SIAM J. COMPUT.
Vol. 31, No. 1, pp. 18–26

A DECOMPOSITION THEOREM FOR MAXIMUM WEIGHT BIPARTITE MATCHINGS[*]

MING-YANG KAO[†], TAK-WAH LAM[‡], WING-KIN SUNG[‡], AND HING-FUNG TING[‡]

Abstract. Let G be a bipartite graph with positive integer weights on the edges and without isolated nodes. Let n, N, and W be the node count, the largest edge weight, and the total weight of G. Let $k(x, y)$ be $\log x/\log(x^2/y)$. We present a new decomposition theorem for maximum weight bipartite matchings and use it to design an $O(\sqrt{n}W/k(n, W/N))$-time algorithm for computing a maximum weight matching of G. This algorithm bridges a long-standing gap between the best known time complexity of computing a maximum weight matching and that of computing a maximum cardinality matching. Given G and a maximum weight matching of G, we can further compute the weight of a maximum weight matching of $G - \{u\}$ for all nodes u in $O(W)$ time.

Key words. all-cavity matchings, maximum weight matchings, minimum weight covers, graph algorithms, unfolded graphs

AMS subject classifications. 05C05, 05C70, 05C85, 68Q25

PII. S0097539799361208

1. Introduction.

Let $G = (X, Y, E)$ be a bipartite graph with positive integer weights on the edges. A *matching* of G is a subset of node-disjoint edges of G. Let $\mathrm{mwm}(G)$ (respectively, $\mathrm{mm}(G)$) denote the maximum weight (respectively, cardinality) of any matching of G. A *maximum weight* matching is one whose weight is $\mathrm{mwm}(G)$. Let N be the largest weight of any edge. Let W be the total weight of G. Let n and m be the numbers of nodes and edges of G; to avoid triviality, we maintain $m = \Omega(n)$ throughout the paper.

The problem of finding a maximum weight matching of a given G has a rich history. The first known polynomial-time algorithm is the $O(n^3)$-time Hungarian method [15]. Fredman and Tarjan [5] used Fibonacci heaps to improve the time to $O(n(m + n \log n))$. Gabow [6] introduced scaling to solve the problem in $O(n^{3/4} m \log N)$ time by taking advantage of the integrality of edge weights. Gabow and Tarjan [7] improved the scaling method to further reduce the time to $O(\sqrt{n}m \log(nN))$. For the case where the edges all have weight 1, i.e., $N = 1$ (and $W = m$), Hopcroft and Karp [11] gave an $O(\sqrt{n}W)$-time algorithm, and Feder and Motwani [4] improved the time complexity to $O(\sqrt{n}W/k(n, m))$, where $k(x, y) = \log x/\log(x^2/y)$. It has remained open whether the gap between the running times of the Gabow–Tarjan algorithm and the latter two algorithms can be closed for the case where $W = o(m \log(nN))$.

We resolve this open problem in the affirmative by giving an $O(\sqrt{n}W/k(n, W/N))$-time algorithm for general W. Note that $W/N = m$ when all the edges have the same weight. The algorithm does not use scaling but instead employs a novel decomposition theorem for weighted bipartite matchings (Theorem 2.2). We also use the theorem to

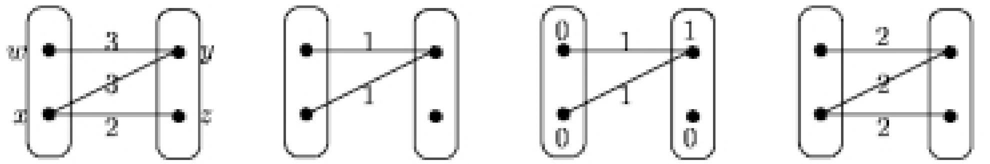
---

[*] Received by the editors September 15, 1999; accepted for publication (in revised form) November 8, 2000; published electronically May 31, 2001. A preliminary version appeared in the *Proceedings of the 7th Annual European Symposium on Algorithms*, Lecture Notes in Comput. Sci. 1643, Springer, Prague, Czech Republic, 1999, pp. 439–449.

http://www.siam.org/journals/sicomp/31-1/36120.html

[†] Department of Computer Science, Yale University, New Haven, CT 06520 (kao-ming-yang@cs.yale.edu). This author's research was supported in part by NSF grant 9531028.

[‡] Department of Computer Science and Information Systems, University of Hong Kong, Hong Kong (twlam@csis.hku.hk, wksung@csis.hku.hk, hfting@csis.hku.hk). The research of these authors was supported in part by Hong Kong RGC grant HKU-7027/98E.



(a) G (b) G_h (c) C_h (d) $G_h^{\triangle}$

FIG. 1. *Consider* $h = 1$. G *is decomposed into* G_h *and* $G_h^{\triangle}$; C_h *is a minimum weight cover of* G_h.

solve the *all-cavity maximum weight matching* problem which, given G and a maximum weight matching of G, asks for $\text{mwm}(G - \{u\})$ for all nodes u in G. This problem has applications to tree comparisons [2, 14]. The case where $N = 1$ has been studied by Chung [2]. Recently, Kao, Lam, Sung, and Ting [12] gave an $O(\sqrt{n}m \log N)$-time algorithm for general N. This paper presents a new algorithm that runs in $O(W)$ time.

Section 2 presents the decomposition theorem and uses it to compute the weight of a maximum weight matching. Section 3 gives an algorithm to construct a maximum weight matching. Section 4 solves the all-cavity matching problem.

2. The decomposition theorem.

In section 2.1, we state the decomposition theorem and use the theorem to design an algorithm to compute the weight $\text{mwm}(G)$ in $O(\sqrt{n}W/k(n, W/N))$ time. In section 2.2, we prove the decomposition theorem. In section 3, we further construct a maximum weight matching itself within the same time bound.

2.1. An algorithm for computing mwm(G).

Let $V(G)$ be the node set of G, i.e., $X \cup Y$. Let $w(u, v)$ denote the weight of an edge $uv \in G$; if u is not adjacent to v, let $w(u, v) = 0$. A *cover* of G is a function $C : X \cup Y \to \{0, 1, 2, \ldots\}$ such that $C(x) + C(y) \geq w(x, y)$ for all $x \in X$ and $y \in Y$. Let $w(C) = \sum_{z \in X \cup Y} C(z)$ be the weight of C. C is a *minimum weight* cover if $w(C)$ is the smallest possible. Let $\text{mwc}(G)$ denote the weight of a minimum weight cover of G. A minimum weight cover is a dual of a maximum weight matching as stated in the next fact.

FACT 2.1 (see [1]). *Let* C *be a cover and* M *be a matching of* G. *The following statements are equivalent.*

1. C *is a minimum weight cover and* M *is a maximum weight matching of* G.
2. $\sum_{uv \in M} w(u, v) = \sum_{u \in X \cup Y} C(u)$.
3. *Every node in* $\{u \mid C(u) > 0\}$ *is matched by some edge in* M, *and* $C(u) + C(v) = w(u, v)$ *for all* $uv \in M$.

For an integer $h \in [1, N]$, we divide G into two lighter bipartite graphs G_h and $G_h^{\triangle}$ as follows:

- G_h is formed by the edges uv of G with $w(u, v) \in [N - h + 1, N]$. Each edge uv in G_h has weight $w(u, v) - (N - h)$. For example, G_1 is formed by the heaviest edges of G, and the weight of each edge is exactly one.
- Let C_h be a minimum weight cover of G_h. $G_h^{\triangle}$ is formed by the edges uv of G with $w(u, v) - C_h(u) - C_h(v) > 0$. The weight of uv is $w(u, v) - C_h(u) - C_h(v)$.

An example is depicted in Figure 1. Note that the total weight of G_h and $G_h^{\triangle}$ is at most W.

The next theorem is the decomposition theorem.

THEOREM 2.2. $\mathrm{mwm}(G) = \mathrm{mwm}(G_h) + \mathrm{mwm}(G_h^{\triangle})$; *in particular,* $\mathrm{mwm}(G) = \mathrm{mm}(G_1) + \mathrm{mwm}(G_1^{\triangle})$.

Proof. See section 2.2. □

Theorem 2.2 suggests the following recursive algorithm to compute $\mathrm{mwm}(G)$.

PROCEDURE Compute-MWM(G).

1. Construct G_1 from G.
2. Compute $\mathrm{mm}(G_1)$ and find a minimum weight cover C_1 of G_1.
3. Construct $G_1^{\triangle}$ from G and C_1.
4. If $G_1^{\triangle}$ is empty, then return $\mathrm{mm}(G_1)$; otherwise, return $\mathrm{mm}(G_1)$+Compute-MWM($G_1^{\triangle}$).

THEOREM 2.3. Compute-MWM(G) *finds* $\mathrm{mwm}(G)$ *in* $O(\sqrt{n}W/k(n, W/N))$ *time.*

Proof. The correctness of Compute-MWM follows from Theorem 2.2. Below, we analyze the running time. We initialize a maximum heap [3] in $O(m)$ time to store the edges of G according to their weights. Let $T(n, W, N)$ be the running time of Compute-MWM excluding this initialization. Let L be the set of the heaviest edges in G. Then Step 1 takes $O(|L| \log m)$ time. In Step 2, we can compute $\mathrm{mm}(G_1)$ in $O(\sqrt{n}|L|/k(n, |L|))$ time [4]. From this matching, C_1 can be found in $O(|L|)$ time [1]. Let L_1 be the set of the edges of G adjacent to some node u with $C_1(u) > 0$; i.e., L_1 consists of the edges of G whose weights are reduced in $G_1^{\triangle}$. Let $\ell_1 = |L_1|$. Step 3 updates every edge of L_1 in the heap in $O(\ell_1 \log m)$ time. As $L \subseteq L_1$, Steps 1 to 3 altogether use $O(\sqrt{n}\ell_1/k(n, \ell_1))$ time. Since the total weight of $G_1^{\triangle}$ is at most $W - \ell_1$, Step 4 uses at most $T(n, W - \ell_1, N')$ time, where $N' < N$ is the maximum edge weight of $G_1^{\triangle}$. In summary, for some positive integer $\ell_1 \le W$,

$$T(n, W, N) = O(\sqrt{n}\ell_1/k(n, \ell_1)) + T(n, W - \ell_1, N'),$$

where $T(n, 0, N') = 0$. By recursion, for some positive integers $\ell_1, \ell_2, \ldots, \ell_p$ with $p \le N$ and $\sum_{1 \le i \le p} \ell_i = W$,

$$\begin{aligned} T(n, W, N) &= O\left(\sqrt{n}\left(\frac{\ell_1}{k(n, \ell_1)} + \frac{\ell_2}{k(n, \ell_2)} + \cdots + \frac{\ell_p}{k(n, \ell_p)}\right)\right) \\ &= O\left(\frac{\sqrt{n}}{\log n}\left(\left(\sum_{1 \le i \le p} \ell_i\right) \log n^2 - \sum_{1 \le i \le p} \ell_i \log \ell_i\right)\right). \end{aligned}$$

Since $x \log x$ is convex, by Jensen's inequality [10],

$$\sum_{1 \le i \le p} \ell_i \log \ell_i \ge \left(\sum_{1 \le i \le p} \ell_i\right) \log \frac{\sum_{1 \le i \le p} \ell_i}{p} \ge W \log \frac{W}{N}.$$

Therefore,

$$\begin{aligned} T(n, W, N) &= O\left(\frac{\sqrt{n}}{\log n}\left(W \log n^2 - W \log \frac{W}{N}\right)\right) \\ &= O\left(\frac{\sqrt{n}W}{\log n / \log(n^2/\frac{W}{N})}\right) = O\big(\sqrt{n}W/k(n, W/N)\big). \end{aligned}$$ □

2.2. Proof of Theorem 2.2. This section proves the statement that $\mathrm{mwm}(G) = \mathrm{mwm}(G_h) + \mathrm{mwm}(G_h^{\triangle})$, where $G_h^{\triangle}$ is defined according to an arbitrary minimum weight cover C_h of G_h. By Fact 2.1, it suffices to prove $\mathrm{mwc}(G) = w(C_h) + \mathrm{mwc}(G_h^{\triangle})$.

To show the direction $\mathrm{mwc}(G) \leq w(C_h) + \mathrm{mwc}(G_h^{\triangle})$, note that any cover D of $G_h^{\triangle}$ augmented with C_h gives a cover C of G, where $C(u) = C_h(u) + D(u)$ for each node u of G. Then $C(u) + C(v) \geq w(u, v)$ for all edges uv of G. Thus, $\mathrm{mwc}(G) \leq w(C_h) + \mathrm{mwc}(G_h^{\triangle})$.

To show the direction $w(C_h) + \mathrm{mwc}(G_h^{\triangle}) \leq \mathrm{mwc}(C)$, let C be a minimum weight cover of G. A node u of G is called *bad* if $C(u) < C_h(u)$. Lemma 2.4 below shows that G must have a minimum weight cover C allowing no bad node. Then we can construct a cover D of $G_h^{\triangle}$ as follows. For each node u of G, define $D(u) = C(u) - C_h(u)$, which must be at least 0. D is a cover of $G_h^{\triangle}$ because for any edge uv of $G_h^{\triangle}$, $D(u) + D(v) = C(u) + C(v) - C_h(u) - C_h(v) \geq w(u, v) - C_h(u) - C_h(v)$. Note that $w(D) = w(C) - w(C_h)$. Thus, $\mathrm{mwc}(G_h^{\triangle}) \leq w(C) - w(C_h)$, or equivalently, $\mathrm{mwc}(G_h^{\triangle}) + w(C_h) \leq \mathrm{mwc}(G)$.

The next lemma concludes the proof of Theorem 2.2.

LEMMA 2.4. *There exists a minimum weight cover of G such that no node of G is bad.*

Proof. Suppose, for the sake of contradiction, that every minimum weight cover allows some bad node. Then we can obtain a contradiction by constructing another minimum weight cover with no bad node.

Let C be a minimum weight cover of G with u as a bad node, i.e., $C(u) < C_h(u)$. Recall that C_h is a minimum weight cover of G_h. Consider a maximum weight matching M of G_h. By Fact 2.1, since $C_h(u) > C(u) \geq 0$, u is matched by an edge in M, say, to a node v, and $C_h(u) + C_h(v) = w(u, v) - (N - h)$. We call v the *mate* of u. Note that v cannot be a bad node; otherwise, $C(u) + C(v) < w(u, v) - (N - h) \leq w(u, v)$ and a contradiction occurs.

Since C is a cover of G, $C(u) + C(v) \geq w(u, v)$. Thus, $C(v) \geq w(u, v) - C(u) \geq N - h + C_h(u) + C_h(v) - C(u)$. Define another cover C' of G as follows. For each bad node defined by C, let v be the mate of u, define $C'(u) = C_h(u)$ and $C'(v) = C(v) - (C_h(u) - C(v))$. Note that u is not a bad node with respect to C', and neither is v since $C'(v) \geq N - h + C_h(v) \geq C_h(v)$. For all other nodes x, $C'(x)$ is the same as $C(x)$. Therefore, if C' is a cover of G, C' allows no bad node. Also, $w(C') = w(C)$.

It remains to prove that C' is a cover of G. By the definition of C', $C'(v) < C(v)$ if and only if v is the mate of a bad node with respect to C. Suppose C' is not a cover of G. Then there exists an edge vt such that $C'(v) + C'(t) \leq w(v, t)$ and v is the mate of a bad node. Recall that the latter implies that $C'(v) \geq N - h + C_h(v)$. In other words,

$$C'(t) < w(v, t) - C'(v) \leq w(v, t) - (N - h) - C_h(v).$$

We can derive a contradiction as follows.

Case 1: $w(v, t) \leq N - h$. Then $C'(t) < -C_h(v) \leq 0$, which contradicts that $C'(t) \geq C_h(t) \geq 0$.

Case 2: $w(v, t) > N - h$. Then G_h contains the edge vt and $C_h(v) + C_h(t) \geq w(v, t) - (N - h)$. Thus, $C'(t) < w(v, t) - (N - h) - C_h(v) \leq C_h(t)$, which contradicts the fact that C' allows no bad node.

In conclusion, C' is a cover of G. Together with the fact that $w(C) = w(C')$, we obtain the desired contradiction that C' is a minimum weight cover of G with no bad node. Lemma 2.4 follows. $\square$

3. Construct a maximum weight matching. The algorithm in section 2.1 only computes the value of $\mathrm{mwm}(G)$. To report the edges involved, we show below how to first construct a minimum weight cover of G in $O(\sqrt{n}W/k(n, W/N))$ time and then use this cover to construct a maximum weight matching in $O(\sqrt{n}m/k(n, m))$ time. Thus, the time required to construct a maximum weight matching is $O(\sqrt{n}W/k(n, W/N))$.

Lemma 3.1. *Assume that* h, G_h, C_h, *and* $G_h^{\triangle}$ *are defined as in section* 2. *Let* $C_h^{\triangle}$ *be any minimum weight cover of* $G_h^{\triangle}$. *If* D *is a function on* $V(G)$ *such that for every* $u \in V(G)$, $D(u) = C_h(u) + C_h^{\triangle}(u)$, *then* D *is a minimum weight cover of* G.

Proof. Consider any edge uv of G. If uv is not in $G_h^{\triangle}$, then $w(u, v) \le C_h(u) + C_h(v) \le D(u) + D(v)$. Assume that uv is in $G_h^{\triangle}$. Note that its weight in $G_h^{\triangle}$ is $w(u, v) - C_h(u) - C_h(v)$. Since $C_h^{\triangle}$ is a cover, $C_h^{\triangle}(u) + C_h^{\triangle}(v) \ge w(u, v) - C_h(u) - C_h(v)$. Thus, $D(u) + D(v) = C_h(u) + C_h^{\triangle}(u) + C_h(v) + C_h^{\triangle}(v) \ge w(u, v)$. It follows that D is a cover of G. To show that D is a minimum weight one, we observe that

$$\begin{aligned} \textstyle\sum_{u \in V(G)} D(u) &= \textstyle\sum_{u \in V(G)} C_h(u) + C_h^{\triangle}(u) \\ &= \textstyle\sum_{u \in V(G)} C_h(u) + \sum_{u \in V(G)} C_h^{\triangle}(u) \\ &= \mathrm{mwm}(G_h) + \mathrm{mwm}(G_h^{\triangle}) && \text{by Fact 2.1} \\ &= \mathrm{mwm}(G) && \text{by Theorem 2.2.} \end{aligned}$$

By Fact 2.1, D is minimum. □

By Lemma 3.1, a minimum weight cover of G can be computed using a recursive procedure similar to Compute-MWM as follows.

Procedure Compute-Min-Cover(G).

1. Construct G_1 from G.
2. Find a minimum weight cover C_1 of G_1.
3. Construct $G_1^{\triangle}$ from G and C_1.
4. If $G_1^{\triangle}$ is empty, then return C_1; otherwise, let $C_1^{\triangle} =$ Compute-Min-Cover($G_1^{\triangle}$) and return D, where for all nodes u in G, $D(u) = C_1(u) + C_1^{\triangle}(u)$.

Theorem 3.2. Compute-Min-Cover(G) *correctly computes a minimum weight cover of* G *in* $O(\sqrt{n}W/k(n, W/N))$ *time.*

Proof. The correctness of Compute-Min-Cover(G) follows from Lemma 3.1. For the time complexity, the analysis is similar to that of Theorem 2.3. □

Now, we show how to recover a maximum weight matching of G from a minimum weight cover D of G.

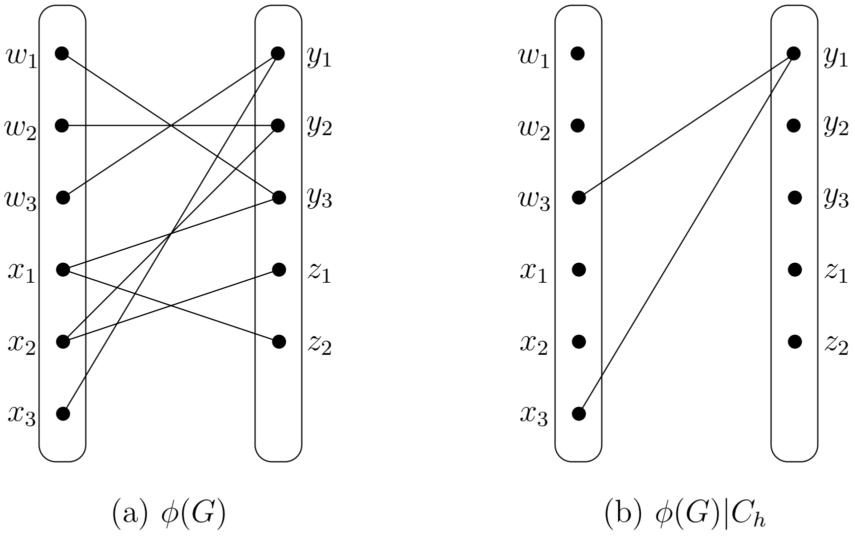
Procedure Recover-Max-Matching(G, D).

1. Let H be the subgraph of G that contains all edges uv with $w(u, v) = D(u) + D(v)$.
2. Make two copies of H. Call them H^a and H^b. For each node u of H, let u^a and u^b denote the corresponding nodes in H^a and H^b, respectively.
3. Union H^a and H^b to form H^{ab}, and add to H^{ab} the set of edges $\{u^a u^b \mid u \in V(H),\ D(u) = 0\}$.
4. Find a maximum cardinality matching K of H^{ab} and return the matching $K^a = \{uv \mid u^a v^a \in K\}$.

Theorem 3.3. Recover-Max-Matching(G, D) *correctly computes a maximum weight matching of* G *in* $O(\sqrt{n}m/k(n, m))$ *time.*

Proof. The running time of Recover-Max-Matching(G, D) is dominated by the construction of K. Since H^{ab} has at most $2n$ nodes and at most $3m$ edges, K can be constructed in $O(\sqrt{n}m/k(n, m))$ time using the Feder–Motwani algorithm [4].

It remains to show that K^a is a maximum weight matching of G. First, we argue that H^{ab} has a perfect matching. Let M be a maximum weight matching of G. By



(a) $\phi(G)$ (b) $\phi(G)|C_h$

FIG. 2. (a) *The unfolded graph* $\phi(G)$ *of the bipartite graph given in Figure* 1(a). (b) *With respect to the cover* C_h *defined in Figure* 1(c)*, the node* y_1 *in* $\phi(G)$ *is the only node satisfying the condition that* $1 \le C_h(y)$*. Thus,* $\phi(G)|C_h$ *comprises only the edges incident to* y_1*.*

Fact 2.1, $D(u) + D(v) = w(u, v)$ for every edge $uv \in M$. Therefore, M is also a matching of H. Let U be the set of nodes in H unmatched by M. By Fact 2.1, $D(u) = 0$ for all $u \in U$. Let Q be $\{u^a u^b \mid u \in U\}$. Let $M^a = \{u^a v^a \mid uv \in M\}$ and $M^b = \{u^b v^b \mid uv \in M\}$. Note that $Q \cup M^a \cup M^b$ forms a matching in H^{ab} and every node in H^{ab} is matched by either Q, M^a, or M^b. Thus, H^{ab} has a perfect matching.

Since K is a maximum cardinality matching of H^{ab}, K must be a perfect matching. For every node u with $D(u) > 0$, u^a must be matched by K. Since there is no edge between u^a and any x^b in H^{ab}, there exists some v^a with $u^a v^a \in K$. Thus, every node u with $D(u) > 0$ must be matched by some edge in K^a. Therefore, $\sum_{uv \in K^a} w(u, v) = \sum_{u \in X \cup Y, D(u)>0} D(u) = \sum_{u \in X \cup Y} D(u) = \text{mwm}(G)$, and K^a is a maximum weight matching of G. $\square$

4. All-cavity maximum weight matchings.

In section 4.1, we introduce the notion of an *unfolded graph*. In section 4.2, we use this notion to design an algorithm which, given a weighted bipartite graph G and a maximum weight matching of G, computes $\text{mwm}(G - \{u\})$ for all nodes u in G using $O(W)$ time.

4.1. Unfolded graphs.

The *unfolded graph* $\phi(G)$ of G is defined as follows.

- For each node u of G, $\phi(G)$ has α copies of u, denoted as $u^1, u^2, \ldots, u^\alpha$, where α is the weight of the heaviest edge incident to u.
- For each edge uv of G, $\phi(G)$ has the edges $u^1 v^\beta, u^2 v^{\beta-1}, \ldots, u^\beta v^1$, where $\beta = w(u, v)$.

See Figure 2(a) for an example. Let M be a matching of G. Consider M as a weighted bipartite graph; then, by definition, $\phi(M) = \bigcup_{uv \in M} \{u^1 v^\beta, \ldots, u^\beta v^1 \mid \beta = w(u, v)\}$ is a matching of $\phi(G)$. The number of edges in $\phi(M)$ is equal to the total weight of the edges in M, i.e., $|\phi(M)| = \sum_{uv \in M} w(u, v)$. The next lemma relates G and $\phi(G)$.

LEMMA 4.1. *Assume that M is a maximum weight matching of G.*

1. $\text{mwm}(G) = \text{mm}(\phi(G))$.
2. *The set $\phi(M)$ is a maximum cardinality matching of $\phi(G)$.*

Proof. Statement 4.1 follows from Statement 4.1. Statement 4.1 is proved as follows. Since M is a maximum weight matching of G, $\text{mwm}(G) = \sum_{uv \in M} w(u, v) = |\phi(M)| \leq \text{mm}(\phi(G))$. By Fact 2.1, $\text{mwm}(G) \geq \text{mm}(\phi(G))$ if and only if $\text{mwc}(G) \geq \text{mwc}(\phi(G))$. We prove the latter as follows. Given a minimum weight cover C of G, we can obtain a cover C' of $\phi(G)$ as follows. For any node u of G, $C'(u^i) = 1$ if $C(u) > 0$ and $i \leq C(u)$; otherwise, $C'(u^i) = 0$. Note that $w(C') = w(C) = \text{mwc}(G)$. Therefore, $\text{mwc}(G) \geq \text{mwc}(\phi(G))$ and $\text{mwm}(G) \geq \text{mm}(\phi(G))$. ☐

4.2. An algorithm for all-cavity maximum weight matchings. Let M be a given maximum weight matching of G.

By Lemma 4.1(2), $\phi(M)$ is a maximum cardinality matching of $\phi(G)$. In light of this maximality, we say that a path in $\phi(G)$ is *alternating* for $\phi(M)$ if (1) its edges alternate between being in $\phi(M)$ and being not in $\phi(M)$ and (2) in the case the first (respectively, last) node is matched by $\phi(M)$, the path contains the matched edge of u as the first (respectively, last) edge. The length of an alternating path is its number of edges. An alternating path may have zero length; in this case, the path contains exactly one unmatched node. An alternating path P can modify $\phi(M)$ to another matching, i.e., $(\phi(M) \cup P) - (\phi(M) \cap P)$. If P is of even length, the resulting matching has the same size as $\phi(M)$. If P is of odd length, P modifies M to a strictly smaller or bigger matching; yet the latter is impossible because $\phi(M)$ is maximum. Intuitively, we would like to maximize the size of the resultant matching and even-length alternating paths are preferred.

Our new algorithm for computing $\text{mwm}(G - \{u\})$ is based on the observation that $\text{mwm}(G - \{u\})$ can be determined by detecting the smallest i such that u^i has an even-length alternating path for $\phi(M)$. Details are as follows.

Definition. For each u^i in $\phi(G)$, let $\rho(u^i) = 0$ if there is an even-length alternating path for $\phi(M)$ starting from u^i; otherwise, let $\rho(u^i) = 1$.

The following lemma states a monotone property of $\rho(u^i)$ over different i's.

LEMMA 4.2. *Consider any node u in G. Let $u^1, u^2, \ldots, u^\beta$ be its corresponding nodes in $\phi(G)$. If $\rho(u^i) = 0$, then $\rho(u^j) = 0$ for all $j \in [i, \beta]$. Furthermore, there exist $\beta - i + 1$ node-disjoint even-length alternating paths $P_i, P_{i+1}, \ldots, P_\beta$ for $\phi(M)$, where each P_j starts from u^j.*

Proof. As $\rho(u^i) = 0$, let $P_i = u_0^{a_0}, v_0^{b_0}, u_1^{a_1}, v_1^{b_1}, \ldots, u_{p-1}^{a_{p-1}}, v_{p-1}^{b_{p-1}}, u_p^{a_p}$ be a shortest even-length alternating path for $\phi(M)$, where $u_0^{a_0} = u^i$.

Based on P_i, we can construct an even-length alternating path P_{i+1} for $\phi(M)$ starting from u^{i+1} as follows. If u^{i+1} is not matched by $\phi(M)$, P_{i+1} is simply a path of zero length. From now on, we assume that u^{i+1} is matched by $\phi(M)$. As P is of even length, $u_p^{a_p}$ is not matched by $\phi(M)$. Then, by the definition of $\phi(M)$, $u_p^{a_p+1}$ is also not matched by $\phi(M)$. Let h be the smallest integer in $[1, p]$ such that $u_h^{a_h+1}$ is not matched by $\phi(M)$. Notice that, for all $\ell < h$, $u_\ell^{a_\ell+1}$ is matched to $v_\ell^{b_\ell-1}$; furthermore, $\phi(G)$ contains an edge between $v_\ell^{b_\ell-1}$ and $u_{\ell+1}^{a_{\ell+1}+1}$. Thus, $P_{i+1} = u^{i+1}, v_0^{b_0-1}, u_1^{a_1+1}, v_1^{b_1-1}, \ldots, u_h^{a_h+1}$ is an even-length alternating path for $\phi(M)$. Similarly, for $j = i + 2, \ldots, \beta$, we can use P_i to define an even-length alternating path P_j for $\phi(M)$ starting from u^j. By construction, $P_i, P_{i+1}, \ldots, P_\beta$ are node-disjoint. ☐

The next lemma is the basis of our cavity matching algorithm. It shows that given $\text{mwm}(G)$ (i.e., the weight of M), we can compute $\text{mwm}(G - \{u\})$ from the values $\rho(u^i)$, and all the $\rho(u^i)$'s can be found in $O(W)$ time.

LEMMA 4.3.

1. $\sum_{1 \leq i \leq \beta} \rho(u^i) = \text{mwm}(G) - \text{mwm}(G - \{u\})$.
2. *For all* $u^i \in \phi(G)$, $\rho(u^i)$ *can be computed in* $O(W)$ *time in total.*

Proof. The two statements are proved as follows.

Statement 1. Let k be the largest integer such that $\rho(u^k) = 1$. By Lemma 4.2, $\rho(u^i) = 1$ for all $1 \leq i \leq k$, and 0 otherwise. Note that if $\rho(u^i) = 1$, u^i must be matched by $\phi(M)$. Thus, $\sum_{1 \leq i \leq \beta} \rho(u^i) = k$. Below, we prove the following two equalities:

(1) $\text{mm}(\phi(G) - \{u^1, \ldots, u^k\}) = \text{mm}(\phi(G)) - k$.

(2) $\text{mm}(\phi(G) - \{u^1, \ldots, u^\beta\}) = \text{mm}(\phi(G) - \{u^1, \ldots, u^k\})$.

Then, by Lemma 4.1, $\text{mwm}(G) = \text{mm}(\phi(G))$ and $\text{mwm}(G - \{u\}) = \text{mm}(\phi(G) - \{u^1, \ldots, u^\beta\})$. Thus, $\text{mwm}(G) - \text{mwm}(G - \{u\}) = k$ and Statement 1 follows.

To show equality (1), let H be the set of edges of $\phi(M)$ incident to u^i with $1 \leq i \leq k$. Let $M' = \phi(M) - H$. Then, $|M'| = |\phi(M)| - k$. We claim that M' is a maximum cardinality matching of $\phi(G) - \{u^1, \ldots, u^k\}$. Hence, $\text{mwm}(\phi(G) - \{u^1, \ldots, u^k\}) = |\phi(M)| - k$, and equality (1) follows. We prove the claim by contradiction. Suppose M' is not a maximum cardinality matching of $\phi(G) - \{u^1, \ldots, u^k\}$. Then, there exists an alternating path P that can modify M' to a larger matching of $\phi(G) - \{u^1, \ldots, u^k\}$ [8, 9]; in particular, the length of P must be odd and both of its endpoints are not matched by M'. P must start from some node v^j with $u^i v^j \in \phi(M)$ and $i < k$; otherwise, P is alternating for $\phi(M)$ in G and $\phi(M)$ cannot be a maximum cardinality matching of $\phi(G)$. Let Q be a path formed by joining $u^i v^j$ with P. Q is an even-length alternating path for $\phi(M)$ starting from u^i in $\phi(G)$. This contradicts the fact that there is no even-length alternating path for $\phi(M)$ starting from u^i for $i < k$.

To show equality (2), we first note that $\text{mm}(\phi(G) - \{u^1, \ldots, u^\beta\}) \leq \text{mm}(\phi(G) - \{u^1, \ldots, u^k\})$. It remains to prove the other direction. By Lemma 4.2, we can find $\beta - k$ node-disjoint even-length alternating paths $P_{k+1}, \ldots, P_\beta$ for $\phi(M)$, which start from $u^{k+1}, \ldots, u^\beta$. P_j starts at u^j. Let $M'' = (\phi(M) \cup (P_{j+1} \cup \cdots \cup P_\beta)) - (\phi(M) \cap (P_{j+1} \cup \cdots \cup P_\beta))$. Note that $|M''| = |\phi(M)|$ and there are no edges in M'' incident to any of $u^{k+1}, \ldots, u^\beta$. M'' is a matching of $\phi(G) - \{u^{k+1}, \ldots, u^\beta\}$ and $M'' - H$ of $\phi(G) - \{u^1, \ldots, u^\beta\}$. $|M'' - H| \geq | M''| - k = |\phi(M)| - k$. Since $\text{mm}(\phi(G) - \{u^1, \ldots, u^k\}) = |\phi(M)| - k$ by equality (1), it follows that $\text{mm}(\phi(G) - \{u^1, \ldots, u^\beta\}) \geq |M'' - H| \geq \text{mm}(\phi(G) - \{u^1, \ldots, u^k\})$. Therefore, equality (2) holds.

Statement 2. We want to determine whether $\rho(u^i) = 0$ for all nodes $u^i \in \phi(G)$ in $O(W)$ time. By definition, $\rho(u^i) = 0$ if and only if there is an even-length alternating path for $\phi(M)$ starting from u^i. Let us partition the nodes of $\phi(G)$ into two parts: $\phi(X) = \{u^i \in \phi(G) \mid u \in X\}$ and $\phi(Y) = \{u^i \in \phi(G) \mid u \in Y\}$. Below, we give the details of computing $\rho(u^i)$ for all $u^i \in \phi(X)$. The case where $u^i \in \phi(Y)$ is symmetric.

Let D be a directed graph over the node set $\phi(X)$. D contains an edge $u^i v^j$ if there exists a node $w^k \in \phi(Y)$ such that $u^i w^k \in \phi(G) - \phi(M)$ and $w^k v^j \in \phi(M)$. Consider any node v^j of D that is unmatched by $\phi(M)$. A directed path in D from v^j to a node u^i corresponds to a path in $\phi(G)$, which is indeed an even-length alternating path for $\phi(M)$ starting from u^i. Therefore, for any $u^i \in \phi(X)$, $\rho(u^i) = 0$ if and only if u^i is reachable from some node in D that is unmatched by $\phi(M)$. We can identify all such u^i by using a depth-first search on D starting with all the nodes unmatched by M. The time required is $O(|D|)$. As $|D| \leq | \phi(G)| = W$, the lemma follows. □

The following procedure computes $\text{mwm}(G - \{u\})$ for all nodes u of G. Let M be a maximum weight matching of G.

```
PROCEDURE Compute-All-Cavity($(G, M)$).
  1. Construct $\phi(G)$ and $\phi(M)$.
  2. For every $j \in [0, n/2]$, determine $A_j$ from $\phi(M)$.
  3. For every node $u^i$ of $\phi(G)$, if $u^i \in \bigcup_j A_j$ then $\rho(u^i) = 0$; otherwise $\rho(u^i) = 1$.
  4. For every node $u$ of $G$, compute $\mathrm{mwm}(G - \{u\}) = \mathrm{mwm}(G) - \sum_{1 \le i \le \beta} \rho(u^i)$,
     where $u^1, u^2, \ldots, u^\beta$ are the nodes corresponding to $u$ in $\phi(G)$.
```

THEOREM 4.4. Compute-All-Cavity(G, M) *correctly computes* $\mathrm{mwm}(G - \{u\})$ *for all* u *of* G *in* $O(W)$ *time.*

Proof. The proof follows from Lemma 4.3 □

Acknowledgments. The authors wish to thank the anonymous referee for extremely helpful comments, which significantly improved the presentation of the paper. In particular, Theorem 2.2 was originally proved using unfolded graphs (see the conference version of this paper [13]); the new proof is based on a suggestion by the referee.

REFERENCES

[1] J. BONDY AND U. MURTY, *Graph Theory with Applications*, North-Holland, New York, 1976.

[2] M. J. CHUNG, $O(n^{2.5})$ *time algorithms for the subgraph homeomorphism problem on trees*, J. Algorithms, 8 (1987), pp. 106–112.

[3] T. H. CORMEN, C. L. LEISERSON, AND R. L. RIVEST, *Introduction to Algorithms*, MIT Press, Cambridge, MA, 1990.

[4] T. FEDER AND R. MOTWANI, *Clique partitions, graph compression and speeding-up algorithms*, J. Comput. System Sci., 51 (1995), pp. 261–272.

[5] M. L. FREDMAN AND R. E. TARJAN, *Fibonacci heaps and their uses in improved network optimization algorithms*, J. ACM, 34 (1987), pp. 596–615.

[6] H. N. GABOW, *Scaling algorithms for network problems*, J. Comput. System Sci., 31 (1985), pp. 148–168.

[7] H. N. GABOW AND R. E. TARJAN, *Faster scaling algorithms for network problems*, SIAM J. Comput., 18 (1989), pp. 1013–1036.

[8] Z. GALIL, *Efficient algorithms for finding maximum matching in graphs*, ACM Computing Surveys, 18 (1986), pp. 23–38.

[9] A. M. H. GERARDS, *Matching*, in Handbooks in Operations Reserach and Management Science 7: Network Models, M. O. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser, eds., North-Holland, Amsterdam, 1995, pp. 135–224.

[10] G. H. HARDY, J. E. LITTLEWOOD, AND G. PÓLYA, *Inequalities*, Cambridge University Press, Cambridge, UK, 1988. Reprint of the 1952 edition.

[11] J. E. HOPCROFT AND R. M. KARP, *An* $n^{5/2}$ *algorithm for maximum matchings in bipartite graphs*, SIAM J. Comput., 2 (1973), pp. 225–231.

[12] M. Y. KAO, T. W. LAM, W. K. SUNG, AND H. F. TING, *All-cavity maximum matchings*, in Proceedings of the 8th Annual International Symposium on Algorithms and Computation, Lecture Notes Comput. Sci. 1350, H. Imai and H. W. Leong, eds., Springer-Verlag, New York, NY, 1997, pp. 364–373.

[13] M. Y. KAO, T. W. LAM, W. K. SUNG, AND H. F. TING, *A decomposition theorem for maximum weight bipartite matchings with applications to evolutionary trees*, in Proceedings of the 7th Annual European Symposium on Algorithms, Lecture Notes in Comput. Sci. 1643, J. Nešetřil, ed., Springer-Verlag, New York, NY, 1999, pp. 438–449.

[14] M. Y. KAO, T. W. LAM, W. K. SUNG, AND H. F. TING, *Cavity matchings, label compressions, and unrooted evolutionary trees*, SIAM J. Comput. 30 (2000), pp. 602–624.

[15] H. W. KUHN, *The Hungarian method for the assignment problem*, Naval Res. Logist. Quarterly, 2 (1955), pp. 83–97.