mathcal{E}_B \cup \mathcal{U}_B$ with respect to M can be determined in $O(m+n)$ time easily. We are now ready to describe our algorithm to compute a maximum weight b-matching in $G = (A \cup B, E)$. The linear program corresponding to the maximum weight b-matching problem and the dual LP are given below.

$$\max \sum_{e\in E} w(e)x_e \qquad \min \left\{ \sum_{v\in V} b_v y_v + \sum_{e\in E} c(e) z_e \right\}$$

$$\sum_{e\in E(v)} x_e \le b(v) \quad \forall v \in A\cup B. \qquad y_a + y_b + z_e \ge w(e) \quad \forall e = (a,b) \in E.$$

$$0 \le x_e \le c(e) \quad \forall e \in E. \qquad y_v \ge 0 \quad \forall v \in A \cup B.$$

$$z_e \ge 0 \quad \forall e \in E.$$

Our algorithm is similar in spirit to the algorithm from Section 2.1 but it also has some subtle differences from that one. The algorithm here also runs for W iterations, where $W = \max_{e\in E} w(e)$. In the first iteration, we consider only edges of weight W: this is the graph H_1. We compute a maximum cardinality b-matching M_1 here and assign vertex potentials y^1_u; however not all edges get covered by these vertex potentials, i.e., there exist edges $e = (a,b)$ in H_1 such that $y^1_a = y^1_b = 0$. More precisely, all the uncovered edges are in $\mathcal{E}^1_A \times (\mathcal{E}^1_B \cup \mathcal{U}^1_B)$; we also know from Proposition 2(4) that each edge in $\mathcal{E}^1_A \times (\mathcal{E}^1_B \cup \mathcal{U}^1_B)$ is *saturated* in M_1.

In fact, this will be an important invariant that we will maintain: every uncovered or "not fully paid for" edge will be saturated. In general, in iteration i we have an *unsatisfied set* $\Psi_i \subseteq E$, which consists of those edges $e = (a,b)$ such that $y^{i-1}_a + y^{i-1}_b < w_{i-1}(e)$ and it will be the case that e is saturated by the previous maximum cardinality b-matching M_{i-1}.

In the i-th iteration, the algorithm works with the unweighted graph $H_i = (A \cup B, F_i)$, where $F_i = \{e = (a,b) \in E\colon w_i(e) - y^{i-1}_a - y^{i-1}_b = 1\}$ and $w_i(e) = w(e) - (W - i)$. Furthermore, the quotas of the vertices are updated according to the unsatisfied set Ψ_{i-1}, namely, $b_i(v) = b(v) - \sum_{e\in\Psi_{i-1}\cap E(v)} c(e)$, where $E(v)$ is the set of edges incident on vertex v. Our task is to obtain a maximum cardinality b-matching in H_i, where vertex quotas are described by the function b_i; we obtain such a matching M'_i by augmenting $M_{i-1}\setminus\Psi_{i-1}$ in the graph H_i. The matching M_i will be M'_i along with all edges in Ψ_{i-1} (these edges are all saturated). We will show that the final matching M_W is a maximum weight b-matching in G. We present our algorithm below.

1. *Initialization*: Set $M_0 = \varnothing$, $\Psi_0 = \varnothing$, and $y^0_v = 0\ \forall v \in V$.
2. For $i = 1$ to W do
   (a) For each $e \in E$: let $w_i(e) = w(e) - (W - i)$.
   (b) Construct $H_i = (A \cup B, F_i)$, where $F_i = \{e = (a,b)\colon w_i(e) - y^{i-1}_a - y^{i-1}_b = 1\}$.
   (c) For each $v \in A \cup B$ do: set $b_i(v) = b(v) - \sum_{e\in\Psi_{i-1}\cap E(v)} c(e)$.
   (d) Find a maximum cardinality b-matching M'_i in H_i (where the quotas of vertices are given by b_i) by augmenting $M_{i-1}\setminus\Psi_{i-1}$ in H_i.
   (e) Using M'_i, partition $A = \mathcal{O}^i_A \cup \mathcal{E}^i_A \cup \mathcal{U}^i_A$ and $B = \mathcal{O}^i_B \cup \mathcal{E}^i_B \cup \mathcal{U}^i_B$.
      - For each $v \in \mathcal{O}^i_A \cup \mathcal{O}^i_B \cup \mathcal{U}^i_A$ do: set $y^i_v = y^{i-1}_v + 1$.
      - For each $v \in \mathcal{E}^i_A \cup \mathcal{E}^i_B \cup \mathcal{U}^i_B$ do: set $y^i_v = y^{i-1}_v$.
   (f) Let M_i be $M'_i \cup \Psi_{i-1}$, where $M_i(e) = c(e)$ for each $e \in \Psi_{i-1}$.
   (g) Let $S_i = \{e = (a,b) \in F_i$ where $a \in \mathcal{E}^i_A$ and $b \in \mathcal{E}^i_B \cup \mathcal{U}^i_B\}$;
   let $C_i = \{e = (a,b) \in \Psi_{i-1}$ such that $w_i(e) - y^i_a - y^i_b = 0\}$.
   (h) $\Psi_i = (\Psi_{i-1}\setminus C_i) \cup S_i$.
3. Return M_W.

It follows from the definition of F_i and from step 2(e) (where vertex potentials are updated) that those edges of F_i with at least one endpoint in $\mathcal{O}^i_A \cup \mathcal{O}^i_B \cup \mathcal{U}^i_A$ are covered or paid for while those with both endpoints in $\mathcal{E}^i_A \times (\mathcal{E}^i_B \cup \mathcal{U}^i_B)$ are uncovered or not fully paid for. The edges of F_i in $\mathcal{E}^i_A \times (\mathcal{E}^i_B \cup \mathcal{U}^i_B)$ form the set S_i and these are added to the set Ψ_i in step 2(h).

Regarding the edges that are already in the set Ψ_{i-1}, it could be the case that some of them get covered now (because of step 2(e) where vertex potentials are updated)—for this to happen, it is necessary that both the endpoints of such an edge are in $\mathcal{O}^i_A \cup \mathcal{O}^i_B \cup \mathcal{U}^i_A$. These newly covered edges form the set C_i and these are no longer present in the unsatisfied set Ψ_i.

Lemma 2. *For each $1 \le i \le W$, the above algorithm maintains conditions (9)–(13) listed below.*

$$y^i_a + y^i_b \ge w_i(e) \qquad \forall e = (a,b) \in E\setminus\Psi_i \tag{9}$$

$$y^i_a + y^i_b < w_i(e) \qquad \text{and} \qquad M_i(e) = c(e) \qquad \forall e \in \Psi_i \tag{10}$$

$$y_u^i > 0 \Rightarrow u \text{ is saturated in } M_i \qquad \forall\, u \in A \cup B \tag{11}$$

$$y_a^i + y_b^i = w_i(e) \qquad \forall \text{ edges } e = (a,b) \in F_i \backslash S_i \text{ such that } M_i'(e) > 0 \tag{12}$$

$$y_u^i \geq 0 \qquad \forall\, u \in A \cup B. \tag{13}$$

Proof. We show by induction on i that conditions (9)–(13) hold for all $1 \leq i \leq W$. The base case corresponds to $i = 1$. The set $\Psi_1 = \mathcal{E}_A^1 \times (\mathcal{E}_B^1 \cup \mathcal{U}_B^1)$ and it is easy to see that conditions (9)–(13) for $i = 1$ follow from Proposition 2 and from how y_u^1-values are set in step 2(e).

We now assume that conditions (9)–(13) hold when $i = k-1$ and show that the conditions corresponding to $i = k$ hold as well. To show condition (9) for $i = k$, assume that $e \in E \backslash \Psi_k$. If $e \in \Psi_{k-1} \backslash \Psi_k$, then e has to be in C_k, in other words, $w_k(e) = y_a^k + y_b^k$. Thus, we can henceforth assume that $e \notin \Psi_{k-1}$. Thus, $y_a^{k-1} + y_b^{k-1} \geq w_{k-1}(e)$ by induction hypothesis. We consider two cases here: (i) $e \in F_k$ and (ii) $e \notin F_k$.

Since F_k consists of edges $e = (a,b)$ such that $w_k(e) - y_a^{k-1} - y_b^{k-1} = 1$ and $w_k(e) = w_{k-1}(e) + 1$, it is easy to see that F_k is exactly the set of those edges $e = (a,b)$ such that $y_a^{k-1} + y_b^{k-1} = w_{k-1}(e)$. If $e \notin F_k$, then we have $y_a^{k-1} + y_b^{k-1} > w_{k-1}(e)$, that is, $y_a^{k-1} + y_b^{k-1} \geq w_{k-1}(e) + 1$. Thus, we have

$$y_a^k + y_b^k \geq y_a^{k-1} + y_b^{k-1} \geq w_{k-1}(e) + 1 = w_k(e).$$

Suppose $e \in F_k$. In this case, at least one of $\{a, b\}$ has to be in $\mathcal{O}_A^k \cup \mathcal{O}_B^k \cup \mathcal{U}_A^k$; otherwise $e \in S_k$ and thus $e \in \Psi_k$. Since a or b (or both) is in $\mathcal{O}_A^k \cup \mathcal{O}_B^k \cup \mathcal{U}_A^k$, we have

$$y_a^k + y_b^k \geq y_a^{k-1} + y_b^{k-1} + 1 = w_{k-1}(e) + 1 = w_k(e).$$

Thus condition (9) holds for $i = k$.

We now show condition (10) for $i = k$. The set $\Psi_k = (\Psi_{k-1} \backslash C_k) \cup S_k$. It is easy to see that every $e = (a,b) \in S_k$ satisfies $y_a^k + y_b^k < w_k(e)$ and $M_k(e) = c(e)$: this is because every such edge belongs to $F_k \cap (\mathcal{E}_A^k \times (\mathcal{E}_B^k \cup \mathcal{U}_B^k))$ and thus $w_k(e) = w_{k-1}(e) + 1 = y_a^{k-1} + y_b^{k-1} + 1 = y_a^k + y_b^k + 1$ and by Proposition 2(4), $M_k(e) = c(e)$.

We now consider the case when $e \in \Psi_{k-1} \backslash C_k$. By induction hypothesis, every edge $e = (a,b)$ in Ψ_{k-1} satisfies $y_a^{k-1} + y_b^{k-1} < w_{k-1}(e)$ and $M_{k-1}(e) = c(e)$. In the k-th iteration we have

$$y_a^k + y_b^k \leq y_a^{k-1} + y_b^{k-1} + 2 \leq w_{k-1}(e) + 1 = w_k(e).$$

We are given that $e \notin C_k$, so $y_a^k + y_b^k \neq w_k(e)$. Thus $y_a^k + y_b^k < w_k(e)$. Since e is in Ψ_{k-1}, it follows that $M_k(e) = c(e)$.

We now show condition (11) for $i = k$. We are given that $y_u^k > 0$. If $y_u^{k-1} > 0$, then it follows from induction hypothesis that u is saturated in M_{k-1}; since M_k' is obtained by augmenting $M_{k-1} \backslash \Psi_{k-1}$ and $M_k = M_k' \cup \Psi_{k-1}$, it follows that $\sum_{e \in E(u)} M_{k-1}(u) = \sum_{e \in E(u)} M_k(u)$, and thus u is saturated in M_k. If $y_u^{k-1} = 0$, then it must be the case that $y_u^k \in \mathcal{O}_A^k \cup \mathcal{O}_B^k \cup \mathcal{U}_A$ and it follows from Proposition 2(2) that u is saturated in M_k.

We now show condition (12) for $i = k$. Let $e = (a,b)$ be an edge such that $M_k'(e) > 0$. We know that M_k' is a maximum cardinality b-matching in H_k: since the edge set of H_k is F_k, this implies that $y_a^{k-1} + y_b^{k-1} = w_{k-1}(e)$. We also know that $e \notin S_k$. It follows from Proposition 2(3) that $e \in (\mathcal{O}_A^k \times \mathcal{E}_B^k) \cup (\mathcal{E}_A^k \times \mathcal{O}_B^k) \cup (\mathcal{U}_A^k \times \mathcal{E}_B^k)$. Thus, we have $y_a^k + y_b^k = y_a^{k-1} + y_b^{k-1} + 1$. Thus, $y_a^k + y_b^k = y_a^{k-1} + y_b^{k-1} + 1 = w_{k-1}(e) + 1 = w_k(e)$.

It is easy to see that condition (13) holds. Initially $y_u^0 = 0$ for all vertices u and Step 2(e) (where certain y_u^i values increase) is the only step where the y_u^i values get updated. Thus $y_u^k \geq 0$ for all u. □

Theorem 5. *The matching M_W is a maximum weight b-matching in G.*

Proof. We prove the optimality of M_W by showing a dual feasible solution $\langle y_u^*, z_e^* \rangle_{(u \in A \cup B,\, e \in E)}$ such that this dual solution and M_W satisfy complementary slackness conditions. Let $y_u^* = y_u^W$ for all $u \in A \cup B$ and define $z_e^* = 0$ if $e \notin \Psi_W$, else $z_e^* = w(e) - y_a^* - y_b^*$ where $e = (a,b)$.

Observe that $w_W(e) = w(e)$ for all edges e. The dual feasibility of $\langle y_u^*, z_e^* \rangle_{(u \in A \cup B,\, e \in E)}$ follows from conditions (9), (10) and (13) and the definition of z_e^* values. Primal complementary slackness follows from conditions (10) and (12) along with the definition of z_e^* values for $e \in \Psi_W$. Dual complementary slackness follow from conditions (10) and (11) along with the observation that if $z_e^* > 0$ then $e \in \Psi_W$. This proves that M_W is primal optimal and $\langle y_u^*, z_e^* \rangle_{(u \in A \cup B,\, e \in E)}$ is dual optimal. □

Thus the maximum weight bipartite capacitated b-matching problem can be decomposed into W unweighted versions of the same problem, where $W = \max_{e \in E} w(e)$. The following computational result is immediate.

Theorem 6. *The maximum weight capacitated b-matching problem in $G = (A \cup B, E)$ can be solved in*
1. *$O(Wnm)$ time using Orlin's maximum flow algorithm (Orlin [36]), or*

2. $O(W\sqrt{\beta}m)$ *time, using Gabow's algorithm* (*Gabow* [14]), *where* $\beta = \sum_{v\in A\cup B} b(e)$, *in the case of simple b-matching* (*where* $c \equiv 1$), *or*
3. $O(W(n_1 m + n_1^3))$ *time, using the algorithm of Ahuja et al.* (*Ahuja et al.* [2]), *where* $n_1 = \min\{|A|, |B|\}$, *or*
4. $O(W(n_1 m + n_1^2\sqrt{m}))$ *time, using the algorithm of Ahuja et al.* (*Ahuja et al.* [2]), *or*
5. $O(W(n_1 m + n_1^2\sqrt{\log C}))$ *time, using the algorithm of Ahuja et al.* (*Ahuja et al.* [2]), *or*
6. $O(Wn_1 m \log(2 + n_1^2/m))$ *time, using the algorithm of Ahuja et al.* (*Ahuja et al.* [2]).

4. Conclusions and open problems. We considered the maximum weight matching problem in $G = (V, E)$ with integral edge weights. We solved this problem via the maximum cardinality matching algorithm—the running time of our algorithm is W times the running time of a maximum cardinality matching algorithm, where W is the largest edge weight. This running time is as good as the current fastest algorithms for the maximum weight matching problem. Our algorithm also computed an optimal dual solution that is integral, thereby showing an integral certificate to the optimality of the computed matching.

We then extended this approach to the maximum weight capacitated b-matching problem in bipartite graphs, where edge weights are in $\{1, 2, \ldots, W\}$. We showed that this problem can also be decomposed into W unweighted and capacitated versions of the same problem. An open problem is to extend this approach to the maximum weight b-matching problem in general graphs.

Acknowledgments. The authors thank the anonymous reviewers for their helpful comments. The first author is supported by the VR Unga Forskare [Grant 2015-03783]. The second author is supported by the Indo-German Max Planck Center for Computer Science grant.

References

[1] Ahuja R, Magnanti T, Orlin J (1993) Network Flows: Theory, Algorithms, and Applications (Pearson, Upper Saddle River, NJ).
[2] Ahuja R, Orlin J, Stein C, Tarjan R (1994) Improved algorithms for bipartite network flow. *SIAM J. Comput.* 23(5):906–933.
[3] Berge C (1958) Sur le couplage maximum d'un graphe. *Comptes rendus hebdomadaires des séances de l'Académie des sciences* 247:258–359.
[4] Cook WJ, Cunningham WH, Pulleyblank WR Schrijver A (1997) *Combinatorial Optimization* (John Wiley & Sons, New York).
[5] Cook WJ A note on matchings and separability. *Discrete Appl. Math.* 10(2):202–209.
[6] Cunningham W, Marsh A (1978) A primal algorithm for optimum matching. Balinski ML, Hoffman AJ, eds. *Polyhedral Combinatorics—Dedicated to the Memory of D. R. Fulkerson* (Springer, Berlin), 50–72.
[7] Cygan M, Gabow H, Sankowski P (2015) Algorithmic applications of Baur-Strassen's theorem: Shortest cycles, diameter and matchings. *J. ACM* 62(4):article 28.
[8] Derigs U (1981) A shortest augmenting path method for solving minimal perfect matching problems. *Networks* 11(4):379–390.
[9] Duan R, Pettie S (2014) Linear-time approximation for maximum weight matching. *J. ACM* 61(1):1–23.
[10] Duan R, Pettie S, Su H-H Scaling algorithms for weighted matching in general graphs. Preprint, arXiv:1411.1919v2.
[11] Dulmage A, Mendelsohn N (1958) Coverings of bipartite graphs. *Canadian J. Math.* 10:517–534.
[12] Edmonds J (1965a) Maximum matching and a polyhedron with (0, 1) vertices. *J. Res. National Bureau Standards Section* 69B:125–130.
[13] Edmonds J (1965b) Paths, trees, and flowers. *Canadian J. Math.* 17:449–467.
[14] Gabow H (1983) An efficient reduction technique for degree-constrained subgraph and bidirected network flow problems. *Proc. 15th Annual ACM Sympos. Theory Comput., STOC* (ACM, New York), 448–456.
[15] Gabow H (1985a) A scaling algorithm for weighted matching on general graphs. *Proc. 26th Annual Sympos. Foundations Comput. Sci., FOCS,* (IEEE Computer Society, Washington, DC), 90–100.
[16] Gabow H (1985b) Scaling algorithms for network problems. *J. Comput. System Sci.* 31:148–168.
[17] Gabow H (1990) Data structures for weighted matching and nearest common ancestors with linking. Johnson DS, ed. *Proc 1st Sympos. Discrete Algorithms, SODA '90* (SIAM, Philadelphia), 434–443.
[18] Gabow H, Sankowski P (2013) Algebraic algorithms for b-matching, shortest undirected paths, and f-factors. *Proc. 54th Sympos. Foundations Comput. Sci., FOCS '13* (IEEE Computer Society, Washington, DC), 137–146.
[19] Gabow H, Tarjan R (1989) Faster scaling algorithms for network problems. *SIAM J. Comput.* 18(5):1013–1036.
[20] Gabow H, Tarjan R (1991) Faster scaling algorithms for general graph-matching problems. *J. ACM* 38(4):815–853.
[21] Gallai T (1964) Maximale Systeme unabhängiger Kanten. *Magyar Tudományos Akadémia—Matematikai Kutató Intézetének Közleményei* 9:401–413.
[22] Goldberg A, Karzanov A (1995) Maximum skew-symmetric flows. Spirakis PG, ed. *Proc. 3rd Eur. Sympos. Algorithms, ESA '95* (Springer, Berlin), 155–170.
[23] Goldberg A, Tarjan R (1987) Solving minimum-cost flow problems by successive approximation. Aho AV, ed. *Proc. 9th Sympos. Theory Comput., STOC '87* (ACM, New York), 7–18.
[24] Goldberg A, Tarjan R (1990) Solving minimum-cost flow problems by successive approximation. *Math. Oper. Res.* 15(3):430–466.
[25] Hoffman A, Oppenheim R (1978) Local unimodularity in the matching polytope. *Ann. Discrete Math.* 2:201–209.
[26] Huang C-C, Kavitha T (2012) Efficient algorithms for maximum weight matchings in general graphs with small edge weights. Rabani Y, ed. *Proc. 23rd Sympos. Discrete Algorithms SODA '12* (SIAM, Philadelphia), 1400–1412.
[27] Kao M-Y, Lam TW, Sung W-K, Ting H-F (2001) A decomposition theorem for maximum weight bipartite matchings. *SIAM J. Comput.* 31(1):18–26.

[28] Kuhn HW (1955) The Hungarian method for the assignment problem. *Naval Res. Logist. Quart.* 2:83–97.
[29] Lawler EL (2011) *Combinatorial Optimization: Networks and Matroids* (Dover Publications, Mineola, NY).
[30] Le Gall F (2014) Power of tensors and fast matrix multiplication. Nabeshima K, Nagasaka K, Winkler F, Szántó, eds. *Proc. 39th Internat. Sympos. Symbolic and Algebraic Comput., ISSAC '14* (ACM, New York), 296–303.
[31] Lee YT, Sidford A Path Finding II: An $\tilde{O}(m\sqrt{(n)})$ Algorithm for the Minimum Cost Flow Problem. Preprint, arXiv: 1312.6713v2.
[32] Lovász L, Plummer M (2009) *Matching Theory* (AMS, Providence, RI).
[33] Mucha M, Sankowski P (2004) Maximum matchings via Gaussian elimination. *Prod. 45th Sympos. Foundations Comput. Sci., FOCS '04* (IEEE Computer Society, Washington, DC), 248–255.
[34] Orlin J (1988) A faster strongly polynomial minimum cost flow algorithm. Simon J, ed. *Proc. 20th Sympos. Theory Comput., STOC '88* (ACM, New York), 377–387.
[35] Orlin J (1993) Parallel algorithms for the assignment and minimum-cost flow problems. *Oper. Res.* 41(2):338–350.
[36] Orlin J (2013) Max flows in $O(nm)$ time, or better. *Proc. 45th Sympos. Theory Comput., STOC '13* (ACM, New York), Vol. 41, 765–774.
[37] Petersen J (1891) Die Theorie der regulären Graphs. *Acta Mathematica* 15:193–220.
[38] Pettie S (2012) A simple reduction from maximum weight matching to maximum cardinality matching. *Inform. Processing Lett.* 112(23):893–898.
[39] Schrijver A (1983) Short proofs on the matching polyhedron. *J. Combinatorial Theory B* 34(1):104–108.
[40] Schrijver A (1998) *Theory of Linear and Integer Programming* (John Wiley & Sons, Chichester, UK).
[41] Schrijver A (2003) *Combinatorial Optimization—Polyhedra and Efficiency* (Springer, Berlin).
[42] Tutte W (1947) The factorization of linear graphs. *J. London Math. Soc.* 22:107–111.