Efficient Algorithms for Maximum Weight Matchings in General Graphs with Small Edge Weights*

Chien-Chung Huang† Telikepalli Kavitha‡

Abstract

Let $G = (V, E)$ be a graph with positive integral edge weights. Our problem is to find a matching of maximum weight in G. We present a simple iterative algorithm for this problem that uses a maximum cardinality matching algorithm as a subroutine. Using the current fastest maximum cardinality matching algorithms, we solve the maximum weight matching problem in $O(W\sqrt{n}m\log_n(n^2/m))$ time, or in $O(Wn^\omega)$ time with high probability, where $n = |V|$, $m = |E|$, W is the largest edge weight, and $\omega < 2.376$ is the exponent of matrix multiplication. In relatively dense graphs, our algorithm performs better than all existing algorithms with $W = o(\log^{1.5} n)$. Our technique hinges on exploiting Edmonds' matching polytope and its dual.

1 Introduction

Our input is a graph $G = (V, E)$ with edge weights given by the function $w : E \to \{1, 2, \cdots W\}$. A *matching* M is a subset of E, in which no two edges 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. Let $n = |V|$ and $m = |E|$.

Matching problems lie at the core of graph theory, polyhedral combinatorics, and linear optimization. Due to their fundamental nature and vast applications, researchers have made extensive efforts in the past decades to design efficient algorithms for computing optimal matchings. In most applications, the optimality criterion is either maximum cardinality (i.e., $W = 1$) or maximum weight. We refer the readers to the book of Schrijver [33] for a comprehensive survey of the performance of various matching algorithms.

In the problem of finding a maximum weight matching in G, the fastest algorithms so far are the $O(n(m + n\log n))$ algorithm by Gabow [17], and the $O(m\log(nW)\sqrt{n\log n\cdot\alpha(m,n)})$ algorithm by Gabow and Tarjan [18], where $\alpha(m, n)$ is the inverse Ackermann function, and yet another algorithm of Gabow [16], which takes $O(W\sqrt{n}m)$ time. In fact, when the graph is bipartite, even faster algorithms are known. See Table 1 for a summary of the most efficient matching algorithms.

The most common approach in attacking the maximum weight matching problem is the primal-dual schema—often called the *Hungarian method* [32] in the special case of bipartite graphs. For general graphs, this approach was initiated by Edmonds [12] and various later algorithms, e.g., [16, 18] can be regarded as refinements of Edmonds' algorithm. The idea is to build up feasible primal and dual solutions simultaneously and show that at 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 try to successively augment it to increase its weight, until no more augmenting is possible. The work of Cunningham and Marsh [7] (and also Derigs [8]) can be regarded as the representative of this approach.

1.1 Our Contributions and Technique.

We will attack the problem from a third angle—by trying to reduce it into W different maximum cardinality matching problems. This will guarantee that we solve the problem in time proportional to the product of W and the time needed for finding a maximum cardinality matching. The same approach has been successfully applied by Kao et al. [29] to bipartite graphs. Its main advantage is that if there is any future improvement in the maximum cardinality matching algorithm, then our algorithm will automatically become faster.

Goldberg and Karzanov [30] gave an $O(\sqrt{n}m\log_n(n^2/m))$ time algorithm to find a maximum cardinality matching in general graphs. Using their algorithm as a subroutine, we show the following.

THEOREM 1.1. *A maximum weight matching in $G = (V, E)$ whose edge weights come from $\{1, \ldots, W\}$ can be computed in $O(W\sqrt{n}m\log_n(n^2/m))$ time, where $|V| = n$ and $|E| = m$.*

*Supported by IMPECS (the Indo-German Max Planck Center for Computer Science).

†Humboldt-Universität zu Berlin, Germany. villars@informatik.hu-berlin.de

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

Maximum cardinality matching in bipartite graphs		*Maximum cardinality matching in general graphs*	
$O(\sqrt{n}m)$	Hopcroft and Karp (1971) [28], Karzanov (1973) [30]	$O(\sqrt{n}m)$	Blum (1990) [4], Micali and Vazirani (1980) [36, 43] Gabow and Tarjan (1991) [18]
$O(\sqrt{n}m\log_n(n^2/m))$	Feder and Motwani (1991) [14] Goldberg and Kennedy (1997) [26]	$O(\sqrt{n}m\log_n(n^2/m))$	Goldberg and Karzanov (1995) [25]
$O(n^\omega)$	Mucha and Sankowski (2004) [37], Harvey (2006) [27]	$O(n^\omega)$	Mucha and Sankowski (2004) [37], Harvey (2006) [27]
Maximum weight matching in bipartite graphs		*Maximum weight matching in general graphs*	
$O(W\sqrt{n}m)$	Gabow (1983) [16], Kao, Lam, Sung and Ting (1999) [29]	$O(W\sqrt{n}m)$	Gabow (1983) [16]
$O(W\sqrt{n}m\log_n(n^2/m))$	Kao, Lam, Sung and Ting (2001) [29]	$O(W\sqrt{n}m\log_n(n^2/m))$	This work
$O(\sqrt{n}m\log(nW))$	Gabow and Tarjan (1988,1989) [20, 21] Goldberg (1993) [24]	$O(m\log(nW)\sqrt{n\log n\cdot\alpha(m,n)})$	Gabow and Tarjan (1991) [18]
$O(\sqrt{n}m\log W)$	Duan and Su (2012) [10]		
$\tilde{O}(Wn^\omega)$	Sankowski (2006) [40]	$O(Wn^\omega)$	This work
$O(nSP_+(n,m,W))$	Edmonds and Karp (1970) [13], Tomizawa (1971) [41]	$O(n(m+n\log n))$	Gabow (1990) [17]
$O(n^{2.5}\log(nW)(\frac{\log\log n}{\log n})^{1/4})$	Cheriyan and Mehlhorn (1996) [5]		

Table 1: The most efficient matching algorithms. $SP_+(n,m,W)$ means the time needed to find a shortest path in a directed graph with n vertices, m edges, and nonnegative integral edge weights where W is the maximum weight.

When the graph is dense, i.e., $m = \Theta(n^2)$, our algorithm performs better than all existing algorithms with $W = o(\log^{2.5} n)$. Mucha and Sankowski [37] (and also Harvey [27]) showed that a maximum cardinality matching in a general graph can be found in $O(n^\omega)$ time, where $\omega < 2.376$ is the exponent of matrix multiplication, with high probability. Using such an algorithm as a subroutine, we get the following result.

THEOREM 1.2. *Given a graph $G = (V, E)$ whose edge weights come from $\{1, \ldots, W\}$, a maximum weight matching in G can be computed with high probability in $O(Wn^\omega)$ time, where $|V| = n$ and $\omega < 2.376$.*

When the graph is relatively dense, i.e., $m = \Omega(n^{1.876})$, our algorithm is the most efficient so far with $W = o(\log^{1.5} n)$. It also slightly improves on Sankowski's algorithm in bipartite graphs [40], which takes $\tilde{O}(Wn^\omega)$ time[1] for the maximum weight bipartite matching problem.

We now highlight the main ideas of our technique. Our algorithm proceeds in iterations. In the i-th iteration, we consider the subgraph (call it H_i) consisting of edges with weights $W, W-1, \cdots, W-i+1$ in G. Our goal now is to construct a maximum weight matching in H_i and we do it by computing a *special* maximum cardinality matching in an unweighted graph G_i. Furthermore, this maximum cardinality matching in G_i can be used to define a dual solution, which is used to define the unweighted graph G_{i+1} for the next iteration.

We analyze how the primal and dual solutions are related to each other using Edmonds' matching polytope and its dual. A surprising aspect about our technique is that the dual solution we use is *not* the optimal dual solution. The benefit of using our dual solution is that it can be computed very efficiently, using Gallai-Edmonds decomposition theorem [11, 22]. We show that at the end of the W iterations, the primal solution and the dual solution satisfy complementary slackness conditions, hence by the duality theorem, the primal solution is optimal.

1.2 Related Work. Matching theory is a well-studied subject and is elaborately treated in many textbooks [1, 33, 34].

In general, matching problems in general graphs are usually much more difficult to deal with than the corresponding problems in bipartite graphs. For instance, a polynomial time algorithm for finding a maximum cardinality matching in bipartite graphs can be traced back to König [31] in 1931. For general graphs, the same problem was not known to be in P until the seminal paper of Edmonds [11] published in 1965, in which he introduced the famous blossom algorithm. The blossom algorithm will be used as a subroutine in our algorithm, so we will briefly review it in Section 2 (though in implementation, we will instead use other faster algo-

[1] The $\tilde{O}$ notation hides a factor of $\log^k n$ for some constant k.

rithms [27, 30, 36, 37] as our subroutine.) In 1965 Edmonds also solved the maximum weight matching problem in general graphs [12]. Additionally, his algorithm yields a characterization of the matching polytope, and is widely regarded as a major breakthrough in polyhedral combinatorics.

Along with bipartite graphs, there have also been several attempts at solving matching problems in more restricted classes of graphs, for instance, in regular bipartite graphs [23], and in planar graphs [38]. For bipartite graphs, Duan and Su [10] have shown in this conference (SODA 2012) that it is possible to find a maximum weight matching in time proportional to the product of $\log W$ and the time needed for finding a maximum cardinality matching.

A related line of research in weighted matchings is to look for approximate maximum weight matchings. For certain applications [2, 35], it may be preferable to sacrifice optimality of the solution in exchange for a faster algorithm. A recent breakthrough was done by Duan and Pettie [9], who gave a near linear time algorithm to compute a $(1-\epsilon)$-approximate solution, for any fixed $\epsilon > 0$.

2 Our Method and Preliminaries

In this section we outline our algorithm and review the following classical concepts on which our algorithm is based: the matching polytope, Edmonds' blossom algorithm, and Gallai-Edmonds decomposition.

Matching Polytope. Let Ω be the set of all odd sized subsets of V of size at least 3, also referred to as *odd sets* of vertices. For each such set $B \in \Omega$, let $E(B)$ be the set of edges *spanned* by B, i.e., $e = (u, v) \in E(B)$ if $u, v \in B$. Edmonds [12] showed how to describe the matching polytope in general graphs. The matching polytope and its dual are given below.

$$\begin{aligned} \max \sum_{e\in E} w_e x_e & \\ \sum_{e\in\delta(v)} x_e \le 1 & \qquad \forall v \in V \\ \sum_{e\in E(B)} x_e \le \frac{|B|-1}{2} & \qquad \forall B \in \Omega \\ x_e \ge 0 & \qquad \forall e \in E. \end{aligned}$$

$$\begin{aligned} \min \sum_{v\in V} y_v + \sum_{B\in\Omega} z_B \frac{|B|-1}{2} & \\ y_u + y_v + \sum_{B: e\in E(B)} z_B \ge w_e & \qquad \forall e = (u, v) \in E \\ y_v \ge 0 & \qquad \forall v \in V \\ z_B \ge 0 & \qquad \forall B \in \Omega \end{aligned}$$

The dual states that we need to cover each edge e by assigning values to its endpoints and the odd sets that span it. Let M be a matching and $\{y_v\}_{\forall v\in V} \cup \{z_B\}_{\forall B\in\Omega}$ be a dual feasible solution in the graph G. Then by the duality theorem, the two solutions are optimal if and only if *complementary slackness* conditions are satisfied.

We state the complementary slackness conditions below. Our maximum weight matching algorithm runs for W iterations: in the i-th iteration, the algorithm computes a maximum weight matching T_i in a graph $H_i = (V, F_i)$, where $F_i = E_W \cup \cdots \cup E_{W+1-i}$. The edge weights in H_i are given by a function $w'_i : F_i \to \{1, \ldots, i+1\}$ and the final graph H_W will be the same as G and have the same weight functions, i.e., $w'_W = w$, thus T_W is the matching that we seek. The proof that T_i is a maximum weight matching in H_i will be via setting the dual variables y^i_u for all $u \in V$ and z^i_B for all $B \in \Omega$ to feasible values so that complementary slackness conditions are obeyed.

The dual feasibility ((2.1) and (2.5)) and complementary slackness conditions ((2.2)-(2.4)) for the i-th iteration are listed below. These conditions will be invariants in our algorithm and we will refer to them as *invariants (2.1)-(2.5).*

$$y^i_u + y^i_v + \sum_{B: e\in E(B)} z^i_B \;\ge\; w'_i(e) \tag{2.1}$$
$$\forall \text{edges } e = (u, v) \text{ in the graph } H_i$$
$$y^i_u + y^i_v + \sum_{B: e\in E(B)} z^i_B \;=\; w'_i(e) \tag{2.2}$$
$$\forall \text{edges } e = (u, v) \in T_i$$
$$y^i_u > 0 \Rightarrow u \text{ is matched in } T_i \qquad \forall\, u \in V \tag{2.3}$$
$$z^i_B > 0 \Rightarrow \frac{|B|-1}{2} \text{ edges of } E(B) \text{ are in } T_i \qquad \forall\, B \in \Omega \tag{2.4}$$
$$y^i_u \ge 0 \text{ and } z^i_B \ge 0 \quad \forall\, u \in V \text{ and } B \in \Omega. \tag{2.5}$$

In the i-th iteration, the main step of our algorithm is the following: we have a matching T_{i-1} which is a maximum weight matching in H_{i-1}, and another matching M_{i-1}, a *maximum cardinality* matching in the unweighted graph G_{i-1}, which is our *working graph* in the $(i-1)$-th iteration. Using Gallai-Edmonds decomposition (described below) on M_{i-1}, roughly speaking, a certain subset M'_{i-1} of T_{i-1} will be identified; M'_{i-1} will belong to the working graph G_i of the i-th iteration. In the i-th iteration, we compute the maximum cardinality

matching M_i by *augmenting* M'_{i-1} in G_i. The matching T_i will be obtained from M_i and N_{i-1}, where N_{i-1} is a subset of M_{i-1} based on Gallai-Edmonds decomposition. The last step is to set suitable values to y^i_u, z^i_B for all $u \in V$ and $B \in \Omega$ so that these values and T_i satisfy invariants (2.1)-(2.5).

Thus a maximum cardinality matching algorithm is a vital subroutine in our algorithm and we overview the blossom algorithm and Gallai-Edmonds decomposition now. We highlight the main features of the blossom algorithm. More details can be found in [33, 34]. Petersen [39] observed in 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 2.1. *Let $G = (V, E)$ be the original graph. Let $G_1 = (V_1, E_1)$ be the* current *graph, where $V_1 \subseteq V \cup \Omega$, a subset of the vertices and the odd sets in G. An edge $(a, b) \in E_1$, if (1) $(u, v) \in E$, (2) $u = a$, or $u \in a \in V_1 \cap \Omega$, and (3) $v = b$, or $v \in b \in V_1 \cap \Omega$.*

Suppose that M_1 is a matching in the current graph G_1. A set of vertices $B = (a_1, a_2, \cdots, a_{2t+1})$ is a blossom *in G_1 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 $1 \le i \le 2t+1$, and (2) $M_1(a_{2i}) = a_{2i+1}$, for $1 \le i \le t$. The first vertex a_1 is called the* base *of the blossom B. (Note that a_1 can be matched to some vertex in $V_1 \setminus B$, or it can be left unmatched). The circuit traversing the vertices of B is called the* defining circuit *of B.*

Closing a blossom. *Let G_1 be the current graph and M_1 a matching in it. Suppose $B = (a_1, a_2, \cdots, a_{2t+1})$ is a blossom in G_1. Then closing the blossom B means that we form a new current graph $G_2 = (V_2, E_2)$, where $V_2 = (V_1 \backslash \{a_i\}_{i=1}^{2t+1}) \cup \{B\}$, and create a new matching M_2 in G_2 as follows:*

$M_2(a) = M_1(a)$ *if* $a \notin B \cup \{M_1(a_1)\}; M_2(B) = M_1(a_1)$.

Note that a blossom must be an odd set in Ω, but not vice versa. Moreover, multiple blossoms may be present in the current graph; which blossoms get closed and which do not depends on the search strategy of Edmonds' blossom algorithm. In our discussion, an odd set $B \in \Omega$ is called a blossom only if it is closed by Edmonds' algorithm.

Below we give a more generalized definition of opening a blossom; the reason for such a generalization will be explained immediately.

DEFINITION 2.2. ***Opening a blossom.*** *Let $G_1 = (V_1, E_1)$ be the current graph and M_1 a matching in it. Suppose further that G_1 is derived from closing a sequence of blossoms in a subgraph $G' = (V, E')$ of the original graph $G = (V, E)$. Assume $B \in V_1 \cap \Omega$ is one of these blossoms and $B = (a_1, a_2, \cdots, a_{2t+1})$. Then opening the blossom B means that we form a new current graph $G_2 = (V_2, E_2)$, where $V_2 = (V_1 \backslash \{B\}) \cup \{a_i\}_{i=1}^{2t+1}$, and create a new matching M_2 in G_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_1 is unmatched in M_2, and $M_2(a_{2i}) = a_{2i+1}, \forall 1 \le i \le t$.*
- *If $M_1(B) = a'$, $a_x \in B$ and $(a_x, a') \in E_2$, then for all $1 \le i \le t$, we have $M_2(v_{(x+2i-1) \bmod (2t+1)}) = v_{(x+2i) \bmod (2t+1)}$ and $M_2(a_x) = a'$.*

Observe that the above definition allows the following: suppose the previous graph is G' and let G_1 be obtained by closing a sequence of blossoms in G'. If we add more edges to G' to form an updated graph G, then we still have a well-defined operation of opening a blossom in the current graph G_1.

Note that our algorithm needs such a definition: in each iteration, we add new edges into the *working graph* (which is unweighted–see the next section for details). At the end of i-th iteration, we may need to open a subset of blossoms which are formed in the previous iterations. The above definition thus guarantees that we can open the blossoms properly.

We are now ready to describe how Edmonds' blossom algorithm works. In each round, it seeks to find an augmenting path by building a Hungarian forest. In building such a 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. Note that a blossom B, once closed, can be a part of another blossom B' in the updated graph. In this case, B is said to be *embedded* in B'. A blossom not embedded in any other blossom is an *outermost* blossom. After the closing of blossoms, if an augmenting path is found, then the matching is augmented along it. Furthermore, all blossoms are re-opened (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, $\tilde{\mathcal{F}}$ the final constructed Hungarian forest. Some of the vertices in $\tilde{G}$ can indeed be blossoms that are closed. 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 $\tilde{G}$;

moreover, by Tutte-Berge formula [3, 42], if we re-open all blossoms, then we have a maximum matching M in the original graph G. Furthermore, the Hungarian forest $\tilde{\mathcal{F}}$ takes the following form, which encodes the Gallai-Edmonds decomposition. (Note that there may exist edges in $\tilde{E}$ that are not present in $\tilde{\mathcal{F}}$).

– A set of trees, whose roots are left unmatched in $\tilde{M}$ in $\tilde{G}$. 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 (resp., even) length starting from the root to this node.
– A set of matched edges. The endpoint nodes of these matched edges are *unreachable*.

PROPOSITION 2.1. *Let $\tilde{\mathcal{F}}$ be the Hungarian forest found at the end of Edmond's blossom algorithm.*

(i) If a node in $\tilde{G}$ is a blossom, then it must be even and it must be an outermost blossom.

(ii) An even node has no edge in $\tilde{E}$ connecting it to another even or unreachable node.

By this proposition, all odd and unreachable nodes are real vertices. Furthermore, by the description of the forest, an odd vertex is matched to an even node, while an unreachable vertex is matched to another unreachable vertex in $\tilde{M}$. Now we present the Gallai-Edmonds decomposition theorem [11, 22].

PROPOSITION 2.2. *Let $\tilde{\mathcal{F}}$ be the Hungarian forest at the termination of Edmonds' blossom algorithm. Let $\mathcal{U}$ and $\mathcal{O}$ be the sets of unreachable and odd vertices, and $\mathcal{E}$ the set of even nodes in $\tilde{\mathcal{F}}$. Then the following hold.*

(i) All vertices in $\mathcal{O} \cup \mathcal{U}$ are matched in all maximum cardinality matchings in G; in all such matchings, all vertices in $\mathcal{O}$ are matched to vertices in $\mathcal{E}$ or to vertices contained in blossoms in $\mathcal{E}$, while all vertices in $\mathcal{U}$ are matched to each other.

(ii) None of the edges in $\mathcal{O} \times (\mathcal{O} \cup \mathcal{U})$ is part of any maximum cardinality matching.

(iii) For each blossom $B \in \Omega$, if B is present in $\tilde{G}$, then for all maximum cardinality matchings M, exactly $\frac{|B|-1}{2}$ edges of $E(B)$ are a part of M.

Proof. For each vertex $v \in V$, if $v \in \mathcal{O}$, set $y_v = 1$; if $v \in \mathcal{U}$, set $y_v = 1/2$; and if $v \in \mathcal{E}$, or $v \in B \in \Omega$, and B a blossom in $\mathcal{E}$, set $y_v = 0$. Furthermore, if $B \in \Omega$ is a blossom in $\mathcal{E}$, let $z_B = 1$, otherwise $z_B = 0$. Recall that there is no edge between an even node and an even/unreachable node. So it can be verified that $\{y_v\}_{v \in V} \cup \{z_B\}_{B \in \Omega}$ is a dual feasible solution in the original graph G. Furthermore, let M be a maximum cardinality matching found at the end of Edmonds' blossom algorithm. Then by complementary slackness conditions, the edge incidence vector $(x_e)_{e \in E}$ of M and $\{y_v\}_{v \in V} \cup \{z_B\}_{B \in \Omega}$ are optimal dual pairs. Moreover all of (i)—(iii) would follow from complementary slackness conditions.

Once the Gallai-Edmonds decomposition is found, we can then use it to define an optimal dual solution easily as we did in the above proof. Although this particular dual is only half-integral (it is known that Edmond's matching polytope is *totally dual integral* [7], hence there is always an optimal integral dual), our algorithm will make use of a $\{0, 1\}$-dual solution that is modified from this half-integral dual solution.

3 Our algorithm

Recall that our input is $G = (V, E)$. The edge set $E = E_W \dot\cup E_{W-1} \dot\cup \cdots \dot\cup E_1$, where E_i is the set of edges in G of weight i. We will be using the following notation throughout this section: for any graph G_i on vertex set $V_i \subset V \cup \Omega$, we use $\tilde{G}_i$ to denote the updated graph *after* running the blossom algorithm on G_i, where blossoms discovered during the algorithm get closed. The vertex set of $\tilde{G}_i$ is $\tilde{V}_i \subset V \cup \Omega$. By Gallai-Edmonds decomposition, $\tilde{V}_i$ can be partitioned into $\mathcal{O}_i \dot\cup \mathcal{U}_i \dot\cup \mathcal{E}_i$.

Let M_i be the maximum cardinality matching computed by the blossom algorithm in G_i. We use $\tilde{M}_i$ to denote M_i after closing all blossoms that we discovered during the course of the blossom algorithm so that $\tilde{M}_i$ is a maximum cardinality matching in $\tilde{G}_i$. By the discussion after Proposition 2.1, we know that $\tilde{M}_i \subseteq (\mathcal{O}_i \times \mathcal{E}_i) \cup (\mathcal{U}_i \times \mathcal{U}_i)$.

Let $X \subseteq V \cup \Omega$ be a subset of vertices and odd sets. We say x is an *element* of X if either $x \in X \cap V$ or $x \in B \in X \cap \Omega$. Suppose that $(u, v) \in E$ and u is an element of $X \subseteq V \cup \Omega$ and v an element of $Y \subseteq V \cup \Omega$. We abuse notation by writing $(u, v) \in X \times Y$. Similarly, if $B, B' \in \Omega$ are blossoms and there is an edge (B, B') in G_i (or $\tilde{G}_i$) because B contains $u \in V$ and B' contains $v \in V$ so that $(u, v) \in E$, we often write (u, v) to denote the edge (B, B') whenever no confusion arises. Finally, for convenience, let $y_u^0 = 0$ for all $u \in V$ and $z_B^0 = 0$ for all $B \in \Omega$.

The algorithm. We are now ready to describe our algorithm. The first iteration is described below.

- Let S_1 be the graph (V, F_1), where $F_1 = E_W$ and all edges have weight 1. That is, $w_1(e) = 1$ for all $e \in E_W$.

- Let $G_1 = S_1$ and $V_1 = V$. Compute a maximum cardinality matching M_1 in G_1. The blossom algorithm also yields the graph $\tilde{G}_1$ with vertex set $\tilde{V}_1$ and a partition of $\tilde{V}_1 = \mathcal{O}_1 \dot\cup \mathcal{U}_1 \dot\cup \mathcal{E}_1$. The matching $\tilde{M}_1$ is of maximum cardinality in $\tilde{G}_1$.
- The dual variables are set as follows:
  - Set $y_u^1 = y_u^0 + 1 = 1$ for all $u \in \mathcal{O}_1 \cup \mathcal{U}_1$, else set $y_u^1 = y_u^0 = 0$.
  - For every blossom $B \in \mathcal{E}_1$ (B must be outermost), set $z_B^1 = z_B^0 + 1 = 1$ (Proposition 2.1 states that there are no blossoms in $\mathcal{O}_1 \dot\cup \mathcal{U}_1$); for all other odd sets $B \in \Omega$, set $z_B^1 = z_B^0 = 0$.

Let H_1 be a graph on vertex set V and edge set $F_1 = E_W$. Let $w_1' : E_W \to \{1, 2\}$ be the edge weight function in H_1 defined as follows. For any $e \in E_W$: set $w_1'(e) = 2$ if $e \in \mathcal{U}_1 \times \mathcal{U}_1$, else set $w_1'(e) = 1$. Let $\tilde{T}_1 = \tilde{M}_1$ and let T_1 be the matching obtained by opening all blossoms in $\tilde{T}_1$. (In this particular iteration $T_1 = M_1$. But in iterations $k \geq 2$, T_k can be different from M_k.)

Recall from Section 2 that in every iteration we show via invariants (2.1)-(2.5) that T_i is a maximum weight matching in the graph H_i. Lemma 3.1 shows that these invariants hold at the end of the first iteration. Thus T_1 will be a maximum weight matching in the graph H_1.

LEMMA 3.1. *Invariants (2.1)-(2.5) hold at the end of the first iteration.*

Proof. Invariant (2.5) is trivial. For invariant (2.1), there are five cases on any edge $e = (u, v) \in E_W$: *(i)* e is spanned by an outermost blossom in $B \in \mathcal{E}_1$, *(ii)* $e \in \mathcal{O}_1 \times \mathcal{E}_1$, *(iii)* $e \in \mathcal{O}_1 \times \mathcal{O}_1$, *(iv)* $e \in \mathcal{O}_1 \times \mathcal{U}_1$, and *(v)* $e \in \mathcal{U}_1 \times \mathcal{U}_1$. Note that due to Proposition 2.1, there is no edge $e \in \mathcal{E}_1 \times (\mathcal{E}_1 \cup \mathcal{U}_1)$ unless e is spanned by a blossom in $\mathcal{E}_1$ (which is case (i)).

In cases (i)-(iv), $w_1'(e) = 1$. The left hand side of invariant (2.1) will be at least 1 since at least one of y_u^1, y_v^1, z_B^1 will be set to 1. In case (v), $w_1'(e) = 2$, however *both* y_u^1 and y_v^1 are set to 1 here. So invariant (2.1) is satisfied.

For invariant (2.2), recall that T_1 is obtained by opening all blossoms in $\tilde{T}_1 = \tilde{M}_1$. Let $e = (u, v) \in T_1$. If e is spanned by an outermost blossom $B \in \mathcal{E}_1$, then $w_1'(e) = 1$, $y_u^1 = y_v^1 = 0$ and $z_B^1 = 1$ and all other odd sets B' spanning e have $z_{B'}^1 = 0$. If u is an element of $\mathcal{E}_1$ and v is an element of $\mathcal{O}_1$, then $w_1'(e) = 1$, $y_u^1 = 0$, $y_v^1 = 1$, and all odd sets B spanning e have $z_B^1 = 0$. Finally, if both u and v are elements of $\mathcal{U}_1$, then $w_1'(e) = 2$, $y_u^1 = y_v^1 = 1$, and all odd sets B spanning e have $z_B^1 = 0$. In all cases, invariant (2.2) holds.

We now show invariants (2.3) and (2.4). If $y_u^1 > 0$ then $u \in \mathcal{O}_1 \cup \mathcal{U}_1$. The matching $\tilde{M}_1$ matches all vertices in $\mathcal{O}_1 \cup \mathcal{U}_1$ by Proposition 2.2. Invariants (2.3) and (2.4) hold because T_1 is obtained by opening all blossoms in $\tilde{T}_1 = \tilde{M}_1$ (see Definition 2.2). Thus all the invariants hold at the end of the first iteration.

The k-th iteration k, for $k \geq 2$. We now describe what happens in iteration k, where $k \geq 2$. During the k-th iteration, the graphs that we will be dealing with are the *starting* graph of the k-th iteration, denoted by S_k, and the *working* graph of the k-th iteration, denoted by G_k. The starting graph S_k has V as its vertex set while the working graph G_k has both blossoms formed in previous iterations and real vertices in V in its vertex set. The basic idea is that we use a matching found at the end of the previous iteration, $k-1$, as an initial matching in the working graph G_k. We then augment it till we have a maximum cardinality matching in G_k. This matching can then be used to define a matching T_k in graph H_k, in which we will maintain invariants (2.1)-(2.5). The optimality of T_k in H_k would follow because of the complementary slackness condition. We now explain the details.

Starting graph. $S_k = (V, F_k)$ has $F_k = F_{k-1} \cup E_{W+1-k}$ as its edge set. F_{k-1} is the edge set of the starting graph S_{k-1} of the previous iteration. The edge weight function $w_k : F_k \to \{1, \ldots, k\}$ in S_k is defined as: for $e \in E_i$, we have $w_k(e) = k + i - W$, where $W + 1 - k \leq i \leq W$. By this definition of w_k, we have: (if $e \in E_{W+1-k}$, then assume $w_{k-1}(e) = 0$)

$$w_k(e) = w_{k-1}(e) + 1 \ \forall e \in F_k.$$

Note that the graph H_k in which we will maintain invariants (2.1)-(2.5) differs from S_k only in its edge weight function. The former has the edge weight function w_k' while the latter has w_k. We will define w_k' precisely later and we will maintain the invariant that

$$w_k(e) \leq w_k'(e) \leq w_k(e) + 1 \ \ \forall e \in F_k \quad \text{in all iterations } k, \text{ where } 1 \leq k \leq W. \tag{3.6}$$

Note that the above invariant holds for $k = 1$ since $w_1(e) = 1$ and $w_1'(e)$ is either 1 or 2 for all $e \in F_1 = E_W$.

Working graph. Let $\mathbb{V}_k$ be the union of the outermost blossoms B formed in previous iterations $1, \ldots, k-1$ with a *positive* value of the corresponding dual variable z_B^{k-1}, and those vertices of V that are outside all these blossoms. Note that by this definition $\mathbb{V}_k$ is a partition over V.

During the $(k-1)$-th iteration, we would have found the Gallai-Edmonds decomposition $\mathcal{E}_{k-1}$, $\mathcal{O}_{k-1}$, and $\mathcal{U}_{k-1}$ of the vertex set in $\tilde{G}_{k-1}$ (the graph $\tilde{G}_k$ will

be described shortly). We will maintain the following invariant:

(3.7) *In any iteration k, if B is a blossom in $\mathcal{U}_k$, then B is outermost and has $z_B^k > 0$.*

Note that the above invariant holds vacuously for $k = 1$ since there are *no* blossoms in $\mathcal{U}_1$. Invariant (3.7) guarantees that each blossom in $\mathcal{U}_{k-1}$ belongs to $\mathbb{V}_k$. Now the vertex set of G_k is defined as $V_k = \mathbb{V}_k \backslash \mathcal{U}_{k-1}$. By this definition, all the vertices/blossoms of $\mathcal{U}_{k-1}$ are absent from V_k, so all edges (u, v) where at least one of u, v is an *element* of $\mathcal{U}_{k-1}$ are obviously missing in G_k. The *relevance* of any remaining edge $e = (u, v)$ is determined using the function:

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

The edge set of G_k consists of those edges e with $\delta_k(e) > 0$.

Let the function δ_k defined above be the edge weight function of G_k. We now claim that all edges in G_k have weight 1. This is because by invariants (2.1) and (2.5) of the previous iteration, we have $y_u^{k-1} + y_v^{k-1} + \sum_{B:(u,v)\in B} z_B^{k-1} \geq w'_{k-1}(e)$ for all edges e in F_k (in case $e \notin F_{k-1}$, let $w'_{k-1}(e) = 0$) and it follows from (3.6) that $w_k(e) = w_{k-1}(e) + 1 \leq w'_{k-1}(e) + 1$. So $\delta_k(e) \leq 1$ for all $e \in F_k$. Since no edge e with $\delta_k(e) \leq 0$ is present in G_k, every edge e in G_k satisfies $\delta_k(e) = 1$. In other words, G_k is an *unweighted* graph.

Claim 1. *For every outermost blossom B in V_k with $z_B^{k-1} > 0$, and for every blossom B^* embedded in B, every edge e in the defining circuits of B and of B^* has $\delta_k(e) = 1$.*

(For the sake of the continuity of our discussion, the proofs of Claim 1 and Claim 2 (stated later) are given at the end of this discussion.)

Claim 1 shows that for any blossom $B \in V_k$, the edges in the defining circuits of B and its embedded blossoms B^* also pass the $\delta_k(\cdot)$ test. That is why B retains its identity as a blossom in G_k. We refer to all the blossoms in V_k and their embedded blossoms as *old* blossoms.

The initial matching M'_{k-1} in G_k and the final matching T_k in H_k. We will compute a maximum cardinality matching M_k in the working graph G_k. In particular, this matching has to be obtained from augmenting a specific initial matching M'_{k-1}, which is derived from the "shrunk" matching $\tilde{T}_{k-1}$ of the previous iteration.

After the maximum cardinality matching M_k in G_k is found in this iteration, we can use it to define another shrunk matching $\tilde{T}_k$. The matching T_k in H_k is then obtained by opening up all blossoms in $\tilde{T}_k$. We now explain the details. We first describe $\tilde{T}_k$ and then M'_{k-1}. The matching $\tilde{T}_k$ is made up of 2 parts: $\tilde{T}_k = \tilde{M}_k \dot{\cup} \tilde{N}_{k-1}$. We define $\tilde{M}_k$ and $\tilde{N}_{k-1}$ below.

- Run the blossom algorithm on G_k to find a maximum cardinality matching M_k in G_k. In particular, M_k has to be *augmented* from M'_{k-1}. After the blossom algorithm is run, we have the graph $\tilde{G}_k$ and $\tilde{M}_k$ is a maximum cardinality matching in it. Recall that G_k has all *old* blossoms closed and $\tilde{G}_k$ is G_k after also closing all the *new* blossoms discovered while running the blossom algorithm. Also, $\tilde{G}_k$ has vertex set $\tilde{V}_k$, which can be partitioned into $\mathcal{E}_k \dot{\cup} \mathcal{O}_k \dot{\cup} \mathcal{U}_k$. Clearly a blossom in $\mathcal{E}_k \dot{\cup} \mathcal{O}_k \dot{\cup} \mathcal{U}_k$, no matter old or new, is an outermost blossom.

- The matching $\tilde{N}_{k-1}$ will be a subset of $\tilde{T}_{k-1}$ from the previous iteration. Precisely, let $\tilde{M}_{k-1} \subseteq \tilde{T}_{k-1}$ be the maximum cardinality matching computed in $\tilde{G}_{k-1}$ in the previous iteration. We can partition $\tilde{M}_{k-1}$ into edges from $\mathcal{U}_{k-1} \times \mathcal{U}_{k-1}$ and edges from $\mathcal{O}_{k-1} \times \mathcal{E}_{k-1}$, where $\tilde{V}_{k-1}$ (the vertex set of $\tilde{G}_{k-1}$) is partitioned into $\mathcal{E}_{k-1} \dot{\cup} \mathcal{O}_{k-1} \dot{\cup} \mathcal{U}_{k-1}$. Let $\tilde{N}_{k-1} \subseteq \tilde{M}_{k-1}$ denote the set of edges in $\tilde{M}_{k-1}$ from $\mathcal{U}_{k-1} \times \mathcal{U}_{k-1}$.

By the above discussion, $\tilde{T}_k = \tilde{M}_k \dot{\cup} \tilde{N}_{k-1}$ is a matching on the vertex set $\tilde{V}_k \dot{\cup} \mathcal{U}_{k-1}$, where $\tilde{M}_k$ is in $\tilde{G}_k$ while $\tilde{N}_{k-1}$ is in $\tilde{G}_{k-1}|_{\mathcal{U}_{k-1}}$, the subgraph of $\tilde{G}_{k-1}$ restricted to $\mathcal{U}_{k-1}$. Now let T_k denote the matching after we open all blossoms in $\tilde{T}_k$. However note that our algorithm does not specifically open all blossoms in $\tilde{T}_k$ to construct T_k in any intermediate iteration k. We retain $\tilde{T}_k$ as it is. It is only in the final iteration, i.e., when $k = W$, do we explicitly open out all blossoms in $\tilde{T}_W$ to obtain T_W.

We now explain how M'_{k-1} is defined. In the $(k-1)$-th iteration, we may decrease the dual variables of certain blossoms (the details will be shown soon). If an outermost blossom B has $z_B^{k-1} = 0$, then we need to open it. And this has to be done recursively.

Claim 2. *Given the matching $\tilde{T}_{k-1} \setminus \tilde{N}_{k-1}$ in $\tilde{G}_{k-1} \cup \tilde{G}_{k-2}|_{\mathcal{U}_{k-2}}$ (if $k = 2$, then $\tilde{G}_{k-2}|_{\mathcal{U}_{k-2}} = \emptyset$), we open out outermost blossoms B whose dual variables $z_B^{k-1} = 0$ and if their embedded blossoms $B^* \subset B$ become outermost and $z_{B^*}^{k-1} = 0$, then we open them as well and this is done recursively. Let M'_{k-1} be the resulting matching. Then, M'_{k-1} is a matching on the vertex set V_k and every edge of M'_{k-1} is contained in G_k.*

Updating the dual variables. The values of the dual variables are updated after the Gallai-Edmonds decomposition $\tilde{V}_k = \mathcal{E}_k \dot\cup \mathcal{O}_k \dot\cup \mathcal{U}_k$ is found.

- For every $u \in V$: if u is an *element* of $\mathcal{O}_k \cup \mathcal{U}_k$, then set $y_u^k = y_u^{k-1} + 1$; else set $y_u^k = y_u^{k-1}$.
- For all odd sets $B \in \Omega$: if B is a blossom in $\mathcal{E}_k$, set $z_B^k = z_B^{k-1} + 1$; if B is a blossom in $\mathcal{O}_k$, then set $z_B^k = z_B^{k-1} - 1$; else set $z_B^k = z_B^{k-1}$. (Note that if B is a blossom *embedded* in another blossom in $\mathcal{E}_k \dot\cup \mathcal{O}_k$, $z_B^k = z_B^{k-1}$.)

When $k = W$, there is one change in the setting of the dual variables: for every $u \in V$, if u is an element of $\mathcal{U}_k$, then set $y_u^k = y_u^{k-1} + 1/2$ (note that for $k < W$, we had set $y_u^k = y_u^{k-1} + 1$ if u is an element of $\mathcal{U}_k$).

Observe that if a blossom $B \in \mathcal{U}_k$, since all *new* blossoms are (embedded in) outermost blossoms in $\mathcal{E}_k$ (by Proposition 2.1), B is an *old* blossom and $B \in V_k$. Then from the definition of V_k, we have $z_B^{k-1} > 0$ and (3.7) holds from the way z_B^k is defined. Note also that if a blossom $B \in \mathcal{O}_k$, it may happen that z_B^k is down to 0. Then we need to open such a blossom (and possibly some of its embedded blossoms) when we prepare the next initial matching M'_k for the next iteration. (See Claim 2).

Proof of Claim 1. We show this by induction. The base case is $k = 2$. Then both B and B^* were formed while running the blossom algorithm in G_1. Since B is in V_2, it follows from Proposition 2.1 that B has to be in $\mathcal{E}_1$ and since only edges from E_W are present in G_1, all the edges in the circuits defining B and B^* have to be in E_W. Let $e = (u, v)$ be any edge in the defining circuits of B or B^*. Then $y_u^1 = y_v^1 = 0$ while $z_B^1 = 1$ (note that $B^* \subset B$) and for any other odd set B' spanning e, we have $z_{B'}^1 = 0$. Hence it follows that $\delta_2(e) = w_2(e) - 1 = 2 - 1 = 1$. This finishes the base case.

For the induction step $k \geq 3$, there are three possibilities. (i) B is a blossom in $\mathcal{E}_{k-1}$, or (ii) B is a blossom in $\mathcal{O}_{k-1}$ or B is embedded in another outermost blossom $B' \in \mathcal{O}_{k-1}$ (this happens because it is possible that $B' \in \mathcal{O}_{k-1}$ may have $z_{B'}^{k-1} = 0$. After B' is opened out, B becomes an outermost blossom with $z_B^{k-1} > 0$. Note that in this case, $B' \in V_{k-1}$ because of Proposition 2.1), or (iii) B is a blossom in $\mathcal{U}_{k-2}$.

Let $e = (u, v)$ be any edge in the defining circuits of B or B^*. In case (i), either e is a part of the working graph G_{k-1}, or a part of the defining circuits of a blossom in V_{k-1} or of a blossom embedded in a blossom in V_{k-1}. In the first case, $\delta_{k-1}(e) = 1$ from the way we define the edge set of G_{k-1}; in the latter two cases, induction hypothesis states that $\delta_{k-1}(e) = 1$. In case (ii), by induction hypothesis, we have $\delta_{k-1}(e) = 1$. (This holds independent of whether $B \in V_{k-1}$ or $B \subset B' \in V_{k-1}$).

In cases (i) and (ii), we will establish that

(3.8)
$$y_u^{k-1} + y_v^{k-1} + \sum_{\beta : e \in E(\beta)} z_\beta^{k-1} = y_u^{k-2} + y_v^{k-2} + \sum_{\beta : e \in E(\beta)} z_\beta^{k-2} + 1.$$

Note that (3.8) and the fact that $w_k(e) = w_{k-1}(e) + 1$ would imply $\delta_k(e) = \delta_{k-1}(e) = 1$ as desired.

In case (i) we have $y_u^{k-1} = y_u^{k-2}$, $y_v^{k-1} = y_v^{k-2}$, and $z_B^{k-1} = z_B^{k-2} + 1$ while all the other odd sets β spanning $e = (u, v)$ have $z_\beta^{k-1} = z_\beta^{k-2}$. This yields (3.8).

In case (ii), $y_u^{k-1} = y_u^{k-2} + 1$, $y_v^{k-1} = y_v^{k-2} + 1$. If B is a blossom in $\mathcal{O}_{k-1}$, then $z_B^{k-1} = z_B^{k-2} - 1$, and for all the other odd sets β spanning e, we have $z_\beta^{k-1} = z_\beta^{k-2}$. If B is embedded in another outermost blossom $B' \in \mathcal{O}_{k-1}$, $z_{B'}^{k-1} = z_{B'}^{k-2} - 1$ and all the other odd sets β spanning e (including B) have $z_\beta^{k-1} = z_\beta^{k-2}$. In both sub-cases, (3.8) is satisfied.

Now we deal with case (iii). By Proposition 2.1, $B \in V_{k-2}$. So induction hypothesis states that $\delta_{k-2}(e) = 1$. Furthermore, we know that $w_k(e) = w_{k-2}(e) + 2$. Since u and v are elements of $\mathcal{U}_{k-2}$, we have $y_u^{k-2} = y_u^{k-3} + 1$, $y_v^{k-2} = y_v^{k-3} + 1$, $z_B^{k-2} = z_B^{k-3}$, and $z_\beta^{k-2} = z_\beta^{k-3}$ for every other odd set β spanning $e = (u, v)$. As no vertex/blossom of $\mathcal{U}_{k-2}$ is present in G_{k-1}, we have $y_u^{k-1} = y_u^{k-2}$, $y_v^{k-1} = y_v^{k-2}$, and $z_\beta^{k-1} = z_\beta^{k-2}$ for every β that spans e. This guarantees that $\delta_k(e) = \delta_{k-2}(e) = 1$.

Proof of Claim 2. First observe that $\tilde{T}_{k-1}$ is a matching on $\tilde{V}_{k-1} \dot\cup \mathcal{U}_{k-2}$, which forms a partition over V. (In the case when $k = 2$, let $\mathcal{U}_{k-2} = \emptyset$.) Furthermore, $\tilde{T}_{k-1} \setminus \tilde{N}_{k-1}$ is a matching on $(\tilde{V}_{k-1} \cup \mathcal{U}_{k-2}) \setminus \mathcal{U}_{k-1}$. Because we open up recursively the outermost blossoms B with $z_B^{k-1} = 0$ in $\tilde{T}_{k-1} \setminus \tilde{N}_{k-1}$, the resulting matching M'_{k-1} is a matching among the outermost blossoms $B \in \Omega \setminus \mathcal{U}_{k-1}$ with $z_B^{k-1} > 0$ and the vertices $v \in V$ not contained in any such blossom B or any blossom in $\mathcal{U}_{k-1}$. Clearly these blossoms and vertices are a subset of V_k.

Next we need to show that for every edge $e = (u, v) \in M'_{k-1}$, $\delta_k(e) = 1$, so that e is present in G_k. For each such edge, there are three possible cases. Case (i): e originally resides in a blossom B in $(\tilde{V}_{k-1} \cup \mathcal{U}_{k-2}) \setminus \mathcal{U}_{k-1}$. Case (ii): $e \in \mathcal{O}_{k-1} \times \mathcal{E}_{k-1}$. Case (iii): $e \in \mathcal{U}_{k-2} \times \mathcal{U}_{k-2}$. We consider these cases below.

- For case (i), since we only decrease the dual variables of blossoms in $\mathcal{O}_{k-1}$, by Proposition 2.1, B is in V_{k-1} and e must be spanned by one such blos-

som B. Claim 1 states that $\delta_{k-1}(e) = 1$. Furthermore, as $y_u^{k-1} = y_u^{k-2}+1$, $y_v^{k-1} = y_v^{k-2}+1$, $z_B^{k-1} = z_B^{k-2}-1$, and $z_{B'}^{k-1} = z_{B'}^{k-2}$ for all other $B' \in \Omega$ that span e, we have $\delta_k(e) = \delta_{k-1}(e)+1-(1+1-1) = 1$.

- For cases (ii) and (iii), e is an edge in $(\mathcal{O}_{k-1} \times \mathcal{E}_{k-1}) \cup (\mathcal{U}_{k-2} \times \mathcal{U}_{k-2})$. In the former case, since e is a part of G_{k-1}, $\delta_{k-1}(e) = 1$. Furthermore, since $y_u^{k-1} = y_u^{k-2} + 1$, $y_v^{k-1} = y_v^{k-2}$, and $z_B^{k-1} = z_B^{k-2}$ for all $B \in \Omega$ that span e, we have $\delta_k(e) = \delta_{k-1}(e) + 1 - (1) = 1$. In the latter case, we can assume $k \geq 3$. Since e is a part of G_{k-2}, we have $\delta_{k-2}(e) = 1$. As $y_u^{k-1} = y_u^{k-2} = y_u^{k-3} + 1$ and $y_v^{k-1} = y_v^{k-2} = y_v^{k-3} + 1$ while $z_B^{k-1} = z_B^{k-2} = z_B^{k-3}$ for all $B \in \Omega$ that spans e, we have $\delta_k(e) = \delta_{k-2}(e)+2-(1+1) = 1$. This completes the proof.

3.1 Correctness of our maximum weight matching algorithm

The graph H_k**.** We now define the graph H_k, in which we show that the matching T_k and the dual variables satisfy invariants (2.1)-(2.5). As mentioned earlier, the graph H_k is the same as the graph $S_k = (V, F_k)$, except for its edge weight function w'_k, defined as follows:

- For $2 \leq k \leq W-1$ and any edge $e = (u, v)$ where both u and v are *elements* of $\mathcal{U}_k$, we set $w'_k(e) = w_k(e) + 1$; for all other edges $e \in F_k$, we have $w'_k(e) = w_k(e)$.

- In the final iteration, i.e., when $k = W$, we define $w'_k(e) = w_k(e)$ for all $e \in F_W$.

Note that (3.6) always holds by the above definition of w'_k. Observe that the edge set of S_W is $F_W = E_W \cup \cdots \cup E_1$, which is the same as the edge set of G. The edge weight function of S_W is $w_W(e) = W + i - W = i$, for $e \in E_i$. Thus the edge weight function $w_W = w = w'_W$ and hence $H_W = S_W = G$. We will maintain the invariant that T_k is a maximum weight matching in H_k, thus T_W will be a maximum weight matching in $H_W = G$.

The invariants. We now show that invariants (2.1)-(2.5) are satisfied by T_k and the dual variables y_u^k, z_B^k for all $u \in V$ and $B \in \Omega$ in the graph H_k. Lemma 3.2 shows that invariants (2.3)-(2.5) hold at the end of the k-th iteration.

LEMMA 3.2. *For all $u \in V$, we have $y_u^k \geq 0$; for all $B \in \Omega$, we have $z_B^k \geq 0$. Furthermore,*

- *if $z_B^k > 0$ then $\frac{|B|-1}{2}$ edges within the blossom B are in T_k, for all blossoms $B \in \Omega$;*

- *if $y_u^k > 0$ then u is matched in T_k, for all vertices $u \in V$.*

Proof. We first show that invariant (2.5) holds. By the algorithm, for any $u \in V$, y_u^k is at least as large as y_u^{k-1}. Since $y_u^1 \in \{0, 1\}$ for all $u \in V$, it follows that $y_u^k \geq 0$ for every $u \in V$. For any odd set $B \in \Omega$, $z_B^1 \in \{0, 1\}$. In iteration k, z_B^k is decreased from z_B^{k-1} only when B is a blossom in $\mathcal{O}_k$. By Proposition 2.1, $B \in V_k$, since it is an old blossom in iteration k. Since V_k contains outermost blossoms with a positive dual variable from iteration $k-1$, we know $z_B^{k-1} > 0$. Thus $z_B^k = z_B^{k-1} - 1 \geq 0$. This completes the proof of invariant (2.5).

For invariant (2.4), note that T_k is obtained by opening all blossoms in $\tilde{T}_k$. From the way we define opening a blossom, $\frac{|B|-1}{2}$ edges belonging to the defining circuits of B and its embedded blossoms will be in T_k.

For invariant (2.3), suppose $y_u^k > 0$ for some $u \in V$. There are two cases here: *Case (i)* $y_u^{k-1} > 0$, and *Case (ii)* $y_u^{k-1} = 0$. We consider them separately.

Case (i): If $y_u^{k-1} > 0$, by invariant (2.3) in iteration $k-1$, u was matched in T_{k-1}. Let N_{k-1} be the matching after we open all blossoms in $\tilde{N}_{k-1}$ in the graph $\tilde{G}_{k-1}|_{\mathcal{U}_{k-1}}$. If u was an *element* of $\mathcal{U}_{k-1}$, then u was matched in N_{k-1}. Since $T_k \supseteq N_{k-1}$, the vertex u is matched in T_k too. If u was not an element of $\mathcal{U}_{k-1}$, then u was matched by the edges of $T_{k-1} \setminus N_{k-1}$ and u is an element of V_k. Recall that M'_{k-1} is obtained by opening certain blossoms in $\tilde{T}_{k-1} \setminus \tilde{N}_{k-1}$ in $\tilde{G}_{k-1} \dot\cup \tilde{G}_{k-2}|_{\mathcal{U}_{k-2}}$ (when $k = 2$, let $\tilde{G}_{k-2}|_{\mathcal{U}_{k-2}} = \emptyset$). Opening all the remaining blossoms in M'_{k-1} would result in $T_{k-1} \setminus N_{k-1}$. Since $\tilde{M}_k$ is obtained by *augmenting* M'_{k-1}, and $\tilde{T}_k = \tilde{M}_k \cup \tilde{N}_{k-1}$, it follows from the way we define opening a blossom that u is matched in T_k.

Case (ii): If $y_u^{k-1} = 0$, then u was necessarily an element of $\mathcal{E}_{k-1}$. For y_u^k to be positive, u must have become an element of $\mathcal{O}_k \cup \mathcal{U}_k$. So either u is a vertex in $\mathcal{O}_k \cup \mathcal{U}_k$ or u is a part of a blossom $B \in \mathcal{O}_k \cup \mathcal{U}_k$. By Proposition 2.2, in the former case, u is matched in $\tilde{M}_k$; in the latter case, B is matched in $\tilde{M}_k$. In both cases, from the way we define opening a blossom, u remains matched in T_k.

We now show in Lemmas 3.3 and 3.4 that invariants (2.1) and (2.2) are maintained.

LEMMA 3.3. *For all edges $e = (u, v)$ in the graph H_k, $y_u^k + y_v^k + \sum_{B: e \in E(B)} z_B^k \geq w'_k(e)$.*

Proof. Recall that the edge set of H_k is F_k. If $e \in F_k \setminus F_{k-1}$, i.e., if $e \in E_{W+1-k}$, let us assume $w'_{k-1}(e) = w_{k-1}(e) = 0$. By invariant (2.1) of the previous iteration and the fact that all dual variables are non-negative, we have:

(3.9)
$$y_u^{k-1}+y_v^{k-1}+\sum_{B:e\in E(B)} z_B^{k-1} \geq w'_{k-1}(e) \quad \forall e=(u,v)\in F_k.$$

It follows from (3.6) and the fact that $w_{k-1}(e)+1 = w_k(e)$ that either $w'_k(e) = w'_{k-1}(e)+2$, or $w'_{k-1}(e)+1 \geq w'_k(e)$. The former case happens only when both u and v are elements of $\mathcal{U}_k$ and $k < W$. Then $y_u^k + y_v^k + \sum_{B:e\in E(B)} z_B^k = y_u^{k-1} + y_v^{k-1} + \sum_{B:e\in E(B)} z_B^{k-1} + 2 \geq w'_{k-1}(e) + 2 = w'_k(e)$, where the inequality follows from (3.9). In this case, invariant (2.1) holds for the k-th iteration. From now on, we will assume that $w'_{k-1}(e)+1 \geq w'_k(e)$.

We need to consider the following cases about the edge $e=(u,v)$: (i) $e \in E(B)$, where B is a blossom in $\mathcal{O}_k \dot\cup \mathcal{E}_k$, or a blossom in $\mathcal{U}_k$ when $k = W$, (ii) $e \in \tilde V_k \times \tilde V_k$ and e is not spanned by a blossom in $\tilde V_k$, (iii) $e \in \mathcal{U}_{k-1} \times \mathcal{U}_{k-1}$, and (iv) $e \in \tilde V_k \times \mathcal{U}_{k-1}$. We discuss these cases separately.

For case (i), we claim that

(3.10)
$$\begin{aligned} & y_u^k + y_v^k + \sum_{B:e\in E(B)} z_B^k \\ \geq\ & y_u^{k-1} + y_v^{k-1} + \sum_{B:e\in E(B)} z_B^{k-1} + 1 \\ \geq\ & w'_{k-1}(e)+1 \geq w'_k(e). \end{aligned}$$

Invariant (2.1) now follows. The second inequality follows from (3.9) while the first inequality can be easily verified from the way we define the dual variables in the k-th iteration. (Note that for the case when $e=(u,v)$ is spanned by some blossom in $\mathcal{U}_W$, $y_u^W = y_u^{W-1} + 1/2$ and $y_v^W = y_v^{W-1} + 1/2$.)

For case (ii), first observe that by Proposition 2.1, there is no edge of G_k between an element of $\mathcal{E}_k$ and an element of $\mathcal{E}_k \cup \mathcal{U}_k$. So given the edge $e=(u,v)$, if u is an element of $\mathcal{E}_k$ and v an element of $\mathcal{E}_k \cup \mathcal{U}_k$, then $\delta_k(e) \leq 0$, implying that $y_u^{k-1} + y_v^{k-1} + \sum_{B:e\in E(B)} z_B^{k-1} \geq w_k(e) = w'_k(e)$. Furthermore, it is easy to see that $y_u^k + y_v^k + \sum_{B:e\in E(B)} z_B^k \geq y_u^{k-1} + y_v^{k-1} + \sum_{B:e\in E(B)} z_B^{k-1}$. Combining the two preceding inequalities yields invariant (2.1).

By the above discussion, we can assume that either u is an element of $\mathcal{O}_k$ and v is an element of $\mathcal{E}_k \dot\cup \mathcal{O}_k \dot\cup \mathcal{U}_k$, or u and v are both elements of $\mathcal{U}_k$ when $k = W$. In both cases, we argue that (3.10) holds and invariant (2.1) then follows: the first inequality can be easily verified from the way we define the dual variables and the second follows from (3.9).

For case (iii), notice that $w'_k(e) = w'_{k-1}(e)$, $y_u^k = y_u^{k-1}$, $y_v^k = y_v^{k-1}$, and all the odd sets B spanning e have $z_B^k = z_B^{k-1}$. Now invariant (2.1) follows from (3.9).

For case (iv), u is either an element of $\mathcal{E}_{k-1}$, or of $\mathcal{O}_{k-1}$, or of $\mathcal{U}_{k-2}$. We will establish the following:

(3.11)
$$\begin{aligned} & y_u^k + y_v^k + \sum_{B:e\in E(B)} z_B^k \\ \geq\ & y_u^{k-1} + y_v^{k-1} + \sum_{B:e\in E(B)} z_B^{k-1} \\ \geq\ & w'_{k-1}(e)+1 \geq w'_k(e). \end{aligned}$$

And invariant (2.1) would follow from (3.11). The first inequality in (3.11) is easy to verify. We show the second inequality in (3.11) holds by considering all possible cases. First suppose that $e \in F_k \backslash F_{k-1}$. Then by assumption $w'_{k-1}(e) = 0$. Since $v \in \mathcal{U}_{k-1}$, $y_v^{k-1} \geq 1$. As all other dual variables are non-negative, the second inequality holds. So from now on, we can assume that $e \in F_{k-1}$ and this implies that $w'_{k-1}(e) = w'_{k-2}(e)+1$. (In case that $e \in F_{k-1}\backslash F_{k-2}$, let $w'_{k-2}(e) = 0$). We consider the identity of u in the following three cases:

Suppose that u is an element of $\mathcal{E}_{k-1}$. By Proposition 2.1, there is no edge in $\mathcal{E}_{k-1} \times \mathcal{U}_{k-1}$ in $\tilde G_{k-1}$. So $e=(u,v)$ does not belong to the edge set of G_{k-1}, implying that $\delta_{k-1}(e) \leq 0$, and we have $y_u^{k-2} + y_v^{k-2} + \sum_{B:e\in E(B)} z_B^{k-2} \geq w_{k-1}(e) = w'_{k-1}(e)$. From the way we define the dual variables in the $(k-1)$-th iteration, $y_u^{k-1} + y_v^{k-1} + \sum_{B:e\in E(B)} z_B^{k-1} \geq y_u^{k-2} + y_v^{k-2} + \sum_{B:e\in E(B)} z_B^{k-2} + 1$. The second inequality of (3.11) follows from the two preceding inequalities.

Next suppose that u is an element of $\mathcal{O}_{k-1}$. Then invariant (2.1) in the $(k-2)$-th iteration guarantees that $y_u^{k-2} + y_v^{k-2} + \sum_{B:e\in E(B)} z_B^{k-2} \geq w'_{k-2}(e)$. As $y_u^{k-1} = y_u^{k-2} + 1$, $y_v^{k-1} = y_v^{k-2} + 1$, all odd sets B spanning e have $z_B^{k-1} = z_B^{k-2}$, and $w'_{k-2}(e) + 2 = w'_{k-1}(e) + 1$, we have the second inequality of (3.11).

Finally, assume that u is an element of $\mathcal{U}_{k-2}$. We can assume that $k \geq 3$. We use Claim 3 here. So $y_u^{k-2} + y_v^{k-2} + \sum_{B:e\in E(B)} z_B^{k-2} \geq w'_{k-2}(e) + 1$. As $y_v^{k-1} = y_v^{k-2} + 1$, $y_u^{k-1} = y_u^{k-2}$, all odd sets B spanning e have $z_B^{k-1} = z_B^{k-2}$, and $w'_{k-2}(e) + 2 = w'_{k-1}(e) + 1$, the second inequality of (3.11) follows and the proof of the whole lemma is complete.

Claim 3. *For any t, where $2 \leq t \leq W$, let $e=(u,v)$ be an edge in F_t so that u is an element of $\mathcal{U}_{t-1}$ and v an element of $\mathcal{U}_t$. Then $y_u^{t-1} + y_v^{t-1} + \sum_{B:e\in E(B)} z_B^{t-1} \geq w'_{t-1}(e)+1$. (If $e \in F_t\backslash F_{t-1}$, assume $w'_{t-1}(e) = 0$.)*

Proof of Claim 3. If $e \in F_t \backslash F_{t-1}$, then $y_u^{t-1} \geq 1$ while all other dual variables are non-negative. This claim holds easily then. So from now on, we can assume that $e \in F_{t-1}$. This would imply that $w'_{t-1}(e) = w'_{t-2}(e)+1$ (In case $e \in F_{t-1}\backslash F_{t-2}$, we can assume $w'_{k-2}(e) = 0$.)

In the following, we prove the claim by induction. The base case is $t = 2$. The vertex v is an element of $\mathcal{O}_1 \dot\cup \mathcal{E}_1$. However, due to Proposition 2.1, v cannot be an element $\mathcal{E}_1$, since u is in $\mathcal{U}_1$. So $v \in \mathcal{O}_1$. The claim holds because both $y_u^1 = y_v^1 = 1$ and all odd sets B spanning e have $z_B^1 = 0$, while $w_1'(e)$ is only 1. This finishes the base case.

By induction hypothesis, we assume that the claim is true for all $\ell < t$. As v is an element of $\mathcal{U}_t$, there are three cases here: Case (i): v is an element of $\mathcal{O}_{t-1}$, Case (ii): v is an element of $\mathcal{E}_{t-1}$, and Case (iii): v is an element of $\mathcal{U}_{t-2}$. We consider them separately.

Case (i): u is an element of $\mathcal{U}_{t-1}$ and v an element of $\mathcal{O}_{t-1}$.

We have $y_u^{t-2}+y_v^{t-2}+\sum_{B:e\in E(B)} z_B^{t-2} \geq w_{t-2}'(e)$ by invariant (2.1) in the $(t-2)$-th iteration. In the $(t-1)$-th iteration, $y_u^{t-1} = y_u^{t-2} + 1$, $y_v^{t-1} = y_v^{t-2} + 1$, and all odd sets $B \in \Omega$ spanning $e = (u, v)$ have $z_B^{t-1} = z_B^{t-2}$. Now since $w_{t-1}'(e) = w_{t-2}'(e) + 1$, we prove the claim.

Case (ii): u is an element of $\mathcal{U}_{t-1}$ and v an element of $\mathcal{E}_{t-1}$.

By Proposition 2.1, there is no edge between $\mathcal{U}_{t-1}$ and $\mathcal{E}_{t-1}$ in $\tilde{G}_{t-1}$, it follows that $\delta_{t-1}(e) \leq 0$. Then $w_{t-1}(e) \leq y_u^{t-2} + y_v^{t-2} + \sum_{B:e\in E(B)} z_B^{t-2}$. And $y_u^{t-1} = y_u^{t-2} + 1$, $y_v^{t-1} = y_v^{t-2}$ and all odd sets B spanning $e = (u, v)$ have $z_B^{t-1} = z_B^{t-2}$. Since $w_{t-1}'(e) = w_{t-1}(e)$, we prove the claim.

Case (iii): u is an element of $\mathcal{U}_{t-1}$ and v an element of $\mathcal{U}_{t-2}$.

Recall that we assume $e \in F_{t-1}$, therefore we can make use of the induction hypothesis: edge $e = (u, v)$ is slack for H_{t-1}. That is, $y_u^{t-2}+y_v^{t-2}+\sum_{B:e\in E(B)} z_B^{t-2} \geq w_{t-2}'(e) + 1$. And $y_u^{t-1} = y_u^{t-2} + 1$, $y_v^{t-1} = y_v^{t-2}$ and all odd sets B spanning $e = (u, v)$ have $z_B^{t-1} = z_B^{t-2}$. Since $w_{t-1}'(e) = w_{t-2}'(e) + 1$, we prove the claim.

LEMMA 3.4. *For all edges $e = (u, v)$ in the matching T_k, we have $y_u^k + y_v^k + \sum_{B:e\in E(B)} z_B^k = w_k'(e)$.*

Proof. Recall that T_k is obtained by opening all blossoms in $\tilde{T}_k = \tilde{M}_k \dot\cup \tilde{N}_{k-1}$ in the graph $\tilde{G}_k \dot\cup \tilde{G}_{k-1}|_{\mathcal{U}_{k-1}}$. Let N_{k-1} be the matching after we open all blossoms of $\tilde{N}_{k-1}$ in $\tilde{G}_{k-1}|_{\mathcal{U}_{k-1}}$.

First assume that $e = (u, v) \in N_{k-1}$. Then u and v are elements of $\mathcal{U}_{k-1}$. Since $N_{k-1} \subseteq T_{k-1}$, it follows from invariant (2.2) of the previous iteration that $y_u^{k-1} + y_v^{k-1} + \sum_{B:e\in E(B)} z_B^{k-1} = w_{k-1}'(e)$. As $\mathcal{U}_{k-1}$ is absent from G_k, $y_u^k = y_u^{k-1}$, $y_v^k = y_v^{k-1}$, and $\sum_{B:e\in E(B)} z_B^k = \sum_{B:e\in E(B)} z_B^{k-1}$. Also $w_k'(e) = w_k(e) = w_{k-1}(e) + 1 = w_{k-1}'(e)$. Thus it follows that $y_u^k + y_v^k + \sum_{B:e\in E(B)} z_B^k = w_k'(e)$.

So let us assume that $e = (u, v) \in T_k \setminus N_{k-1}$. Then e is either present in G_k as an edge, or e is a part of the defining circuits of a blossom in V_k, or of some of its embedded blossoms By Claim 1 and the definition of the edge set of G_k, $\delta_k(e) = 1$. This means that (3.12) holds:

$$y_u^{k-1} + y_v^{k-1} + \sum_{B:e\in E(B)} z_B^{k-1} = w_k(e) - 1. \tag{3.12}$$

We have the following cases: (i) $e \in \mathcal{E}_k \times \mathcal{O}_k$, or (ii) $e \in \mathcal{U}_k \times \mathcal{U}_k$, or (iii) e is a part of the defining circuits of a blossom $B \in \mathcal{E}_k \dot\cup \mathcal{O}_k \dot\cup \mathcal{U}_k$, or some of its embedded blossoms.

For case (i), $y_u^k = y_u^{k-1}$, $y_v^k = y_v^{k-1} + 1$, and all odd sets B spanning e have $z_B^{k-1} = z_B^{k-2}$. Since $w_k'(e) = w_k(e)$, the tightness of e follows from (3.12).

For case (ii), all odd sets B spanning e have $z_B^{k-1} = z_B^{k-2}$. When $k < W$, $y_u^k = y_u^{k-1} + 1$, $y_v^k = y_v^{k-1} + 1$ and $w_k'(e) = w_k(e) + 1$; when $k = W$, $y_u^k = y_u^{k-1} + 1/2$, $y_v^k = y_v^{k-1} + 1/2$, and $w_k'(e) = w_k(e)$. The tightness of e follows from (3.12) and the above observations.

For case (iii), let us consider the three subcases.

- $B \in \mathcal{E}_k$. Then $y_u^k = y_u^{k-1}$, $y_v^k = y_v^{k-1}$, and $z_B^k = z_B^{k-1} + 1$. All other odd sets B' spanning e have $z_{B'}^k = z_{B'}^{k-1}$. Also $w_k'(e) = w_k(e)$. The tightness of $e = (u, v)$ now follows from (3.12).

- $B \in \mathcal{O}_k$. Then $y_u^k = y_u^{k-1} + 1$, $y_v^k = y_v^{k-1} + 1$, and $z_B^k = z_B^{k-1} - 1$. All other odd sets B' spanning e have $z_{B'}^k = z_{B'}^{k-1}$. Also $w_k'(e) = w_k(e)$. The tightness of $e = (u, v)$ now follows from (3.12).

- $B \in \mathcal{U}_k$. All odd sets B' (including B) spanning e have $z_{B'}^k = z_{B'}^{k-1}$. For $k < W$, $y_u^k = y_u^{k-1} + 1$, $y_v^k = y_v^{k-1} + 1$, and $w_k'(e) = w_k(e) + 1$. For $k = W$, $y_u^k = y_u^{k-1} + 1/2$, $y_v^k = y_v^{k-1} + 1/2$, and $w_k'(e) = w_k(e)$. The tightness of e in both subcases follows from (3.12).

We have thus shown Theorem 3.1 stated below. We can now conclude that T_k is a maximum weight matching in H_k. Hence the matching T_W returned by our algorithm is a maximum weight matching in $H_W = G$.

THEOREM 3.1. *For every k, where $1 \leq k \leq W$, invariants (2.1)-(2.5) are maintained in our algorithm.*

3.2 Implementation We now discuss how to implement our algorithm efficiently. In the k-th iteration we need to perform the following tasks.

(i) Form the working graph G_k with the vertex set V_k.

(ii) Compute a maximum cardinality matching M_k in G_k. For $k \geq 2$, M_k is obtained by augmenting M'_{k-1}.

(iii) Close the blossoms in M_k to form the matching $\tilde{M}_k$ in the graph $\tilde{G}_k$ with the vertex set $\tilde{V}_k$.

(iv) Form a Hungarian forest $\tilde{F}$ based on $\tilde{G}_k$ and $\tilde{M}_k$. Determine the Gallai-Edmonds decomposition of $\tilde{V}_k = \mathcal{E}_k \dot{\cup} \mathcal{O}_k \dot{\cup} \mathcal{U}_k$, and define the dual variables $\{y_u^k\}_{u \in V} \cup \{z_B^k\}_{B \in \Omega}$ based on this decomposition.

(v) If $k < W$, then form M'_k as the initial matching for the next iteration.

Task (iv) can be done in $O(m)$ time, since the blossoms are closed. For task (ii), computing M_k is straightforward if $k = 1$. When $k \geq 2$, first find *any* maximum cardinality matching and let its cardinality be t. Create $|V_k| - 2t$ dummy vertices and connect each of them to all vertices in V_k left unmatched by M'_{k-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 vertices in V_k left unmatched by M'_{k-1} can be matched to dummy vertices. So this perfect matching restricted to the rest of the vertices in V_k will be the desired matching M_k. The entire task (ii) thus can be done in either $O(\sqrt{n}m \log_n(n^2/m))$ time [25], or it can be done with high probability in $O(n^\omega)$ time [27, 37].

For task (iii), several algorithms [16, 18, 36] can close the blossoms in $O(m)$ time if the given matching is already of maximum cardinality. All these algorithms make use of the disjoint set union structure [19].

For tasks (i) and (v), we will maintain a separate forest data structure whose roots will form a partition over V. Initially, this forest contains all vertices in V and each of them is a root. When we execute task (iii), if a blossom is detected, then this blossom is a new vertex in this forest and its children are those vertices in V or other blossoms embedded in it. The defining circuit of this blossom, along with the base, is also recorded. (This will be required when we open the blossom.) Finally, we associate each non-leaf node B in this forest with its corresponding dual variable z_B. This information will be needed when we decide to open a blossom or not at the end of each iteration while forming M'_k for the next iteration. Note that since the vertices and the blossoms form a laminar family, there can be at most $2n - 1$ vertices in this forest.

In performing task (v), i.e., recursively opening outermost blossoms B with $z_B^k = 0$, if a blossom B is opened, then we remove B also from this data structure. It is easy to see that in the beginning of the k-th iteration, the roots of the forest are exactly the vertices of $\mathbb{V}_k$ for the k-th iteration. Remove the blossoms and vertices of $\mathcal{U}_{k-1}$ from $\mathbb{V}_k$ to obtain V_k. While deciding the edge set of the working graph G_k, i.e., those edges e with $\delta_k(e) = 1$, we can do the following: for all edges $e \in E$, update $\delta_k(e)$ at the end of each iteration (whether e is a part of the starting graph S_k or not). Observe that in each iteration, the odd sets $B \in \Omega$ whose dual variables thar are really changed (increasing/decreasing by 1) are those of the outermost blossoms in $\mathcal{O}_k \dot{\cup} \mathcal{E}_k$. By the forest data structure we maintain, we can easily compute $\delta_k(e)$ for all edges $e \in E$ in $O(n + m)$ time in each iteration.

Finally, by Definition 2.2, observe that when a blossom B is opened—either its base is left unmatched, or one of its vertices is matched to some other vertex not in B, how the remaining vertices in B should be matched to each other can be easily decided, since we record its defining circuit. Thus Tasks (i) and (v) and the task of maintaining this forest data structure during the whole iteration can be done in $O(n + m)$ time. At the very end, using such a data structure, we can open all blossoms to transform $\tilde{T}_W$ to T_W at the end of our algorithm. This takes $O(n)$ time.

We have thus shown the following result.

THEOREM 3.2. *A maximum weight matching in $G = (V, E)$ whose edge weights come from $\{1, \ldots, W\}$ can be computed in time $O(W \cdot t(m, n))$, where $t(m, n)$ is the time for computing a maximum cardinality matching in G.*

Theorems 1.1 and 1.2 stated in Section 1 follow from the above result.

Acknowledgement

We are grateful to Seth Pettie for pointing out several references to us.

References

[1] R. Ahuja, T. Magnati, and J. Orlin. Network Flows: Theory, Algorithms, and Applications.

[2] T. Anderson, S. Owiki, J. Saxe, and C. Thacker. High-speed switch scheduling for local-area networks. In *J. ACM Transactions on Computer Systems* 11(4): 319-352, 1993.

[3] C. Berge. Sur le couplage maximum d'un graphe In *Comptes Rendus Hebdomadaires des Séances de l'Académie des Sciences* 247: 258-359, 1958.

[4] N. Blum. A new approach to maximum matching in general graphs. In 17th *ICALP*: 586-597, 1990.

[5] J. Cheriyan and K. Mehlhorn. Algorithms for Dense Graphs and Networks on the Random Access Computer. In *Algorithmica* 15: 521549, 1996.

[6] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. In *J. of Symbolic Computation* 9(3): 251-280, 1990.

[7] W. Cunningham and A. Marsh. A primal algorithm for optimum matching. In *Polyhdedral Combinatorics—Dedicated to the Memory of D.R. Fulkerson*: 50-72, 1978.

[8] U. Derigs. A shortest augmenting path method for solving minimal perfect matching problems. In Networks 11 : 379-390, 1981.

[9] R. Duan and S. Pettie. Approximating maximum weight matching in near-linear time. In 50th *FOCS*: 673-682, 2010.

[10] R. Duan and H.-H. Su. A Scaling Algorithm for Maximum Weight Matchings in Bipartite Graphs. In 23rd *SODA*, 2012.

[11] J. Edmonds. Paths, trees, and flowers. In *Canadian J. of Mathematics* 17: 449-467, 1965.

[12] J. Edmonds. Maximum matching and a polyhedron with $(0, 1)$ vertices. In *J. of Res. Nat. Bureau of Standards Section* 69B: 125-130, 1965.

[13] J. Edmonds and R. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. In *Combinatorial Structures and their Applications*: 93–96, 1970.

[14] T. Feder and R. Motwani . Clique partitions, graph compression and speeding-up algorithms. In *J. of Computer and System Sciences* 51(2): 261-272, 1995. Conference version in STOC 1991.

[15] H. Gabow. An efficient implementation of Edmonds' algorithm for maximum matching on graphs. In *J. of ACM*23: 221-234, 1976.

[16] H. Gabow. Scaling algorithms for network problems. In 24th *FOCS*: 248-257, 1983.

[17] H. Gabow. Data structures for weighted matching and nearest common ancestors with linking. In 1st *SODA*: 434-443, 1990.

[18] H. Gabow and R. Tarjan. Faster scaling algorithms for general graph-matching problems. In *J. of ACM* 38: 815-853, 1991.

[19] H. Gabow and R. Tarjan. A Linear-Time Algorithm for a Special Case of Disjoint Set Union. In *J. of Computing System*30(2): 209-221, 1985.

[20] H. Gabow and R. Tarjan. Almost-optimum speed-ups of algorithms for bipartite matching and related problems. In 20th *STOC*, 514-527, 1988.

[21] H. Gabow and R. Tarjan. Faster scaling algorithms for network problems. In *SIAM J. Comput.* 18: 1013-1036, 1989.

[22] T. Gallai. Maximale systeme unabhängiger kanten. In *A Magyar Tudományos Akadémia—Matematikai Kutató Intézetének Közleményei* 9: 401-413, 1964.

[23] A. Goel, M. Kapralov, and S. Khanna Perfect matchings in $O(n \log n)$ time in regular bipartite graphs. In 42nd *STOC*: 39-46, 2010.

[24] A. Goldberg. Scaling Algorithms for the Shortest Paths Problem. In *SIAM J. Comput.*24(3): 494-504, 1995. Conference version in SODA 1993.

[25] A. Goldberg and A. Karzanov. Maximum skew-symmetric flows. In 3th *ESA*: 155-170, 1995.

[26] A. Goldberg and R. Kennedy. Global price updates help. In *SIAM Journal on Discrete Mathematics*10: 551-572, 1997.

[27] N. Harvey. Algebraic algorithms for matching and matroid problems. To appear in *SIAM J. Comput.*. Conference version in FOCS 2006.

[28] J. Hopcroft and R. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. In *SIAM J. Comput.* 2: 225-231, 1973. Conference Version in 12th Symposium on Switching and Automata Theory.

[29] M.-Y. Kao, T. W. Lam, W.-K. Sung, H.-F. Ting. A decomposition theorem for maximum weight bipartite matchings. In *SIAM J. Comput.* 31(1): 18-26, 2001.

[30] A. Karzanov. O nakhozhedenii maksimal'nogo potoka v setykh spetsial'nogo vida i nekotorykh prilozheniyakh. In *Matematicheskie Voprosy Upravleniya Proizvodstvom* 31(1): 81-94, 1973.

[31] D. Kőnig. Graphok és matrixok In Matematikai és Fizikai Lapok 38: 116-119, 1931.

[32] H.W. Kuhn. The Hungarian method for the assignment problem In Naval Research Logistics Quarterly2: 83-97, 1955.

[33] A. Schrijver. Combinatorial Optimization - Polyhedra and Efficiency.

[34] L. Lovász and M. Plummer. Matching theory.

[35] T. Anderson, S. Owiki, J. Saxe, and C. Thacker. The iSLIP scheduling algorithm for input-queued switches. In *IEEE/ACM Transactions on Networking* 7(2): 188-201, 1999.

[36] S. Micali and V. Vazirani. An $O(\sqrt{|V|}|E|)$ algorithm for finding maximum matching in general graphs. In 21st *FOCS*: 434-443, 1980.

[37] M. Mucha and P. Sankowski. Maximum matchings via Gaussian elimination. In 45st *FOCS*: 248-255, 2004.

[38] M. Mucha and P. Sankowski. Maximum Matchings in Planar Graphs via Gaussian Elimination In *Algorithmica*45(1): 3-20, 2006.

[39] J. Petersen. Die theorie der regulären graphs, In *Acta Mathematica* 15: 193-220, 1891.

[40] P. Sankowski. Maximum weight bipartite matching in matrix multiplication Time. In *Theoretical Computer Science* 410: 4480-4488, 2009. Conference version in ICALP 2006.

[41] N. Tomizawa. On some techniques useful for solution of transportation network problems. In *Networks* 1: 173-194, 1971.

[42] W. Tutte. The factorization of linear graphs, In *J. of the London Mathematical Society* 22: 107-111, 1947.

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