Algorithmica (2022) 84:1952–1992
https://doi.org/10.1007/s00453-022-00949-5

Approximate Generalized Matching: *f*-Matchings and *f*-Edge Covers

Dawei Huang[1] · **Seth Pettie**[1]

Received: 11 May 2020 / Accepted: 26 January 2022 / Published online: 25 February 2022

Abstract
We present almost linear time approximation schemes for several generalized matching problems on nonbipartite graphs. Our results include $O_\epsilon(m\alpha(m,n))$-time algorithms for $(1-\epsilon)$-maximum weight f-matching and $(1+\epsilon)$-approximate minimum weight f-edge cover. As a byproduct, we also obtain direct algorithms for the exact cardinality versions of these problems running in $O(m\alpha(m,n)\sqrt{f(V)})$ time, where $f(V)$ is the sum of degree constraint on the entire vertex set. The technical contributions of this work include an efficient method for maintaining *relaxed complementary slackness* in generalized matching problems and approximation-preserving reductions between the f-matching and f-edge cover problems.

Keywords Matching · f-Factors · Edge covers

1 Introduction

Many combinatorial optimization problems are known to be reducible to computing optimal matchings in non-bipartite graphs [6,7]. These problems include computing b-matchings, f-factors, f-edge covers, T-joins, undirected shortest paths (with no negative cycles), and bidirected flows; see [8,12,19,23]. These problems have been investigated heavily since Tutte's work in the 1950s [22,25]. However, the existing reductions to graph matching are often inadequate: they blow up the size of the input [19], use auxiliary space [10], or piggyback on specific matching algorithms [10] like the Micali-Vazirani algorithm [20,26,27]. Moreover, some existing reductions destroy the dual structure of optimal solutions and are therefore not *approximation preserving*.

✉ Dawei Huang
dwhuang9@gmail.com

✉ Seth Pettie
pettie@umich.edu

1 EECS Department, University of Michigan, 2260 Hayward St., Ann Arbor, MI 48109, USA

In this paper we design algorithms for computing f-matchings and f-edge covers (both defined below) in a direct fashion, or through efficient, approximation-preserving reductions. Because our algorithms are based on the LP formulations of these problems (in contrast to approaches using *shortest* augmenting walks [10,20,26,27]), they easily adapt to *weighted* and *approximate* variants of the problems. Let us define these problems formally. Let $G = (V, E)$ be a graph that possibly contains parallel edges and self loops. For any subset F of edges, we use $\deg_F(v)$ to indicate the degree of vertex v in the subgraph induced by F. Notice that each self-loop on v that is in F contributes 2 to this degree. We use n to denote the number of vertices and m to denote the number of edges, counting multiplicities. We define f-matching and f-edge covers as follows.

- f-matching: An f-matching is a subset $F \subseteq E$ such that $\deg_F(v) \le f(v)$. F is *perfect* if the degree constraints hold with equality. In this case it is also called an *f-factor*.
- f-edge cover: An f-edge cover is a subset $F \subseteq E$ such that $\deg_F(v) \ge f(v)$. It is *perfect* if all degree constraints hold with equality.

The *maximum weight f-matching problem* asks, given a graph $G = (V, E)$ and a weight function w on E, for an f-matching F that maximizes $\sum_{e \in F} w(e)$. Similarly, the *minimum weight f-edge cover problem* and the *minimum weight f-factor problem* ask for an f-edge cover and an f-factor that minimize their respective weight.

For these three problems, we can assume, without loss of generality, that all the weights are nonnegative, but for different reasons. For maximum weight f-matching, it is safe to ignore any negative weight edges as discarding negative weight edges from F can only improve the solution. For minimum weight f-edge cover, any optimum solution must include the set of *all* negative weight edges. Hence we can include them into the solution and update the degree constraint accordingly. For minimum weight f-factor, since all f-factors are of the same size, we can translate all the weights by $-\min_{e \in E} w(e)$ without changing the optimal solution so that the resulting graph has only nonnegative weights.

Classic Reductions

The classical reduction from f-matching to standard graph matching uses the *b-matching* problem as a stepping stone. A b-matching is a function $x : E \to \mathbb{Z}_{\ge 0}$ (where $x(e)$ indicates *how many* copies of e are in the matching) such that $\sum_{e \in \delta(v)} x(e) \le b(v)$, i.e., the number of matched edges incident to v, counting multiplicity, is at most $b(v)$. The maximum weight f-matching problem on $G = (V, E, w)$ can be reduced to b-matching by subdividing each edge $e = (u, v) \in E$ into a path (u, u_e, v_e, v). Here u_e, v_e are new vertices. We set the weight of the new edges to be $w(u, u_e) = w(v_e, v) = w(u, v) + W$ and $w(u_e, v_e) = 2W$, where W is the maximum weight in the original graph. The capacity function b is given by $b(u_e) = b(v_e) = 1$ for the new vertices and $b(u) = f(u)$ for the original vertices.

To see this reduction correctly reduces the maximum weight f-matching problem to the maximum weight b-matching problem, we first notice that if the original graph has an f-matching M of weight W^*, then the new graph must contain a b-matching of weight at least $2W^* + 2Wm$, where m is the number of edges in the original graph: for every unmatched edge (u, v), we take the edge (u_e, v_e) into the b-matching and leave the two edges (u, u_e) and (v, v_e) unmatched. Otherwise we take only the edges (u, u_e) and (v, v_e). For the other direction, we notice that if M is a maximum weight b-matching in the new graph, for each subdivision of original edge (u, v), we can assume without loss of generality that M' must either take only the middle edge (u_e, v_e), or the two side edges (u, u_e) and (v, v_e): By the degree constraints on u_e and v_e, taking the middle edge will prevent the b-matching from taking any of the side edges, and vice versa. Moreover, we cannot take only one of the side edges since in that case we can swap it for the middle edge without decreasing the total weight. Therefore, a b-matching of weight $2W^* + 2mW$ must also correspond to an f-matching of weight W^* of the original graph.

This reduction blows up the number of vertices to $O(m)$ and is not *approximation preserving*. The b-matching problem is easily reduced to standard matching by replicating each vertex u $b(u)$ times, and replacing each edge (u, v) with a bipartite $b(u)\times b(v)$ clique on its endpoints' replicas. This step of the reduction is approximation preserving, but blows up the number of vertices and edges. Both reductions together reduce f-matching to a graph matching problem on $O(m)$ vertices and $O(f_{\max} m)$ edges. Gabow [10] gave a method for solving f-matching in $O(m\sqrt{f(V)})$ time using black-box calls to single iterations of the Micali-Vazirani [20,26,27] algorithm.

Observe that f-matching and f-edge cover are *complementary* problems: if C is an f_C-edge cover, the complementary edge set $F = E\backslash C$ is necessarily an f_F-matching, where $f_F(v) = \deg(v) - f_C(v)$. Complementarity implies that any polynomial-time algorithm for one problem solves the other in polynomial time, but it says nothing about the precise complexity of solving them exactly or approximately. Indeed, this phenomenon is very well known in the realm of NP-complete problems. For example, Maximum Independent Set and Minimum Vertex Cover are complementary problems, but have completely different approximation profiles: Minimum Vertex Cover has a well-known polynomial time 2-approximation algorithm, while it is NP-hard to approximate Maximum Independent Set within $n^{1-\epsilon}$ for any $\epsilon > 0$ [16,28]. Gabow's $O(m\sqrt{f_F(V)})$ cardinality f_F-matching algorithm [10] implies that f_C-edge cover is computed in $O(m\sqrt{2m - f_C(V)}) = O(m^{3/2})$ time, and says nothing about the approximability of f_C-edge cover. As far as we are aware, the fastest approximation algorithms for f_C-edge cover (see [17]) treat it as a general weighted Set Cover problem on 2-element sets. Chvátal's analysis [2] shows the greedy algorithm is an $H(2)$-approximation, where $H(2) = 3/2$ is the 2nd harmonic number.

Our interest in the *approximate* f-edge cover problem is inspired by a new application to anonymizing data in environments where users have different privacy demands; see [1,17,18]. Here the data records correspond to edges and the privacy demand of v is measured by $f(v)$; the goal is to anonymize as few records to satisfy everyone's privacy demands.

New Results

We give new algorithms for computing f-matchings and f-edge covers approximately and exactly.

- We give an $O_\epsilon(m\alpha(m, n)$-time $(1-\epsilon)$-approximation algorithm for maximum weight f-matching problem. The algorithm generalizes the $(1-\epsilon)$-approximate maximum weight matching algorithm by Duan and Pettie [4] and improves on the

$O(f(V)(m + n \log n))$ running time of Gabow [11], which computes an exactly optimum solution. The main technical contribution is the application of *relaxed complementary slackness* [4,14,15] on f-matchings, and a new version of DFS-based search procedure algorithm for looking for a *maximal* set of *edge-disjoint* augmenting paths in $O(m\alpha(m, n))$ time.
- We show that a folklore reduction from minimum weight 1-edge cover to maximum weight 1-matching (matching) is approximation-preserving, in the sense that any $(1 - \epsilon)$-approximation for matching gives a $(1 + \epsilon)$-approximation for edge cover. This implies that 1-edge cover can be $(1 + \epsilon)$-approximated in $O_\epsilon(m)$ time [4], and that one can apply any number of simple and practical algorithms [3,4,21] to approximate 1-edge cover. This simple reduction does *not* extend to f-matchings/f-edge covers when f is arbitrary.
- We give an $O_\epsilon(m\alpha(m, n))$-time $(1 + \epsilon)$-approximation algorithm for weighted f_C-edge cover, for any f_C. Our algorithm follows from two results, both of which are somewhat surprising. First, any approximate weighted f_F-matching algorithm *that reports a* $(1 \pm \epsilon)$-*optimal dual solution* can be transformed into a $(1 + O(\epsilon))$-approximate weighted f_C-edge cover algorithm. Second, such an f_F-matching algorithm exists, and its running time is $O_\epsilon(m\alpha(m, n))$. The first claim is clearly false if we drop the approximate dual solution requirement (for the same reason that an $O(1)$-approximate vertex cover does not translate into an $O(1)$-approximate maximum independent set), and the second is surprising because the running time is independent of the demand function f_F and the magnitude of the edge weights.
- As corollaries of these reductions, we obtain a new exact algorithm for minimum cardinality f_C-edge cover running in $O(m\alpha(m, n)\sqrt{f_C(V)})$ time, rather than $O(m^{3/2})$ time ([10]), and a direct algorithm for cardinality f_F-matching that runs in $O(m\alpha(m, n)\sqrt{f_F(V)})$ time, without reduction [10] to the Micali-Vazirani algorithm [20,26,27].

The blossom structure and LP characterization of b-matching is considerably simpler than the corresponding blossoms/LPs for f-matching and f-edge cover. In the interest of simplicity, one might want efficient code that solves (approximate) b-matching directly, without viewing it as a special case of the f-matching problem.[1] We do not know of such a direct algorithm. Indeed, the structure of b-matching blossoms seems to rely on strict complementary slackness, and is *incompatible* with our main technical tool, relaxed complementary slackness.[2] Thus, for somewhat technical reasons, we are forced to solve approximation b-matching using more sophisticated f-matching tools.

Comparison to Previous Results

Our almost linear time $(1 - \epsilon)$-approximation algorithm for maximum weight f-matching can be seen as a direct generalization of the Duan-Pettie algorithm for

[1] The b-matching problem can be regarded as an f-matching problem on a multigraph in which there is implicitly an infinite supply of each edge.

[2] Using relaxed complementary slackness, matched and unmatched edges have *different* eligibility criteria (to be included in augmenting paths and blossoms) whereas b-matching blossoms require that all copies of an edge—matched and unmatched alike—are all eligible or all ineligible.

approximate maximum weight matching (1-matching) [4]. The key technical ingredient is the generalization of *relaxed complementary slackness*, see [4,14,15], to f-matching, and a corresponding implementation of Edmonds' Search with relaxed complementary slackness. The former relies heavily on the ideas (blossoms, augmenting walks) defined in [11]. Our implementation of Edmonds' search involves finding augmenting walks in batches. The procedure of [15, Sect. 8] for matching finds a maximal set of vertex-disjoint augmenting paths. We develop a corresponding procedure that finds a maximal set of edge-disjoint augmenting walks *and* cycles. Including alternating cycles in the output allows us to conduct the search in almost linear time, and keep the search more organized and tree-structured.[3]

Structure of the Paper

In Sect. 2 we give an introduction to the LP-formulation of generalized matching problems and Gabow's formulation [11] for their blossoms and augmenting walks. In Sect. 3.1 we show that a folklore reduction from 1-edge cover to 1-matching is approximation-preserving and in Sect. 3.2 we reduce approximate f-edge cover to approximate f-matching. In Sect. 4 we give an $O(Wm\alpha(m,n)\epsilon^{-1})$-time algorithm for $(1-\epsilon)$-approximate f-matching in graphs with weights in $[0, W]$ and then speed it up to $O(m\alpha(m,n)\epsilon^{-1}\log\epsilon^{-1})$, independent of the weight function. Sect. 5 gives $O(m\alpha(m,n))$ algorithm to compute a maximal set of augmenting walks and alternating cycles; cf. [15, Sect. 8].

2 Basis of *f*-Matching and *f*-Edge Cover

This section reviews basic algorithmic concepts from matching theory and their generalizations to the f-matching and f-edge cover problems, e.g., LPs, blossoms, and augmenting walks. These ideas lay the foundation for generalizing the Duan-Pettie algorithm [4] for Approximate Maximum Weight Matching to Approximate Maximum Weight f-Matching and Approximate Minimum Weight f-Edge Cover.

Notation

The input is a multigraph $G = (V, E)$ with a *nonnegative* weight function $w : E \mapsto \mathbb{R}_{\geq 0}$. For any vertex v, define $\delta(v)$ and $\delta_0(v)$ be the set of non-loop edges and self-loops, respectively, incident on v. For $S \subseteq V$, let $\delta(S)$ and $\gamma(S)$ be the sets of edges with exactly one endpoint and both endpoints in S, respectively, so $\delta_0(v) \subseteq \gamma(S)$ if $v \in S$. For $T \subseteq E$, $\delta_T(S)$ denotes the intersection of $\delta(S)$ and T. By definition, $\deg_T(S) = |\delta_T(S)|$.

2.1 LP formulation

The maximum weight f-matching problem can be expressed as maximizing $\sum_{e\in E} w(e)x(e)$, subject to the following constraints:

[3] These issues only arise when finding augmenting paths in batches, not one-at-a-time [11], and when the problem is f-matching, not matching.

$$\sum_{e\in\delta(v)} x(e) + \sum_{e\in\delta_0(v)} 2x(e) \le f(v), \text{ for all } v \in V,$$
$$\sum_{e\in\gamma(B)\cup I} x(e) \le \left\lfloor \frac{f(B)+|I|}{2} \right\rfloor, \text{ for all } B \subseteq V, I \subseteq \delta(B),$$
$$0 \le x(e) \le 1, \text{ for all } e \in E. \tag{1}$$

Here, the blossom constraint $\sum_{e\in\gamma(B)\cup I} x(e) \le \left\lfloor \frac{f(B)+|I|}{2} \right\rfloor$ is a generalization of blossom constraint $\sum_{e\in\gamma(B)} x(e) \le \left\lfloor \frac{|B|}{2} \right\rfloor$ in ordinary matching. The reason that we have a subset I of incident edges in the sum is that the subset allows us to distinguish between matched edges that have both endpoints inside B with those with exactly one endpoint. Any basic feasible solution x of this LP is integral [23, Chapter 33], and can therefore be interpreted as a membership vector of an f-matching F. To certify (approximate) optimality of a solution, the algorithm works with the dual LP, which is:

$$\begin{array}{ll} \text{minimize} & \sum_{v\in V} f(v)y(v) + \sum_{B\subseteq V, I\subseteq\delta(B)} \left\lfloor \frac{f(B)+|I|}{2} \right\rfloor z(B,I) + \sum_e u(e), \\ \text{subject to} & yz_F(e) + u(e) \ge w(e), \text{ for all } e \in E, \\ & y(v) \ge 0, z(B,I) \ge 0, u(e) \ge 0. \end{array} \tag{2}$$

Here the aggregated dual $yz_F : E \mapsto \mathbb{R}_{\ge 0}$ is defined as:

$$yz_F(u,v) = y(u) + y(v) + \sum_{\substack{B,I:(u,v)\in\gamma(B)\cup I,\\ I\subseteq\delta(B))}} z(B,I).$$

Notice that u can be equal to v when the edge is a self-loop. Unlike matching, each z-value here is associated with the combination of a vertex set B and a subset I of its incident edges.

The minimum weight f-edge cover problem can be expressed as minimizing $\sum_{e\in E} w(e)x(e)$, subject to:

$$\sum_{e\in\delta(v)} x(e) + \sum_{e\in\delta_0(v)} 2x(e) \ge f(v), \text{ for all } v \in V,$$
$$\sum_{e\in\gamma(B)\cup(\delta(B)\setminus I)} x(e) \ge \left\lceil \frac{f(B)-|I|}{2} \right\rceil, \text{ for all } B \subseteq V \text{ and } I \subseteq \delta(B),$$
$$0 \le x(e) \le 1, \text{ for all } e \in E. \tag{3}$$

With the dual program being:

$$\begin{aligned} \text{maximize} \quad & \sum_{v\in V} f(v)y(v) + \sum_{B\subseteq V, I\subseteq\delta(B)} \left\lceil \frac{f(B)-|I|}{2} \right\rceil z(B,I) - \sum_{e\in E} u(e), \\ \text{subject to} \quad & yz_C(e) - u(e) \le w(e), \ \text{ for all } e \in E, \\ & y(v) \ge 0, z(B,I) \ge 0, u(e) \ge 0, \end{aligned} \tag{4}$$

where

$$yz_C(u,v) = y(u) + y(v) + \sum_{\substack{B,I:(u,v)\in\gamma(B)\cup(\delta(B)\setminus I) \\ I\subseteq\delta(B)}} z(B,I).$$

Both of our f-matching and f-edge cover algorithms maintain a dynamic *feasible* solution $F \subseteq E$ that satisfies the primal constraints[4] following Gabow [11]. We call edges in F *matched* and all other edges *unmatched*, which is referred to as the *type* of an edge. A vertex v is *saturated* if $\deg_F(v) = f(v)$. It is *unsaturated/oversaturated* if $\deg_F(v)$ is smaller/greater than $f(v)$. Given an f-matching F, the *deficiency* $\mathrm{def}(v)$ of a vertex v is defined as $\mathrm{def}(v) = f(v) - \deg_F(v)$. Similarly, for an f-edge cover C, the *surplus* of a vertex is defined as $\mathrm{surp}(v) = \deg_C(v) - f(v)$.

2.2 Blossoms

We follow Gabow's [11] definitions and terminology for f-matching blossoms, augmenting walks, etc. A *blossom* is a tuple $(B, E_B, \beta(B), \eta(B))$ where B is the vertex set, E_B is the edge set, $\beta(B) \in B$ is the *base vertex*, and $\eta(B) \subset \delta(\beta(B)) \cap \delta(B)$, $|\eta(B)| \le 1$, is the *base edge set*, which may be empty. We often refer to the blossom by referring to its vertex set B. Blossoms can be defined inductively as follows.

Definition 1 (See [11, Definition 4.2]) A single vertex v forms a *trivial blossom*, or a *singleton*. Here $B = \{v\}$, $E_B = \emptyset$, $\beta(B) = v$, and $\eta(B) = \emptyset$.

Inductively, let $B_0, B_1, \ldots, B_{l-1}$ be a sequence of disjoint singletons or nontrivial blossoms. Suppose there exists a closed walk $C_B = \{e_0, e_1, \ldots, e_{l-1}\} \subseteq E$ starting and ending with B_0 such that $e_i \in B_i \times B_{i+1 \pmod l}$. The vertex set $B = \bigcup_{i=0}^{l-1} B_i$ is identified with a blossom if the following are satisfied:

1. *Base Requirement:* If B_0 is a singleton, the two edges incident to B_0 on C_B, i.e., e_0 and e_{l-1}, must both be matched or both be unmatched.
2. *Alternation Requirement:* Fix a B_i, $i \ne 0$. If B_i is a singleton, exactly one of e_{i-1} and e_i is matched. If B_i is a nontrivial blossom, then $\eta(B_i) \ne \emptyset$ and must be either $\{e_{i-1}\}$ or $\{e_i\}$.

The edge set of the blossom B is $E_B = C_B \cup (\bigcup_{i=0}^{l-1} E_{B_i})$ and its base is $\beta(B) = \beta(B_0)$. If B_0 is not a singleton, $\eta(B) = \eta(B_0)$. If B_0 is a singleton, $\eta(B)$ may either be empty or contain one edge, which is in $\delta(B) \cap \delta(B_0)$ that is the opposite type of e_0 and e_{l-1}.

[4] We use yz_F and yz_C to denote the aggregated dual yz for f-matching and f-edge cover respectively. We will omit the subscript if it is clear from the context.

Blossoms are classified as *light/heavy* [11, p. 32]. If B_0 is a singleton, B is light/heavy if e_0 and e_{l-1} are both unmatched/matched. Otherwise, B is light/heavy if B_0 is light/heavy. Note that blossoms in the ordinary matching problem (1-matching) are always light, since no vertex is adjacent to 2 matched edges.

One purpose of blossoms is to identify parts of graph that can be contracted and treated *similar* to individual vertices when searching for augmenting walks. This is formalized by Lemma 1, which can be seen as a restatement of Lemma 4.4 from [11] for f-matchings.

Lemma 1 *Let v be an arbitrary vertex in B. There exists an even length alternating walk $P_0(v)$ (whose length could be 0) and an odd length alternating walk $P_1(v)$ from $\beta(B)$ to v using edges in E_B. Moreover, the terminal edge incident to $\beta(B)$, if it exists, must have a different type than the edge in $\eta(B)$, if any. In other words, this edge must be matched if B is heavy and unmatched if B is light.*

Proof We prove this by induction. The base case is a blossom B consisting of singletons $\langle v_0, v_1, \ldots, v_{l-1}\rangle$, where $v = v_i$ for some $0 \leq i < l$. Then one of the two walks $\langle v_0, v_1, \ldots, v_i\rangle$ and $\langle v_0, v_{l-1}, v_{l-2} \ldots, v_i\rangle$ must be odd and the other must be even.

Now for the inductive step: Consider the cycle $C_B = \langle B_0, e_0, B_1, \ldots, e_{l-2}, B_{l-1}$ $e_{l-1}, B_0\rangle$ where B_i's are singletons or contracted blossoms. Suppose the claim holds inductively for all nontrivial blossoms in $B_0, B_1, \ldots, B_{l-1}$. Let v be an arbitrary vertex in B. We use $P_{B_i,j}(u)$ $(0 \leq i < l, j \in \{0, 1\}, u \in B_i)$ to denote the walk $P_0(u)$ and $P_1(u)$ guaranteed in blossom B_i. There are two cases:

Case *1:* When v is contained in a singleton B_k. We examine the two walks $\widehat{P} = \langle B_0, e_0, B_1, e_1, \ldots, e_{k-1}, B_k\rangle$ and $\widehat{P}' = \langle B_0, e_{l-1}, B_{l-1}, e_{l-2}, \ldots, e_k, B_k\rangle$. Notice that $\widehat{P}$ and $\widehat{P}'$ are walks in the graph obtained by contracting all subblossoms $B_0, B_1, \ldots, B_{l-1}$ of B. By the inductive hypothesis, we can extend $\widehat{P}$ and $\widehat{P}'$ to P and P' in the original graph G by replacing each B_i with the walk in the original graph connecting the endpoints of e_{i-1} and e_i of the appropriate parity. In particular, if e_{i-1} and e_i are of different types, we replace B_i with the even length walk guaranteed by the induction hypothesis. Otherwise, we replace it with the odd length walk. Notice that by the alternation requirement, one of P and P' must be odd and the other must be even.

Case *2:* When v is contained in a non-trivial blossom B_k, $0 \leq k < l$. Without loss of generality, $\{e_{k-1}\} = \eta(B_k)$. Consider the contracted walk $\widehat{P} = \langle e_0, e_1, \ldots, e_{k-1}\rangle$. We extend $\widehat{P}$ to an alternating walk P in E_B terminating at e_{k-1} similar to Case 1. Then $P_0(v)$ and $P_1(v)$ are obtained by concatenating P with the alternating walk $P_{B_k,0}(v)$ or $P_{B_k,1}(v)$, whichever has the right parity.

Notice that in both cases, the *base requirement* in Definition 1 guarantees the starting edge of both alternating walks $P_1(v)$ and $P_0(v)$ alternates with the base edge $\eta(B)$. □

The main difference between blossoms in generalized matching problems and blossoms in ordinary matching is that $P_0(v)$ and $P_1(v)$ are *both* meaningful for finding augmenting walks or blossoms. In ordinary matching, since each vertex has at most 1 matched edge incident to it, an alternating walk enters the blossom at the base vertex via a matched edge and must leave with an unmatched edge. As a result the subwalk inside the blossom is always even. In generalized matching problems, this subwalk can be either even or odd, and may contain a cycle. In general, an alternating walk enters the blossom at the base edge and can leave the blossom at *any* nonbase edge.

Similar to ordinary matching algorithms, we *contract* blossoms in order to find augmenting structures to improve our f-matching. Contracting a blossom B means replacing B with a single vertex v with an f-value $f(v) = \sum_{v' \in B} f(v') - 2|M \cap E[B]|$. Here $E[B]$ is the set of edges induced by the vertex set B.

Next we extend to notion of *maturity* from [11, p. 43] to f-matching and f-edge cover. Let us focus on f-matching first. Due to complementary slackness, we can only assign a positive z-value for the pair (B, I) if it satisfies the constraint $|F \cap (\gamma(B) \cup I)| \leq \lfloor (f(B) + |I|)/2 \rfloor$ with equality. For ordinary matching, this requirement is implied by the combinatorial definition of blossoms. However, this is not the case for generalized matching, so we need a blossom to be *mature* to fulfill the complementary slackness property.

Definition 2 (Mature Blossom) A blossom is *mature* w.r.t an f-matching F if it satisfies the following:

1. Every vertex $v \in B \setminus \{\beta(B)\}$ is saturated.
2. $\mathrm{def}(\beta(B)) = 0$ or 1. If $\mathrm{def}(\beta(B)) = 1$, B must be a light blossom and $\eta(B) = \emptyset$; If $\mathrm{def}(\beta(B)) = 0$, $\eta(B) \neq \emptyset$.

The algorithm only contracts and manipulates mature blossoms. The definition for maturity is motivated by the requirement that a blossom processed by the algorithm must satisfy the following two properties:

- Complementary slackness: A dual variable can be positive only if its primal constraint is satisfied with equality. In our algorithm, a blossom can have a positive z-value only if $|F \cap (\gamma(B) \cup I(B))| = \left\lfloor \frac{f(B)+|I(B)|}{2} \right\rfloor$, for a particular $I(B) \subseteq \delta(B)$ that we are going to define momentarily.
- Topology of augmenting walks: An augmenting walk in G can only start with an unmatched edge. As a result, an augmenting walk in the contracted graph must start with a singleton or an unsaturated light blossom. If a blossom is unsaturated, it must be eligible to start an augmenting walk, and thus must be light.

According to Definition 2, a mature blossom cannot be both heavy and unsaturated. Now we show that a mature blossom satisfies its corresponding primal constraint with equality. To show this fact, we first define the I-set of a blossom B [11, p. 44], which is the set $I(B)$ associated with blossom B for which we will assign a positive z-value, given by:

$$I(B) = \delta_F(B) \oplus \eta(B),$$

where $\oplus$ is the symmetric difference operator (XOR). All other subsets I of $\delta(B)$ will have $z(B, I) = 0$. If B is a mature blossom, then we have $|F \cap (\gamma(B) \cup I(B))| = \left\lfloor \frac{f(B)+|I(B)|}{2} \right\rfloor$

Lemma 2 *If an f-matching blossom B is mature, we have* $|F \cap (\gamma(B) \cup I(B))| = \left\lfloor \frac{f(B)+|I(B)|}{2} \right\rfloor$.

Proof We first sketch the idea of the proof. Assume for simplicity that the deficiency is 0 for every vertex $v \in B$, i.e., there are exactly $f(v)$ matched edges incident to v, and every edge in $I(B)$ is matched. Then every matched edge $e \in F \cap \gamma(B)$ contributes 2 to $f(B)$, one for each endpoint, and every edge $e \in I(B) \cap F$ contributes 1 to $f(B)$ and 1 to $|I(B)|$. Thus we have:

$$2|F \cap (\gamma(B) \cup I(B))| = f(B) + |I(B)|.$$

Now we eliminate the assumption by a case analysis for the deficiency of $\beta(B)$. If $\mathrm{def}(\beta(B)) = 0$, the assumption on deficiency holds, while all but possibly 1 edge in $I(B)$, namely the base edge, are matched. This makes $f(B) + |I(B)|$ at least $2|F \cap (\gamma(B) \cup I(B))|$ and at most $2|F \cap (\gamma(B) \cup I(B))| + 1$, and the equality $|F \cap (\gamma(B) \cup I(B))| = \left\lfloor \frac{f(B)+|I(B)|}{2} \right\rfloor$ follows.

When $\mathrm{def}(\beta(B)) = 1$, since $\eta(B) = \emptyset$, the assumption that $I(B)$ only contains matched edges holds. Since exactly 1 of B's vertices has deficiency 1, we have:

$$2|F \cap (\gamma(B) \cup I(B))| + 1 = f(B) + |I(B)|$$

And the equality $|F \cap (\gamma(B) \cup I(B))| = \left\lfloor \frac{f(B)+|I(B)|}{2} \right\rfloor$ follows. □

We complete the discussion by giving the definition for maturity and the corresponding properties for mature blossoms in f-edge cover. The details are similar to f-matching.

Definition 3 (Mature Blossom for f-edge cover) A blossom is *mature* w.r.t an f-edge cover F if it satisfies the following:

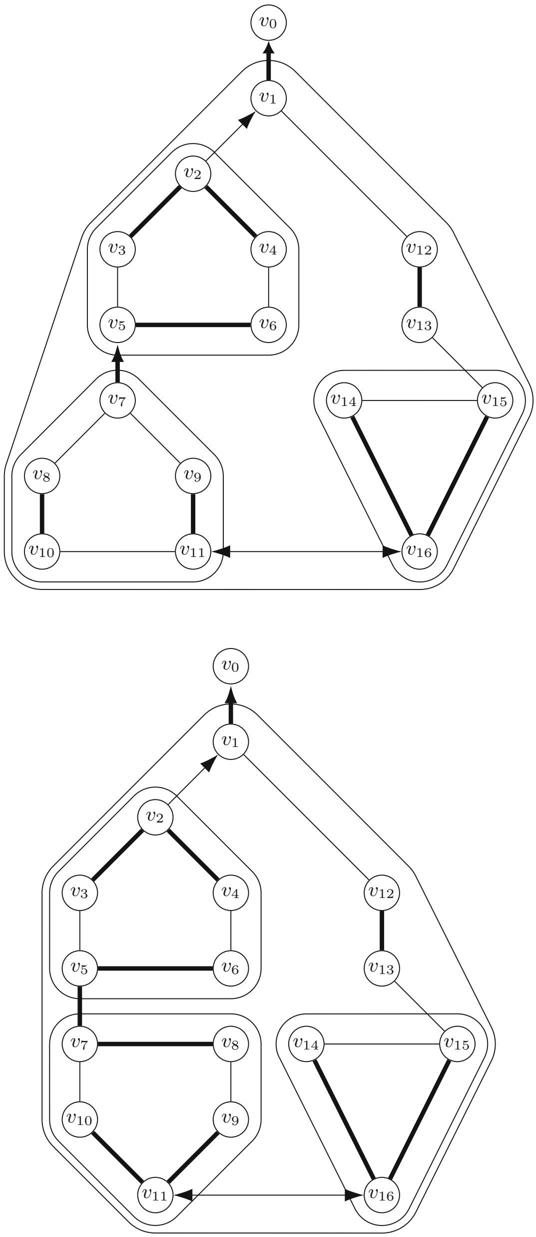
1. Every vertex $v \in B \setminus \{\beta(B)\}$ is saturated: $\deg_F(v) = f(v)$.
2. $\mathrm{surp}(\beta(B)) = 0$ or 1. If $\mathrm{surp}(\beta(B)) = 1$, B must be a heavy blossom and $\eta(B) = \emptyset$; If $\mathrm{surp}(\beta(B)) = 0$, $\eta(B) \neq \emptyset$.

Lemma 3 *If an f-edge-cover blosom B is mature, we have* $|F \cap (\gamma(B) \cup (\delta(B) \setminus I(B)))| = \left\lceil \frac{f(B)-|I(B)|}{2} \right\rceil$.

2.3 Augmenting/reducing walks

Augmenting walks are analagous to augmenting paths from ordinary matching (Fig. 1). Complications arise from the fact that an f-matching blossom cannot be treated identically to a single vertex after it is contracted. For example, in Fig. 2, the two edges (v_0, v_1) and (v_4, v_6) incident to blossom $\{v_1, v_2, v_4, v_5, v_3\}$ are of the same type, both before and after augmenting along the walk $\langle v_0, v_1, v_3, v_5, v_4, v_6 \rangle$. This can never happen in ordinary matching! Moreover, augmenting walks can begin and end at the same vertex and can visit the same vertex multiple times. Hence a naive contraction of a blossom into a single vertex loses key information about the internal structure of blossoms. Definition 4, taken from Gabow [11, p. 28, p. 44] characterizes when a walk in the contracted graph can be extended to an augmenting walk.

Fig. 1 Two examples of contractible blossoms: Bold edges are matched and thin ones are unmatched. Blossoms are circled with a border. Base edges are represented with arrow pointing away from the blossom



Definition 4 Let $\widehat{G}$ be the graph obtained from G by contracting a laminar set Ω of blossoms. Let $\widehat{P} = \langle B_0, e_0, B_1, e_1, \ldots, B_{l-1}, e_{l-1}, B_l \rangle$ be a walk in $\widehat{G}$. Here $\{e_i\}$ are edges and $\{B_i\}$ are nontrivial blossoms or singletons, with $e_i \in B_i \times B_{i+1}$ for all $0 \leq i < l$. We say $\widehat{P}$ is an *augmenting walk* with respect to the f-matching F if the following requirements are satisfied:

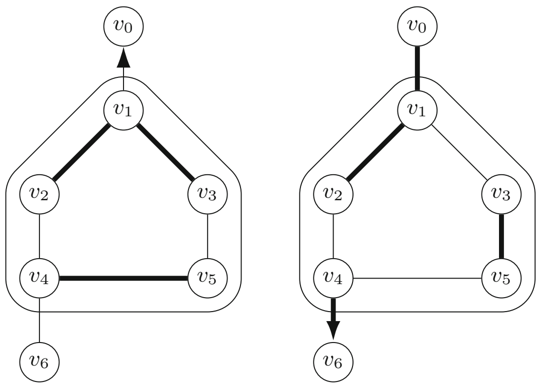


Fig. 2 An example for how a blossom changes with an augmentation: here the augmenting walk is $\langle v_0, v_1, v_3, v_5, v_4, v_6\rangle$. Notice that after rematching, the base edge of the blossom changes from (v_0, v_1) to (v_4, u_6), and the blossom turns from a heavy blossom to a light one

1. *Terminal Vertices Requirement* The terminals B_0 and B_l must be unsaturated singletons or unsaturated light nontrivial blossoms. If P is a closed walk ($B_0 = B_l$), B_0 must be a singleton and $\mathrm{def}(\beta(B_0)) \geq 2$.
2. *Terminal Edges Requirement* If the terminal vertex B_0 (B_l) is a singleton, the incident terminal edge e_0 (e_{l-1}) must be unmatched. Otherwise it can be either matched or unmatched.
3. *Alternation Requirement* Let $B_i, 0 < i < l$, be an internal blossom. If B_i is a singleton, exactly one of e_{i-1} and e_i is matched. If B_i is a nontrivial blossom, $\eta(B_i) \neq \emptyset$ and must be one of $\{e_{i-1}\}$ or $\{e_i\}$.

A natural consequence of the above definition is that an augmenting walk $\widehat{P}$ in $\widehat{G}$ can be extended to an augmenting walk P in G. This is proved exactly as in Lemma 1. We call P the *preimage* of $\widehat{P}$ in G and $\widehat{P}$ the image of P in $\widehat{G}$.

Definition 5 Let $\widehat{P}$ be an augmenting walk in $\widehat{G}$. An *augmentation* along $\widehat{P}$ makes the following changes to F and Ω.

1. Let P be the preimage of $\widehat{P}$ in G. Update F to $F \oplus P$.
2. If $B \in \Omega$ is a blossom intersecting P, we set $\eta(B) \leftarrow (P \cap \delta(B)) \setminus \eta(B)$ and set $\beta(B)$ to the vertex in B that is incident to the edge in $\eta(B)$. Notice that $|P \cap \delta(B)| = 1$ or 2, and in the case when $|P \cap \delta(B)| = 1$, we must have $\eta(B) = \emptyset$.

Some remarks can be made here regarding connection to augmenting walks and mature blossoms.

- A blossom that is not mature may contain an augmenting walk. Specifically, suppose B is light and unsaturated. If any nonbase vertex $v \neq \beta(B)$ in B is also unsaturated, the odd length alternating walk from $\beta(B)$ to v satisfies the definition of an augmenting walk. Alternatively, if $\beta(B)$ has deficiency of 2 or more, the odd length alternating walk from $\beta(B)$ to $\beta(B)$ is also augmenting. For these reasons, the algorithm is designed such that immature blossoms are never contracted.
- Augmentation never destroys maturity. In particular, it never creates an unsaturated heavy blossom. As a result, all blossoms we maintain stay mature throughout the entirety of the algorithm.

In f-edge cover, the corresponding notion is called *reducing walk*. The definition of reducing walk can be naturally obtained from Definition 4 while replacing "unsaturated", "deficiency", and "light" with "oversaturated", "surplus", and "heavy", and exchanging "matched" and "unmatched". It is also worth pointing out that if an f-matching F and an f'-edge cover F' are *complement to* each other, i.e., $F' = E \setminus F$ and $f(v) + f'(v) = \deg(v)$, and they have the same blossom set Ω, then an augmenting walk $\widehat{P}$ for F is also a reducing walk for F'.

2.4 Complementary slackness

To characterize an (approximately) optimal solution, we maintain dual functions: $y : V \mapsto \mathbb{R}_{\geq 0}$ and $z : 2^V \mapsto \mathbb{R}_{\geq 0}$. Here $z(B)$ is short for $z(B, I(B))$. We do not explicitly maintain the edge dual $u : E \mapsto \mathbb{R}_{\geq 0}$ since its minimizing value can be explicitly given by $u(e) = \max\{w(e) - yz(e), 0\}$. For f-matching F, the following property characterizes an approximate maximum weight f-matching:

Property 1 (*Approximate Complementary Slackness for f-matching*) Let $\delta_1, \delta_2 \geq 0$ be nonnegative parameters. We say an f-matching F, duals y, z, and the set of blossoms Ω satisfies (δ_1, δ_2)-approximate complementary slackness if the following hold:

1. *Approximate Domination* For each unmatched edge $e \in E \setminus F$, $yz(e) \geq w(e) - \delta_1$.
2. *Approximate Tightness* For each matched edge $e \in F$, $yz(e) \leq w(e) + \delta_2$.
3. *Blossom Maturity* For each blossom $B \in \Omega$, $|F \cap (\gamma(B) \cup I(B))| = \left\lfloor \frac{f(B)+|I(B)|}{2} \right\rfloor$.
4. *Unsaturated Vertices' Duals* For each unsaturated vertex v, $y(v) = 0$.

Lemma 4 *Let F be an f-matching in G along with duals y, z and let F^* be the maximum weight f-matching. If F, Ω, y, z satisfy Property 1 with parameters δ_1 and δ_2, we have*

$$w(F) \geq w(F^*) - \delta_1|F^*| - \delta_2|F|.$$

Proof We first define $u : E \mapsto \mathbb{R}$ as

$$u(e) = \begin{cases} w(e) - yz(e) + \delta_2, & \text{if } e \in F. \\ 0, & \text{otherwise.} \end{cases}$$

From approximate tightness, we have $u(e) \geq 0$ for all $e \in E$. Therefore, $yz(e) + u(e) \geq w(e) - \delta_1$ for <u>all</u> $e \in E$ and $yz(e) + u(e) = w(e) + \delta_2$ for all $e \in F$. This gives the following:

$$\begin{aligned} w(F) &= \sum_{e \in F} w(e) = \sum_{e \in F} (yz(e) + u(e) - \delta_2) \\ &= \sum_{v \in V} \deg_F(v) y(v) + \sum_{B \in \Omega} |F \cap (\gamma(B) \cup I(B))| z(B) + \sum_{e \in F} u(e) - |F|\delta_2 \end{aligned}$$

By Property 1 (Unsaturated Vertices' Duals, Blossom Maturity, and the definition of u), this is equal to

$$
\begin{aligned}
&= \sum_{v \in V} f(v)y(v) + \sum_{B \in \Omega} \left\lfloor \frac{f(B) + |I(B)|}{2} \right\rfloor z(B) + \sum_{e \in E} u(e) - |F|\delta_2 \\
&\geq \sum_{v \in V} \deg_{F^*}(v)y(v) + \sum_{B \in \Omega} |F^* \cap (\gamma(B) \cup I(B))| z(B) + \sum_{e \in F^*} u(e) - |F|\delta_2 \\
&= \sum_{e \in F^*} (yz(e) + u(e)) - |F|\delta_2 \\
&\geq \sum_{e \in F^*} (w(e) - \delta_1) - |F|\delta_2 = w(F^*) - |F^*|\delta_1 - |F|\delta_2.
\end{aligned}
$$

□

We can easily extend the proof of Lemma 4 to show that if we have multiplicative errors for approximate domination/tightness, F is an approximately optimal solution. Formally, if we have the following multiplicative version of Property 1:

Property 2 (*Approximate Complementary Slackness for f-matching with Multiplicative Error*) Let $0 \leq \epsilon_1, \epsilon_2 < 1$ be nonnegative parameters. We say an f-matching F, duals y, z, and the set of blossoms Ω satisfies (ϵ_1, ϵ_2)-multiplicative approximate complementary slackness if it satisfies Property 1(3,4), with Property 1(1,2) being replaced with:

1. *Approximate Domination* For each unmatched edge $e \in E \setminus F$, $yz(e) \geq (1 - \epsilon_1)w(e)$.
2. *Approximate Tightness* For each matched edge $e \in F$, $yz(e) \leq (1 + \epsilon_2)w(e)$.

We can show the following:

Lemma 5 *Let F be an f-matching in G along with duals y, z and let F^* be the maximum weight f-matching. If F, Ω, y, z satisfy Property 2 with parameters ϵ_1 and ϵ_2, we have*

$$w(F) \geq (1 - \epsilon_1)(1 + \epsilon_2)^{-1} w(F^*)$$

We also give the corresponding theorems for f-edge covers:

Property 3 (*Approximate Complementary Slackness for f-edge cover*) Let $\delta_1, \delta_2 \geq 0$ be positive parameters. We say an f-edge cover C, with duals y, z and blossom family Ω satisfies the (δ_1, δ_2)-approximate complementary slackness if the following requirements holds:

1. *Approximate Domination.* For each unmatched edge $e \in E \setminus C$, $yz_C(e) \leq w(e) + \delta_1$.
2. *Approximate Tightness.* For each matched edge $e \in C$, $yz_C(e) \geq w(e) - \delta_2$.
3. *Blossom Maturity.* For each blossom $B \in \Omega$, $|C \cap (\gamma(B) \cup (\delta(B) \setminus I_C(B)))| = \left\lceil \frac{f(B) - |I_C(B)|}{2} \right\rceil$.

4. *Oversaturated Vertices' Duals.* For each oversaturated vertex v, $y(v) = 0$.

Property 4 (*Approximate Complementary Slackness for f-edge cover with Multiplicative Error*) Let $0 \le \epsilon_1, \epsilon_2 < 1$ be positive parameters. We say an f-edge cover C, with duals y, z and blossom family Ω satisfies the (ϵ_1, ϵ_2)-approximate complementary slackness if it satisfies Property 3(3,4), with Property 3(1,2) being replaced with:

1. *Approximate Domination.* For each unmatched edge $e \in E \setminus C$, $yz_C(e) \le (1 + \epsilon_1)w(e)$.
2. *Approximate Tightness.* For each matched edge $e \in C$, $yz_C(e) \ge (1 - \epsilon_2)w(e)$.

Recall that we are using the aggregated duals yz_C for f-edge cover:

$$yz_C(u, v) = y(u) + y(v) + \sum_{B:(u,v)\in\gamma(B)\cup(\delta(B)\setminus I_C(B))} z(B)$$

Lemma 6 *Let C be an f-edge cover with duals y, z, Ω satisfying Property 3 with parameters δ_1 and δ_2, and let C^* be the minimum weight f-edge cover. We have $w(C) \le w(C^*) + \delta_1|C^*| + \delta_2|C|$.*

Lemma 7 *Let C be an f-edge cover with duals y, z, Ω satisfying Property 4 with parameters ϵ_1 and ϵ_2, and let C^* be the minimum weight f-edge cover. We have $w(C) \le (1 + \epsilon_1)(1 - \epsilon_2)^{-1}w(C^*)$.*

3 Connection Between *f*-Matchings and *f*-Edge Covers

The classical approach to solve the f-edge cover problem is to reduce it to f-matching. Specifically, looking for a minimum weight f_C-edge cover C for some function f_C can be seen as choosing edges that are *not* in C, which is a maximum weight f_F-matching where $f_F(u) = \deg(u) - f_C(u)$.

The main drawback of this reduction is that it yields inefficient algorithms. For example, Gabow's algorithms [11] for solving maximum weight f_F-matching scales linearly with $f_F(V)$, which makes it undesirable when f_C is small. Even when $f_C(V) = O(n)$, Gabow's algorithm runs in $O(m^2 + mn \log n)$ time. Moreover, this reduction is not approximation-preserving. In other words, the complement of an *arbitrary* $(1 - \epsilon)$-approximate maximum weight f_F-matching is not guaranteed to be a $(1 + \epsilon)$-approximate f_C-edge cover.

In this section we establish two results: First we prove that a folklore reduction from 1-edge cover to matching in nonnegative weight graphs is approximation preserving. This allows us to use an efficient approximate matching algorithm for ordinary matching, such as [4], to solve the weighted 1-edge cover problem. Then we establish the connection between approximate f_F-matching and approximate f_C-edge cover using approximate complementary slackness from the previous section. This will give a $(1 + \epsilon)$-approximate minimum weight f-edge cover algorithm from our $(1 - \epsilon)$ approximate maximum weight f-matching algorithm.

3.1 Approximate preserving reduction from 1-edge cover to 1-matchings

The edge cover problem is a special case of f-edge cover where f is 1 everywhere. The minimum weight edge cover problem is reducible to maximum weight matching, simply by reweighting edges [23]. The reduction is as follows: Let $e(v)$ be any edge with minimum weight incident to v and let $\mu(v) = w(e(v))$. Define a new weight function w' as follows

$$w'(u, v) = \mu(u) + \mu(v) - w(u, v).$$

Schrijver [23, Chapter 27] showed the following theorem:

Theorem 1 *Let M^* be a maximum weight matching with respect to a nonnegative weight function w', and $C = M^* \cup \{e(v) : v \in V \setminus V(M)\}$. Then C is a minimum weight edge cover with respect to weight function w.*

We show this reduction is also approximation preserving. Recall that the generally weighted versions of these problems are reducible to the *non-negatively* weighted versions in linear time.

Theorem 2 *Let M' be a $(1-\epsilon)$-maximum weight matching with respect to nonnegative weight function w', and $C' = M' \cup \{e(v) : v \in V \setminus V(M')\}$. Then C' is a $(1+\epsilon)$-minimum weight edge cover with respect to weight function w.*

Proof Let C^* and M^* be the optimal edge cover and matching defined previously. By construction, we have

$$\begin{aligned} w(C') &= w(M') + \mu(V \setminus V(M')) \\ &= \mu(V(M')) - w'(M') + \mu(V \setminus V(M')) \\ &= \mu(V) - w'(M') \end{aligned}$$

Similarly, we have $w(C^*) = \mu(V) - w'(M^*)$. Then

$$\begin{aligned} w(C') &= \mu(V) - w'(M) \\ &\le \mu(V) - (1-\epsilon) w'(M^*) \\ &= w(C^*) + \epsilon w'(M^*) \\ &\le w(C^*) + \epsilon w(C^*) \\ &= (1+\epsilon) w(C^*). \end{aligned}$$

The second to last inequality holds because $M^* \subseteq C^*$ and, by definition, $w'(u, v) = \mu(u) + \mu(v) - w(u, v) \le 2w(u, v) - w(u, v) = w(u, v)$. □

The reduction does not naturally extend to f-edge cover. In the next section we will show how to obtain a $(1+\epsilon)$-approximate f-edge cover algorithm from a $(1-\epsilon)$-approximate f-matching within the primal-dual framework.

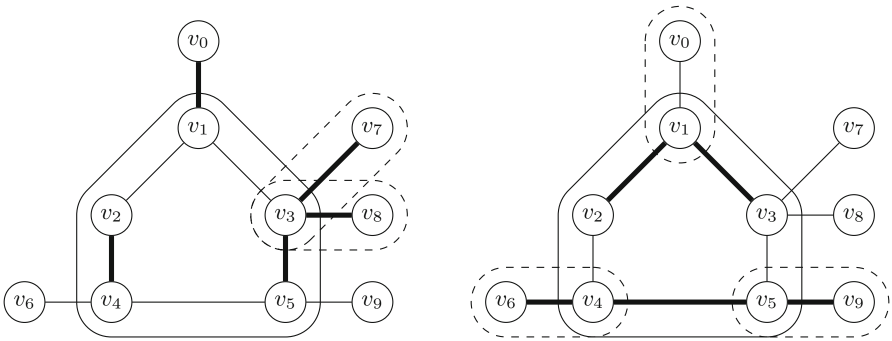


Fig. 3 Illustration on relation between I-set of an f-matching and the I-set of its complementary f'-edge cover. Left: an f-matching and its blossom set. Right: Its complementary f-edge cover. Their I-sets are circled (dashed)

3.2 From *f*-edge cover to *f*-matching

We show that a primal-dual algorithm computing a $(1-\epsilon)$-approximate f-matching can be used to compute a $(1+\epsilon)$-approximate f-edge cover. In particular, we show that if we have an f'-matching F with blossoms Ω and duals y, z satisfying Property 1, and an f-edge cover C that is F's complement, then the same blossom set Ω and duals y, z can be also used to certify Property 3 for C with a same set of parameters. This is formally stated in the following lemma:

Lemma 8 *If the duals y, z, Ω and an f'-matching F satisfy Property 1 with parameters δ_1', δ_2', then the same duals y, z, Ω and the complementary f-edge cover $C = E \setminus F$ satisfies Property 3 with parameters $\delta_1 = \delta_2'$ and $\delta_2 = \delta_1'$.*

Proof It is easy to see a vertex is oversatured in an f-edge cover if and only if it is unsaturated in its complementary f-matchings. Therefore, Property 3(4) (Oversaturated Vertices' Duals) and Property 1(4) (Unsaturated Vertices' Duals) are equivalent to each other.

To show Property 3(1) is equivalent to Property 1(2), and Property 3(2) is equivalent to Property 1(1), it suffices to show that the function yz_F for f'-matching F agrees with the function yz_C for its complementary f-edge cover C (Fig. 3). Recall that

$$yz_C(u,v) = y(u) + y(v) + \sum_{B:(u,v)\in\gamma(B)\cup(\delta(B)\setminus I_C(B))} z(B),$$
$$yz_F(u,v) = y(u) + y(v) + \sum_{B:(u,v)\in\gamma(B)\cup I_F(B)} z(B).$$

Here I_C and I_F refer to the I-sets of a blossom with respect to the f-edge cover C and the f'-matching F. This reduces to showing that $I_F(B) = \delta(B) \setminus I_C(B)$:

$$\begin{aligned} I_F(B) &= \eta(B) \oplus \delta_F(B) = \eta(B) \oplus (\delta(B) \oplus \delta_C(B)) \\ &= \delta(B) \oplus (\eta(B) \oplus \delta_C(B)) = \delta(B) \setminus I_C(B). \end{aligned}$$

Therefore, in $yz_F(e)$ and $yz_C(e)$, z-values are summed up over the same set of blossoms in Ω. In other words, $yz_F(e) = yz_C(e)$ for each $e \in E$ and the claim follows

To prove that Property 1(3) implies Property 3(3), we argue by definition that the maturity of an f'-matching blossom implies the maturity of the corresponding f-edge-cover blossom. Equality is then implied by Lemma 2 and Lemma 3. Indeed, by how we define our f'-matching F and f-edge cover C, a vertex's surplus with respect to C and f is equal to a vertex's deficiency with respect to F and f'. Moreover, the blossom is heavy/light for f'-matching iff it is light/heavy for the corresponding f-edge cover. Since the base edge is defined to be the same, maturity of one blossom implies the other. This completes the proof. □

4 Approximation Algorithms for *f*-Matching and *f*-Edge Cover

In this section, we prove the main result by giving an approximation algorithm for computing $(1-\epsilon)$-approximate maximum weight f-matching. The crux of the result is an implementation of Edmonds' search with relaxed complementary slackness as the eligibility criterion. The notion of approximate complementary slackness was introduced by Gabow and Tarjan for both bipartite matching [14] and general matching [15]. Gabow gave an implementation of Edmonds' search with *exact* complementary slackness for the f-matching problem [11], which finds augmenting walks one at a time. The main contribution of this section is to adapt [11] to approximate complementary slackness to facilitate finding augmenting walks in batches.

To illustrate how this works, we will first give an approximation algorithm for f-matching in graphs with small edge weights. Let $w(\cdot)$ be a positive weight function $w : E \mapsto \{0, \ldots, W\}$. The algorithm computes a $(1-\epsilon)$-approximate maximum weight f-matching in $O(m\alpha(m,n)W\epsilon^{-1})$ time, independent of f. We also show how to use scaling techniques to transform this algorithm to run in $O(m\alpha(m,n)\epsilon^{-1}\log\epsilon^{-1})$ time, independent of W.

4.1 Approximation for small weights

The main procedure in our $O(m\alpha(m,n)W\epsilon^{-1})$ time algorithm is a variation on Edmonds' search. In one iteration, Edmonds' search finds a set of augmenting walks using *eligible edges*, creates and dissolves blossoms, and performs *dual adjustments* on y and z while maintaining the following Invariant:

Invariant 1 (Approximate Complementary Slackness) *Let $\delta > 0$ be some parameter such that $w(e)$ is a multiple of δ, for all $e \in E$:*

1. *Granularity. y-values are multiples of $\delta/2$ and z-values are multiples of δ.*
2. *Approximate Domination. For each unmatched edge and each blossom edge $e \in (E \setminus F) \cup (\bigcup_{B\in\Omega} E_B)$, $yz(e) \geq w(e) - \delta$.*
3. *Approximate Tightness. For each matched and each blossom edge $e \in F \cup (\bigcup_{B\in\Omega} E_B)$, $yz(e) \leq w(e)$.*

4. *Blossom Maturity. For each blossom* $B \in \Omega$, $|F \cap (\gamma(B) \cup I(B))| = \left\lfloor \frac{f(B)+|I(B)|}{2} \right\rfloor$. *Root blossoms in* Ω *have positive z-values.*
5. *Unsaturated Vertices. All unsaturated vertices have the same y-value; their y-values are strictly less than the y-values of other vertices.*

Notice that here we relax Property 1(4) to allow unsaturated vertices to have positive y-values. The purpose of Edmonds' search is to decrease the y-values for all unsaturated vertices while maintaining Invariant 1. Following [4,5,15], we define the following eligibility criterion:

Criterion 1 *An edge* (u, v) *is* eligible *if it satisfies one of the following:*

1. $e \in E_B$ *for some* $B \in \Omega$.
2. $e \notin F$ *and* $yz(e) = w(e) - \delta$.
3. $e \in F$ *and* $yz(e) = w(e)$.

A key property of this definition is that it is asymmetric for matched and unmatched edges that are not in any blossom. As a result, if we augment along an eligible augmenting walk P, all edges in P, except for those in contracted blossoms, will become ineligible; and its image in the contracted graph will become entirely ineligible.

We define G_{elig} to be the graph obtained from G by discarding all ineligible edges, and let $\widehat{G_{elig}} = G_{elig}/\Omega$ be obtained from G_{elig} by contracting all blossoms in Ω. For initialization, we set $F = \emptyset$, $y = W/2$, $z = 0$, $\Omega = \emptyset$. Edmonds' search repeatedly executes the following steps: *Augmentation*, *Blossom Formation*, *Dual Adjustment*, and *Blossom Dissolution* until all unsaturated vertices have 0 y-values. See Fig. 4.[5]

Now we define what we mean by *reachable* vertices in Steps 1–3 of the algorithm, as well as the inner/outer labelling of nontrivial blossoms and singletons. This is analogous to the reachable/inner/outer vertices in Edmonds' Search for ordinary matching [4,5], except that we cannot simply treat a contracted blossom like a single vertex. The corresponding definition for f-matching is given in Gabow [11, p. 46]. For completeness, we restate these definitions and further supplement them with the notion of *alternation*, which provides further insights for reachability.

We start by defining *alternation* which follows from Definition 4 of an augmenting walk. We say two distinct edges e, e' incident to a blossom/singleton B *alternate* if either B is a singleton and e and e' are of different types, or B is a nontrivial blossom and $|\eta(B) \cap \{e, e'\}| = 1$. An *alternating walk/cycle* in the contracted graph is a walk/cycle where every two consecutive edges alternates. An augmenting walk is an alternating walk with its terminal edges and terminal vertices satisfying the requirement specified in Definition 4.

$\widehat{S}$ is the set of blossoms and vertices in $\widehat{G_{elig}}$ that are reachable from an unsaturated singleton or an unsaturated light blossom via an eligible alternating walk. It can be obtained by inductively constructing an alternating search tree rooted at an unsaturated singleton or an unsaturated light blossom. We label the root nodes *outer*. For a nonroot vertices v in $\widehat{S}$, let $\tau(v)$ be the edge in $\widehat{S}$ pointing to the parent of v. The inner/outer status of v is defined as follows:

[5] In an actual implementation, the inner/outer labelling can be computed in the search in Blossom Formation step. The labelling continues to be valid after contracting a maximal set of blossoms.

1. *Augmentation.* Find a set of edge-disjoint augmenting walks $\widehat{\Psi}$ and a set of alternating cycles $\widehat{\mathcal{C}}$ in $\widehat{G_{elig}}$, such that after removing their edges from $\widehat{G_{elig}}$, $\widehat{G_{elig}}$ does not contain any augmenting walk. Let Ψ and $\mathcal{C}$ be the preimages of $\widehat{\Psi}$ and $\widehat{\mathcal{C}}$ in G_{elig}. Update $F \leftarrow F \oplus \left(\left(\bigcup_{P\in\Psi} P\right) \cup \left(\bigcup_{C\in\mathcal{C}} C\right)\right)$. After this step, the new $\widehat{G_{elig}}$ contains no augmenting walk.
2. *Blossom Formation.* Find a maximal set Ω' of nested blossoms reachable from an unsaturated vertex/blossom in $\widehat{G_{elig}}$. Update $\Omega \leftarrow \Omega \cup \Omega'$ and then update $\widehat{G_{elig}}$ to be G/Ω. After this step, $\widehat{G_{elig}}$ contains no blossom reachable from an unsaturated vertex/blossom.
3. *Dual Adjustment.* Let $\widehat{S}$ be the set of vertices from $\widehat{G_{elig}}$ reachable from an unsaturated vertex via an eligible alternating walk. We classify vertices in $\widehat{S}$ into $\widehat{V_{in}}$, the set of inner vertices and $\widehat{V_{out}}$, the set of outer vertices.[5] Let V_{in} and V_{out} be the set of original vertices in V represented by $\widehat{V_{in}}$ and $\widehat{V_{out}}$. Adjust the y and z values as follows:

$$y(v) \leftarrow y(v) - \delta/2, \text{ if } v \in V_{out}$$
$$y(v) \leftarrow y(v) + \delta/2, \text{ if } v \in V_{in}$$
$$z(B) \leftarrow z(B) + \delta, \text{ if } B \text{ is a root blossom in } \widehat{V_{out}}$$
$$z(B) \leftarrow z(B) - \delta, \text{ if } B \text{ is a root blossom in } \widehat{V_{in}}$$

   Here a root blossom is an inclusionwise maximal blossom in Ω.
4. *Blossom Dissolution.* After Dual Adjustment some root blossoms in Ω might have 0 z-values. Remove them from Ω until none exists. Update Ω and $\widehat{G_{elig}}$.

Fig. 4 A $(1-\epsilon)$-approximate f-matching algorithm for small integer weights

Definition 6 [11, p. 46] A vertex v is *outer* if one of the following is satisfied:

1. v is the root of a search tree.
2. v is a singleton and $\tau(v) \in F$.
3. v is a nontrivial blossom and $\{\tau(v)\} = \eta(v)$.

Otherwise, one the the following holds and v is classified as *inner*:

1. v is a singleton and $\tau(v) \in E \backslash F$.
2. v is a nontrivial blossom and $\{\tau(v)\} \neq \eta(v)$.

An individual search tree in $\widehat{S}$, call it $\widehat{T}$, can be grown by repeatedly attaching a child v to its parent u using an edge (u, v) that is *eligible for u* in $\widehat{S}$; See Gabow [11, p. 46]. Let B_u denote the root blossom in Ω containing u. We say an edge $(u, v) \in E$ is *eligible for u* if it is eligible according to Criterion 1 and one of the following is satisfied:

1. u is an outer singleton and $e \notin F$.
2. B_u is an outer blossom and $\{e\} \neq \eta(B_u)$.
3. u is an inner singleton and $e \in F$.

4. B_u is an inner blossom and $\{e\} = \eta(B_u)$.

Hence, $\widehat{S}$ consists of singletons and blossoms that are reachable from an unsaturated singleton or light blossom, via an eligible alternating path. We call such blossoms and singletons *reachable*, and all other singletons and blossoms *unreachable*. A vertex v from the original graph G_{elig} is reachable (unreachable) if B_v is reachable (unreachable) in $\widehat{G_{elig}}$.[6]

In Edmonds' Search, primal and dual variables are initialized in a way that Property 1(1) (Approximate Domination) is always satisfied, and Property 1 (Approximate Tightness) is vacuous (as the f-matching is initially empty) but Property 1(4) (Unsaturated Vertices) is not. For this reason, there is a large gap between primal and dual objective, besides the error introduced by *approximate* tightness and domination, at the beginning of the algorithm. This gap is given by the following, assuming exact tightness and domination is satisfied (i.e. $\delta_1 = \delta_2 = 0$ in Property 1):

$$\begin{aligned} & yz(V) - w(F) \\ &\quad = \sum_{v \in V} f(v)y(v) + \sum_{B \in \Omega} \left\lfloor \frac{f(B) + |I(B)|}{2} \right\rfloor z(B) + \sum_{e \in E} u(e) - \sum_{e \in F} w(e) \\ &\quad = \sum_{v \in V} \operatorname{def}(v)y(v). \end{aligned}$$

The goal of the algorithm can be seen as bridging the gap between the primal objective and dual objective while preserving all other complementary slackness properties. It can be achieve in two ways. Augmentations enlarge the f-matching by augmenting F along some augmenting walk P. This will reduce the total deficiency on the vertex set V. Dual Adjustments change the dual variables in a way that decreases the y-value on unsaturated vertices while maintaining other approximate complementary slackness conditions. In this algorithm, the progress of Edmonds' Search is measured by the latter, i.e., the overall reduction in y-values of unsaturated vertices.

The correctness of our algorithm reduces to showing that *Augmentation*, *Blossom Formation*, *Blossom Dissolution*, and *Dual Adjustment* all preserve Invariant 1.

Lemma 9 *The Augmentation step, Blossom Formation step and Blossom Dissolution step preserve Invariant 1.*

Proof We first show that the identity of $I(B)$ is invariant under an augmentation; in particular, augmenting along an augmenting walk that intersects B does not change $I(B)$. As a result, the function $yz(\cdot)$ is invariant under augmentation. This is a restatement of Lemma 5.3 in [11], for completeness, we restate the proof.

We use $I(B)$, $\eta(B)$ and $I'(B)$, $\eta'(B)$ to denote the I-set and base edge of B before and after the augmentation. By Definition 4 (augmenting walks), if P intersects B, then

$$\delta_P(B) = \eta(B) \cup \eta'(B) = \eta(B) \oplus \eta'(B).$$

[6] Of course, if B_v is inner and reachable in $\widehat{G}$, this only implies that $\beta(B_v)$ is reachable from an unsaturated vertex in G; other vertices in B_v may not be reachable in G.

Let F and F' be the f-matching before and after augmentation. We have

$$\delta_{F'}(B) = \delta_F(B) \oplus \delta_P(B)$$

Combining both equations, we have

$$\delta_{F'}(B) = \delta_F(B) \oplus (\eta(B) \oplus \eta'(B))$$

Hence

$$I'(B) = \delta_{F'}(B) \oplus \eta'(B) = \delta_F(B) \oplus \eta(B) = I(B).$$

By Invariant 1, any blossom edge $e \in \bigcup_{B \in \Omega} E_B$ satisfies both approximate domination as well as approximate tightness, so it continues to satisfy these Invariants after augmentation. For any eligible edge not in E_B for any $B \in \Omega$, by Criterion 1, if e is matched, $yz(e) = w(e) - \delta$, thus after the Augmentation step its duals satisfy approximate domination. If e is unmatched, $yz(e) = w(e)$, so its duals satisfy approximate tightness after the Augmentation step.

Augmentation also preserves the maturity of blossoms. For any vertex v in a nonterminal blossom B, $\deg_F(v) = \deg_{F'}(v) = f(v)$, so maturity is naturally preserved. If B is a terminal blossom, we have $\deg_F(v) = f(v) - 1$ for $v = \beta(B)$ and $\deg_F(v) = f(v)$ for all $v \neq \beta(B)$. Moreover, after augmentation B always has a base edge $\eta(B) = \delta_P(B)$. Therefore, B is also mature after augmentation.

All the newly formed blossoms in this step must be mature and have 0 z-values, so the value of the yz function is unchanged and all the invariants are preserved.

For blossom dissolution step, discarding blossoms with zero z-values preserves the value of the yz function and hence preserves the invariants. □

The crux of the proof is to show that Dual Adjustment also preserves Invariant 1, in particular approximate domination and approximate tightness. Before proving the correctness of Dual Adjustment, we first prove the following parity lemma, which was first used in [15]; we generalize it to f-matching:

Lemma 10 (Parity) *Let $\widehat{S}$ be the search forest defined as above. Let S be the preimage of $\widehat{S}$ in G. The y-value of every vertex in S has the same parity, as a multiple of $\delta/2$.*

Proof The claim clearly holds after initialization as all vertices have the same y-values. Now notice that every *eligible edges* $e = (u, v)$ that straddles two distinct singletons or nontrivial blossom must have its yz-value being $w(e)$ or $w(e) - \delta$. Since $w(e)$ is by assumption an integral multiple of δ, $yz(e)$ is also a multiple of δ. Because z-values are always multiples of δ, $y(u)$ and $y(v)$ must both be odd or even as a multiple of $\delta/2$.

Therefore it suffices to show that every vertex in a blossom $B \in \Omega$ has the same parity.

To prove this, we only need to show that the Blossom Formation step only groups vertices with the same parity together. This is because new blossoms B are formed when we encounter a cycle of nontrivial blossoms and singletons $C_B = \langle B_0, e_0, B_1, e_1, \ldots, B_{l-1}, e_{l-1} \rangle$ whose edges are eligible. Therefore the endpoints of those edges share the same parity. Hence by induction, all vertices in B also share the same parity. The Dual Adjustment step also preserves this property as vertices in a blossom will have the same inner/outer classification and thus have their y-values all incremented or decremented by $\delta/2$. □

The following theorem is a generalization of Lemma 5.8 in [11] to approximate complementary slackness. The proof follows from the same framework but has a slightly more complicated case analysis.

Lemma 11 *Dual Adjustment and Blossom Dissolution preserves Invariant 1.*

Proof We focus on part 2 (Approximate Domination) and part 3 (Tightness) of Property 1. Part 1 (Granularity) is naturally preserved since we are adjusting y-values by $\delta/2$ and z-values by δ. Part 5 (Unsaturated vertices duals) is also preserved because unsaturated vertices are labelled as outer and their dual is adjusted by the same amount. Maturity of blossoms is not affected by Dual Adjustment. Although after dual adjustment, some (inner) root blossoms might have 0 z-values, such blossoms are removed in Blossom Dissolution step so part 4 for Invariant 1 is restored at the end of the iteration.

Similar to ordinary matching, preservation of approximate domination and tightness can be argued using a case analysis on vertices and blossoms dual. Notice that there are more cases to consider in f-matching compared to ordinary matching. Different cases can be generated for an edge (u, v) by considering the inner/outer classification of both endpoints, whether (u, v) is matched, whether (u, v) is the base edge for its respective endpoints, if they are in blossoms, and whether (u, v) is eligible. In the following analysis, we follow the framework in Lemma 5.8 [11] to narrow down the number of meaningful cases to just 8. Notice that Lemma 5.8 [11] can be seen as a version of this lemma for exact complementary slackness. Although one can expect the same conclusion to hold, the proof still differs in details.

We consider an edge $e = (u, v)$. If u and v are both unreachable, or both in the same root blossom, $yz(u, v)$ clearly remains unchanged after Dual Adjustment.

Therefore we can assume $B_u \neq B_v$ and at least one of them, say B_u, is reachable. Every reachable endpoint will contribute a change of $\pm\delta/2$ to $yz(u, v)$. This is the adjustment of $y(u)$, plus the adjustment of $z(B_u)$ if $e \in I(B_u)$. Define $\Delta(u)$ to be the net change of the quantity $y(u) + \sum_{e \in I(B_u)} z(B_u)$. By definition of Dual Adjustment, we have the following scenarios:

- $\Delta(u) = +\delta/2$: This occurs if u is an inner singleton, or B_u is an outer blossom with $e \in I(B_u)$, or an inner blossom with $e \notin I(B_u)$.
- $\Delta(u) = -\delta/2$: This occurs if u is an outer singleton, or B_u is an inner blossom with $e \in I(B_u)$, or an outer blossom with $e \notin I(B_u)$.

Then we consider the effect of a Dual Adjustment on the edge $e = (u, v)$. First we consider the case when exactly one of B_u and B_v, say B_u, is in $\widehat{S}$. In this case only u will introduce a change on $yz(u, v)$:

Case *1: u* is an inner singleton: Here $\Delta(u) = +\delta/2$. In this case approximate domination is preserved, so we only need to worry about approximate tightness and hence assume $e \in F$. Since B_v is not in $\widehat{S}$, e cannot be eligible for B_u or B_v would have been included in $\widehat{S}$ as a child of B_u. Because $e \in F$, e cannot be eligible. Hence $yz(e) < w(e)$. By Granularity, $yz(e) \leq w(e) - \delta/2$. Therefore we have $yz(e) \leq w(e)$ after the Dual Adjustment.

Case *2: u is an outer singleton*: Here $\Delta(u) = -\delta/2$. In this case tightness is preserved and we only need to worry about approximate domination when $e \notin F$. Similar to Case 1, e must be ineligible and $yz(e) \geq w(e) - \delta/2$. After Dual Adjustment we have $yz(e) \geq w(e) - \delta$.

Case *3: B_u is an inner blossom*: We divide the cases according to whether e is matched or not.

Subcase 3.1: $e \in F$. If $e \notin \eta(B_u)$, then $e \in I(B_u)$ and $\Delta(u) = -\delta/2$. In this case tightness is preserved. If $e \in \eta(B_u)$, then $e \notin I(B_u)$ and $\Delta(u) = +\delta/2$. But e cannot be eligible since otherwise B_v would be in the search tree, so we have $yz(e) \leq w(e) - \delta/2$ and $yz(e) \leq w(e)$ after Dual Adjustment.

Subcase 3.2: $e \notin F$. This is basically symmetric to Subcase 3.1. If $e \in \eta(B_u)$, then $e \in I(B_u)$ and $\Delta(u) = -\delta/2$. But e cannot be eligible therefore $yz(e) \geq w(e) - \delta/2$, and $yz(e) \geq w(e) - \delta$ after Dual Adjustment. If $e \notin \eta(B_u)$, then $e \notin I(B_u)$ and $\Delta(u) = +\delta/2$, so approximate Domination is preserved.

Case *4: B_u is an outer blossom*:

Subcase 4.1: $e \in F$. If $e \in \eta(B_u)$, then B_v must be the parent of B_u in the search tree, contradicting the fact that $B_v \notin \widehat{S}$. Thus $e \notin \eta(B_u)$, so $e \in I(B_u)$ and $\Delta(u) = +\delta/2$. Since B_v is not reachable, e cannot be eligible, so $yz(u, v) \leq w(e) - \delta/2$ before Dual Adjustment and $yz(u, v) \leq w(e)$ afterward.

Subcase 4.2: $e \notin F$. Similarly, $e \notin \eta(B_u)$, so $e \notin I(B_u)$ and $\Delta(u) = -\delta/2$. Similarly B_v is not reachable so e cannot be eligible. Therefore we have $yz(u, v) \geq w(e) - \delta/2$ and $yz(u, v) \geq w(e) - \delta$ after Dual Adjustment.

This completes the case when exactly one of e's endpoints is reachable. The following part will complete the argument for when both endpoints are reachable. We argue that three scenarios can happen: either $\Delta(u)$ and $\Delta(v)$ are of opposite signs and cancel each other out, or $\Delta(u)$ and $\Delta(v)$ are of the same sign and the sign aligns with the property we wish to keep, or if neither case holds, we use Lemma 10 (Parity) to argue that there is enough room for dual adjustment not to violate approximate domination or tightness.

We first examine tree edges in $\widehat{S}$. In this case we assume B_u is the parent of B_v and e is the parent edge of B_v. Hence e must be eligible for B_u. We argue by the sign of $\Delta(u)$.

Case *5: If e is a tree edge and $\Delta(u) = +\delta/2$*:

There are three cases here: u is an inner singleton, B_u is an outer blossom with $e \in I(B_u)$, or B_u is an inner blossom with $e \notin I(B_u)$. We first observe that in all three cases, $e \in F$. This is straightforward when u is an inner singleton. If B_u is an outer blossom with $e \in I(B_u)$, we know that since B_u is outer, $e \notin \eta(B_u)$, so therefore $e \in F$. If B_u is an inner singleton with $e \notin I(B_u)$, since B_u is inner, $e \in \eta(B_u)$, so combined with the fact that $e \notin I(B_u)$ we have $e \in F$.

Notice that since B_u is the parent of B_v, and $e \in F$, v can be an outer singleton, or B_v is an outer blossom with $e \in \eta(B_v)$, or B_v is an inner blossom with $e \notin \eta(B_v)$. In the second case $e \notin I(B_v)$ and in the third case $e \in I(B_v)$. In all three cases we have $\Delta(v) = -\delta/2$, and $yz(e)$ remains unchanged.

Case *6:* If e is a tree edge and $\Delta(u) = -\delta/2$: Case 6 is symmetric to Case 5. B_u can either be an outer singleton, an inner blossom with $e \in I(B_u)$, or an outer blossom with $e \notin I(B_u)$. In all cases, the fact that e must be eligible for B_u implies $e \notin F$, and B_v can only be an inner singleton, an outer blossom with $e \in I(B_u)$, or an inner blossom with $e \notin I(B_v)$. Hence we have $\Delta(v) = +\delta/2$ so $yz(e)$ remains unchanged.

Now suppose B_u and B_v are both in $\widehat{S}$ but (u, v) is not a tree edge. We still break the cases according to the sign of $\Delta(u)$ and $\Delta(v)$. Here we only need to consider when $\Delta(u) = \Delta(v)$, since otherwise they cancel each other and $yz(e)$ remains constant.

Case *7:* If e is a tree edge and $\Delta(u) = \Delta(v) = \delta/2$. In this case $yz(e)$ is incremented by δ. Therefore we only need to worry about tightness when $e \in F$. Notice that B_u can only be an inner singleton, an outer blossom with $e \in I(B_u)$ or an inner blossom with $e \notin I(B_u)$. When B_u is an outer blossom, $e \notin \eta(B_u)$. When B_u is an inner blossom, since $e \in F$ and $e \notin I(B_u)$, $e \in \eta(B_u)$. The same holds for the other endpoint B_v.

It is easy to verify that in all cases, e is eligible for B_u (or B_v) if and only if e is eligible. But notice that after Augmentation and Blossom Formation steps, there is no augmenting walk or reachable blossom in $\widehat{G_{elig}}$, i.e., there cannot be an edge (u, v) that is eligible for both endpoints B_u and B_v since otherwise one can find an augmenting walk or a new reachable blossom. Thus e is ineligible and $yz(e) < w(e)$. But by Invariant 1(1) (Granularity) and Lemma 10 (Parity), both $w(e)$ and $yz(e)$ must be multiples of δ. Therefore we have $yz(e) \le w(e) - \delta$. This implies $yz(e) \le w(e)$ after Dual Adjustment.

Case *8:* If e is a tree edge and $\Delta(u) = \Delta(v) = -\delta/2$. Here $yz(e)$ is decremented by δ. Similar to the case above, we can assume $e \notin F$ and only focus on approximate domination. B_u can be an outer singleton, inner blossom with $e \in I(B_u)$, or outer blossom with $e \notin I(B_u)$. Since $e \notin F$, $e \in I(B_u)$ if and only if $e \in \eta(B_u)$. Therefore if e is eligible, e must be eligible for both B_u and B_v. But similar to Case 7, e being eligible for both endpoints will lead to the discovery of an additional blossom or augmenting walk in G_{elig}, which is impossible after Augmentation and Blossom Formation. Therefore we conclude in this case e is ineligible and $yz(e) > w(e) - \delta$. By Lemma 10 (Parity), we have $yz(e) \ge w(e)$ before Dual Adjustment, so approximate domination still holds after Dual Adjustment. □

Theorem 3 *A* $(1-\epsilon)$*-approximate* f*-matching can be computed in* $O(Wm\alpha(m,n)\epsilon^{-1})$ *time.*

Proof We initialize the f-matching to be $\emptyset$ and $y(v) = W/2$ for all v. Set $\delta = 1/\lceil \epsilon^{-1} \rceil \le \epsilon$. Since each iteration decreases y-values by $\delta/2$, y-values of unsaturated vertices takes $(W/2)/(\delta/2) = O(W\epsilon^{-1})$ iterations to reach 0, thereby satisfying Property 1 with $\delta_1 = \delta$, $\delta_2 = 0$. By invoking Lemma 4, with F^* being the optimum f-matching, we have

$$w(F) \ge w(F^*) - |F^*|\delta \ge w(F^*) - w(F^*)\delta \ge (1-\epsilon)w(F^*).$$

For the running time, each iteration of Augmentation, Blossom Formation, Dual Adjustment, and Blossom Dissolution can be implemented in $O(m\alpha(m,n))$. We defer the detailed implementation to Sect. 5. There are a total of $W/\delta = O(W\epsilon^{-1})$ iterations, so the running time is $O(Wm\alpha(m,n)\epsilon^{-1})$. □

As a result of Lemma 8 and Lemma 6, we also obtain the following result:

Corollary 1 *A* $(1+\epsilon)$*-approximate* f*-edge cover can be computed in* $O(Wm\alpha(m,n)\epsilon^{-1})$ *time.*

Proof Given a weighted graph G and degree constraint function f, let $f' = \deg - f$ be the complement of f. With some paramter δ we run the algorithm from Theorem 3 to find an f'-matching F' that satisfies Property 1 with parameters $(\delta, 0)$. By Lemma 8, its complement $F = E \setminus F'$ satisfies Property 3 with parameter $(0, \delta)$. By Lemma 6, we have

$$w(F) - \delta|F| \leq w(F^*)$$
$$(1-\delta)w(F) \leq w(F^*)$$

Then we can choose a $\delta = \Theta(\epsilon)$ to guarantee that we get an $(1+\epsilon)$-approximate minimum weight f-edge cover. □

Also notice that when $W = O(1)$ is constant, Theorem 3 and Corollary 1 are the fastest known approximation algorithms for these problems.

4.2 A scaling algorithm for general weights

In this section, we can modify the $O(Wm\alpha(m,n)\epsilon^{-1})$ weighted f-matching algorithm to work on graphs with general real weights. The modification is based on the scaling framework in [4]. If the weights are arbitrary reals, we can round them to integers in $[W]$, $W = \text{poly}(n)$, with negligible loss in accuracy. Thus we can assume without loss of generality that all weights are $O(\log n)$-bit integers. The idea is to divide the algorithm into into $L = \log W + 1$ scales that execute Edmonds' search with exponentially diminishing δ. The goal of each scale is to use $O(\epsilon^{-1})$ Edmonds' searches to halve the y-values of all unsaturated vertices while maintaining a more relaxed version of approximate complementary slackness. By manipulating the weight function, approximate domination, which is weak at the beginning, is strengthened over scales, while approximate tightness is weakened in exchange. Assume without loss of generality that $W > 1$ and $\epsilon < 1$ are powers of two. We define δ_i, $0 \leq i \leq L$ be the error parameter for each scale, where $\delta_0 = \epsilon W$ and $\delta_i = \delta_{i-1}/2$ for $0 < i \leq L$. Each scale works with a new weight function w_i which is the old weight function rounded down to the nearest multiple of δ_i, i.e, $w_i(e) = \delta_i \lfloor w(e)/\delta_i \rfloor$. In the last scale $W_L = w$. We maintain a scaled version of Invariant 1 at each scale:

Invariant 2 (Scaled approximate complementary slackness with positive unsaturated vertices) *At scale* $i = 0, 1, \ldots, L = \log W$*, we maintain the* f*-matching* F*, blossoms* Ω*, and duals* y*,* z *to satisfy the following invariant:*

1. *Granularity. All* y*-values are multiples of* $\delta_i/2$*, and* z*-values are multiples of* δ_i*.*
2. *Approximate Domination. For each* $e \notin F$ *or* $e \in E_B$ *for some* $B \in \Omega$*,* $yz(e) \geq w_i(e) - \delta_i$*.*
3. *Approximate Tightness. For each* $e \in F \cup (\bigcup_{B\in\Omega} E_B)$*, let* $j_e \leq i$ *be the index of last scale that* e *joined the set* $F \cup \bigcup_{B\in\Omega} E_B$*. We have* $yz(e) \leq w_i(e) + 2\delta_{j_e} - 2\delta_i$*.*

4. *Mature Blossoms. For each blossom* $B \in \Omega$, $|F \cap (\gamma(B) \cup I(B))| = \left\lfloor \frac{f(B)+|I(B)|}{2} \right\rfloor$.
5. *Unsaturated Vertices' Duals. The* y*-values of all unsaturated vertices are the same and less than the* y*-values of other vertices.*

Based on Invariant 2, Edmonds' search will use the following Eligibility criterion:

Criterion 2 *At scale* i*, an edge* $e \in E$ *is eligible if one of the following holds*

1. $e \in E_B$ *for some* $B \in \Omega$.
2. $e \notin F$ *and* $yz(e) = w_i(e) - \delta_i$.
3. $e \in F$ *and* $yz(e) - w_i(e)$ *is a nonnegative integer multiple of* δ_i.

This is similar to Criterion 1 except for we have a relaxed criterion for when $e \in F$. This relaxation is due to the fact that tightness is weakened at termination of each scale, and the eligibility criterion is then relaxed to accommodate it. We argue below that this relaxation does not affect the correctness of Edmonds' Search.

Before the start of scale 0, the algorithm initializes F, Ω, y, z similar to the algorithm for small edge weights: $y(u) \leftarrow W/2, \Omega \leftarrow \emptyset, F \leftarrow \emptyset$. At scale i, the duals of unsaturated vertices start at $W/2^{i+1}$. We execute $(W/2^{i+2})/(\delta_i/2) = O(\epsilon^{-1})$ iterations of Edmonds' search with parameter δ_i, using Criterion 2 of eligibility. The scale terminates when the y-values of unsaturated vertices are reduced to $W/2^{i+2}$, or in the last iteration, as they reach 0.

Notice that although the invariant and the eligibility criterion are changed, the fact that Edmonds' search preserves the complementary slackness invariant still holds. The proof of Lemma 11 goes through, as long as the definition of eligibility guarantees the following parity property:

Lemma 12 *At any point of scale* i*, let* $\overline{S}$ *be the set of vertices in* G_{elig} *reachable from an unsaturated vertex using eligible edges. The* y*-value of any vertex* $v \in V$ *with* $B_v \in \overline{S}$ *has the same parity as a multiple of* $\delta_i/2$.

We omit the proof of Lemma 12. The details are similar to Lemma 11, using Criterion 2 in lieu of Criterion 1.

Now we sketch why Criterion 2 ensures Invariant 2, in particular, how it ensures approximate domination and approximate tightness. We will not prove it formally as the details are very similar to Lemma 11 and Lemma 9.

Observe that primal and dual variables initially satisfy Invariant 2, in particular parts 2 and 3. This is because all edges have yz-values equal to W, and no edge is in $M \bigcup_{B \in \Omega} E_B$.

Notice that dual adjustment never changes the yz-values of edges inside any blossom $B \in \Omega$, while it will have the following effect on edge e if its endpoints lie in different blossoms.

1. If $e \notin F$ and is ineligible, $yz(e)$ might decrease but will never drop below the threshold for eligibility, i.e., it will not drop below $w_i(e) - \delta_i$.
2. If $e \notin F$ and is eligible, $yz(e)$ will never decrease.
3. If $e \in F$ and is ineligible, $yz(e)$ might increase but will never exceed the threshold for eligibility, i.e., it will not raise above $w_i(e) + 2\delta_{j_e} - 2\delta_i$.
4. If $e \in F$ and is eligible, $yz(e)$ will never increase.

In other words, with the proper definition of Eligibility, Dual Adjustment will not destroy approximate domination and approximate tightness. Therefore Edmonds' search within scale i will preserve Invariant 2.

We also need to manipulate the duals between different scales to ensure Invariant 2. Formally, after completion of scale i, we increment all the y-values by δ_{i+1}, i.e., if yz' and yz are the function before and after dual adjustment, $yz(e) = yz'(e) + 2\delta_{i+1}$. No change is made to F, Ω and z. This will ensure both approximate domination and approximate tightness hold at scale $i+1$. At the previous scale we have approximate domination $yz(e) \geq w_i(e) - \delta_i$. The weights at scale i and $i+1$ satisfy $w_{i+1}(e) \leq w_i(e) + \delta_{i+1}$. Thus, after dual adjustment,

$$\begin{aligned} yz(e) &= yz'(e) + 2\delta_{i+1} \\ &\geq w_i(e) - \delta_i + 2\delta_{i+1} \\ &\geq w_{i+1}(e) - \delta_{i+1} - \delta_i + 2\delta_{i+1} \\ &= w_{i+1}(e) - \delta_{i+1} \end{aligned}$$

For approximate tightness, we have

$$yz(e) - w_{i+1}(e) \leq yz(e) - w_i(e) \leq 2\delta_{j_e} - 2\delta_i + 2\delta_{i+1} = 2\delta_{j_e} - 2\delta_{i+1},$$

since $\delta_{i+1} = \delta_i/2$.

This step is the main motivation for the definition of Invariant 2 (3), as approximate tightness is gradually relaxed in this step. The algorithm terminates when the y-values of all unsaturated vertices reach 0. It terminates with an f-matching F and its corresponding duals y, z and Ω satisfying the following property:

Property 5 (Final Complementary Slackness)

1. *Approximate Domination.* For all $e \notin F$ or $e \in E_B$ for any $B \in \Omega$, $yz(e) \geq w(e) - \delta_L$.
2. *Approximate Tightness.* For all $e \in F \cup (\bigcup_{B\in\Omega} E_B)$, let j_e be the index of the last scale that e joined $F \cup (\bigcup_{B\in\Omega} E_B)$. We have $yz(e) \leq w(e) + 2\delta_{j_e}$.
3. *Blossom Maturity.* For all blossoms $B \in \Omega$, $|F \cap (\gamma(B) \cup I(B))| = \left\lfloor \frac{f(V)+|I(B)|}{2} \right\rfloor$).
4. *Unsaturated Vertices' Duals.* The y-values of all unsaturated vertices are 0.

This implies approximate domination and approximate tightness are satisfied within some factor $1 \pm O(\epsilon)$. For approximate domination this is easy to see since $w(e) \geq 1$ and $\delta_L = \epsilon/2$, thus $yz(e) \geq (1-\epsilon/2)w(e)$ if $e \notin F$. For approximate tightness, we can lower bound the weight of e if e last entered F or a blossom at scale $j = j_e$. Throughout scale j, the y-values are at least $W/2^{j+2}$, so $w(e) \geq w_j(e) \geq 2(W/2^{j+2}) - \delta_j$. Since $\delta_j = \epsilon W/2^j$, $yz(e) \leq w(e) + 2\delta_j \leq (1+4\epsilon)w(e)$ when $e \in F$.

This implies we have $O(\epsilon)$ multiplicative error for both approximate domination and approximate tightness. Together with the Lemma 5, we can show F is a $(1-\epsilon)$-approximate maximum weight f-matching.

The running time of the algorithm is $O(m\epsilon^{-1} \log W)$ because there are $\log W + 1$ scales, and each scale consists of $O(\epsilon^{-1})$ iterations of Edmonds' search, which can be implemented in $O(m\alpha(m,n))$.

Theorem 4 *A $(1-\epsilon)$-approximate maximum weight f-matching can be computed in $O(m\alpha(m,n)\epsilon^{-1}\log W)$ time.*

Corollary 2 *A $(1+\epsilon)$-approximate minimum weight f-edge cover can be computed in $O(m\alpha(m,n)\epsilon^{-1}\log W)$ time.*

4.3 A $O_\epsilon(m\alpha(m,n))$ Algorithm

We also point out that by applying techniques in [4, Sect. 3.2], the algorithm can be modified to run in time independent of W. The main idea is to force the algorithm to ignore an edge e for all but $O(\log\epsilon^{-1})$ scales. First, we index edges by the first scale that it can ever become eligible. Since at scale i, y-values can drop at most to $W/2^{i+1}$, any edge with weight below $W/2^i$ cannot be eligible at scale i. Let $\mu_i = W/2^i$ and $\text{scale}(e)$ be the unique i such that $w(e) \in [\mu_i, \mu_{i-1})$. Notice that we can ignore e in any scale $j < \text{scale}(e)$. Moreover, we will also forcibly ignore e at scale $j > \text{scale}(e)+\lambda$ where $\lambda = \log\epsilon^{-1} + O(1)$. Ignoring an otherwise eligible edge might cause violations of approximate tightness and approximate domination. However, since the y-values of free vertices are $O(\epsilon w(e))$ at this point, this violation will only amount to $O(\epsilon w(e))$.

To see this, notice that μ_i is also an upper bound to the amount of change to $yz(e)$ caused by all Dual Adjustment *after* scale i. Hence, after scale $\text{scale}(e)+\lambda$, the total amount of violation to approximate tightness and approximate domination on e can be bounded by $\mu_{\text{scale}(e)+\lambda} = O(\epsilon)\mu_{\text{scale}(e)} = O(\epsilon)w(e)$, which guarantees we still get a $(1-O(\epsilon))$-approximate solution.

Therefore, every edge takes part in at most $\log\epsilon^{-1} + O(1)$ scales, with $O(\epsilon^{-1})$ cost per scale. The total running time is $O(m\alpha(m,n)\epsilon^{-1}\log\epsilon^{-1})$. We are omitting the full proof of Theorem 5.

Theorem 5 *A $(1-\epsilon)$-approximate maximum weight f-matching and a $(1+\epsilon)$-approximate minimum weight f-edge cover can be computed in $O(m\alpha(m,n)\epsilon^{-1}\log\epsilon^{-1})$ time, independent of the weight function.*

5 An $O(m\alpha(m,n))$ Augmenting Walk Algorithm

In this section, we show how to implement the augmentation and blossom formation steps in $O(m\alpha(m,n))$ time. The goal of the augmentation step is to find a set of augmenting walks *and* alternating cycles in the contracted eligible subgraph, such that after the removal of these cycles and walks, the subgraph no longer contains any augmenting walks. In the blossom formation step, we are given a contracted graph without any augmenting walks. The goal is to find a maximal set of reachable and contractable blossoms, i.e., a set of blossoms whose contraction will leave the graph without any reachable and contractable blossoms.

We formalize this problem, called *Disjoint Paths and Blossoms Problem*, as follows:

Definition 7 In the *Disjoint Path and Blossoms Problem*, we are given a graph $G = (V, E)$, where V is partitioned into two sets V_s, V_b, an f-matching M, and a partial function $\eta : V_b \mapsto E$ such that $\eta(v) \in \delta(v)$ if $\eta(v)$ exists. Here $v \in V_b$ represents a contracted blossom and $\eta(v)$ the incident base edge, if any. The goal is to find a set of alternating cycles $\mathcal{C}$, a set of augmenting walks Ψ where all cycles and walks are mutually edge disjoint, such that after removing all edges in $\mathcal{C}$ and Ψ, the remaining graph G does not contain any augmenting walks. Moreover, we also output a laminar set of nested blossoms Ω' that G/Ω' does not contain any reachable contractable blossom.

There are several subtleties in this definition. G here is used as a contracted graph obtained by contracting a set of nested blossoms Ω from an underlying graph. Therefore, augmenting walks and alternating cycles are defined according to Definition 4 and the definition of *alternation* from Sect. 4.1, by treating $\eta(v)$ as v's base edge when v represents a nontrivial blossom. As a result, an alternating cycle in G might not have even length in G and an augmenting walk might not have odd length in G. It is guaranteed, by Lemma 1, that the pre-images of these walks and cycles in the underlying graph are odd and even, respectively. Moreover, it is *not* guaranteed that no augmenting walk exists in the underlying graph after removing the image of Ψ and $\mathcal{C}$ in it.[7] However, it is sufficient since in the proof of Lemma 11, we only use the fact that the *contracted graph* does not contain any augmenting walks.

This problem is noticeably different from the problem solved in [15, Sect. 8] for 1-matching. Instead of looking for a *maximal* set of *vertex disjoint* augmenting paths, we look for a set of *edge disjoint* augmenting walks Ψ in conjunction with a set of *alternating cycles* $\mathcal{C}$ whose removal removes all augmenting walks from G.

Both algorithms try to search for a set of augmenting paths/walks by building an alternating structure S (not necessarily a tree) whose topology is defined in Sect. 4.1. However, in 1-matching, the search structure branches only at outer singletons and blossoms, while in f-matching, it also branches at inner singletons. As a result, when the search process reaches a vertex v, it also assigns v an inner/outer tag to remind the algorithm whether it is looking for an unmatched/non-η edge, or a matched/η edge to continue extending the structure. A vertex can obtain both inner and outer tags, but only one of them is the *primary* one where the search procedure uses it to decide which edge to explore next. If a vertex has two tags, then the non-primary one must be *exhausted*, meaning the algorithm has already finished exploring all eligible edges with respect to that tag.

A key difference between 1-matching and f-matching is that augmenting walks can be non-simple, i.e., they may contain an alternating cycle as a subwalk. Consequently, when the search process reaches an outer (inner) singleton u, it can potentially find, through an unmatched (matched) edge an inner (outer) singleton v that has already been visited before in the same search, and proceed to discover an augmenting walk. This phenomenon is illustrated in Fig. 5. If the algorithm intends to discover $(v_0, v_1, v_2, v_3, v_4, v_1, v_5, v_6)$ as an augmenting walk, it will reach v_1 with inner tag twice; first from v_0, then from v_4. Notice that in ordinary matching, edge (v_1, v_5) can-

[7] This is because multiple augmenting walks in the underlying graph can intersect a single blossom in Ω before we contract the blossom, while after contracting a blossom, any augmenting walk or alternating cycle going through the blossom will forbid the other walks and cycle to use the same blossom again (as it must go through the base edge).

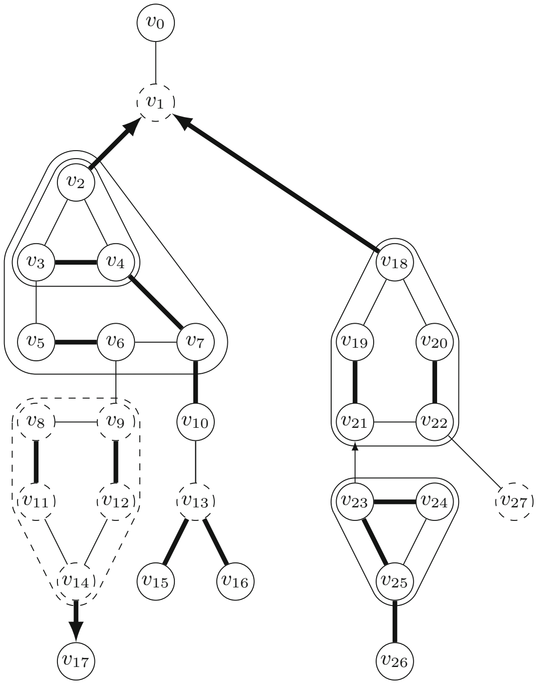


Fig. 5 An example of an eligible alternating search tree. Outer blossoms and singletons are labeled using solid boundaries while inner blossoms and singletons have dashed boundaries

not exist and edge (v_4, v_1) is ignored as it provides no useful information regarding whether v_1 is an inner/outer vertex.

One might propose to ignore and discard the edge (v_4, v_1) and return the simple path (v_0, v_1, v_5, v_6). However, edges like (v_4, v_1) cannot simply be discarded from future searches as they might participate in other augmenting walks, say $(v_{10}, v_9, v_4, v_1, v_8, v_7)$ that is edge disjoint from (v_0, v_1, v_5, v_6). To achieve a linear running time, it is essential that edges of this type only get scanned a constant number of times.

Following the spirit of DFS, we wish to maintain that every search path is not self-intersecting, i.e., each vertex is visited at most once with an *inner* tag, and once with an *outer* tag. Whenever we discover an edge that leads to a self-intersection (e.g. (v_4, v_1) in Fig. 5), we augment along the alternating cycle introduced by this edge (e.g. $(v_1, v_2, v_3, v_4, v_1)$) and thereby remove every edge on the cycle from the future searches. We backtrack to v_1 and the search continues (to the edge (v_1, v_5)). This action has the same effect as allowing augmentation along the non-simple augmenting walk $((v_0, v_1, v_2, v_3, v_4, v_1, v_5, v_6))$, but conceptually avoids a self-intersecting search structure and thus makes the analysis simpler (Fig. 6).

Overview of the Algorithm

In this algorithm, we follow a standard recursive framework for computing a maximal set of edge disjoint paths, see [23, Chapter 9]. The algorithm proceeds in phases. In phase i, $i \geq 1$, we choose a vertex r that is still unsaturated after augmentations in phases $1, 2, \ldots, i-1$, and call a procedure SEARCH-ONE from this vertex. SEARCH-ONE searches for an augmenting walk from r, and terminates with a pos-

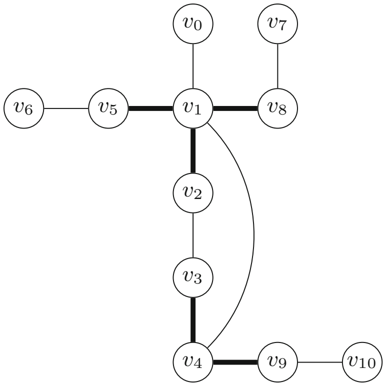


Fig. 6 Example of a self-intersecting search structure and nonsimple augmenting walk. Here v_0 is the root of the search structure and $\{v_0, v_2, v_4, v_5\}$ is the set of outer vertices and $\{v_1, v_3\}$ is the set of inner vertices. The search begins with v_0 and proceed to v_1, v_2, v_3, v_4 in order. The procedure then scan the edge (v_4, v_1) and because it connect an outer vertex to an inner vertex that is already visited, it might ignore the edge and backtrack to v_1 and return the augmenting walk $\langle v_0, v_1, v_5, v_6\rangle$. However, although (v_1, v_4) is scanned and ignored, it cannot be discarded from future search as another augmenting walk, such as the dashed walk $\langle v_7, v_8, v_1, v_4, v_9, v_{10}\rangle$ might make use of the edge (v_1, v_4) and Ψ will not be maximal if (v_1, v_4) participating in some augmenting walks

sibly empty augmenting walk P_i along with a set of disjoint alternating cycles $\mathcal{C}_i$. It guarantees that either P_i is nonempty, or in the case when P_i is empty, no augmenting walk starting from r can reach another unsaturated vertex without intersecting $\mathcal{C}_i$. It then augments along P_i and $\mathcal{C}_i$. The phase ends by discarding the set of edges encountered by the search procedure.

Formally, the input to SEARCH-ONE is a subgraph $G_i = (V_i, E_i)$ of G, an f_i-matching M_i where $M_i \subseteq M$ and $f_i(v) \leq f(v)$ for all $v \in V_i$, and an unsaturated vertex $r \in V_i$ with respect to f_i and M_i. SEARCH-ONE finds an augmenting walk P_i (possibly empty), a set of alternating cycles $\mathcal{C}_i$ and a set of edges $H_i \subseteq E_i$ that satisfy the following property.

Property 6 Any augmenting walk that intersects H_i must also intersect P_i or a cycle in $\mathcal{C}_i$.

After SEARCH-ONE terminates, we finish the phase by removing H_i from G_i. If P_i is empty, we also remove the vertex r from G_i. Let the resulting graph be G_{i+1}. Define the f_{i+1}-matching M_{i+1} by $M_{i+1} = M_i \setminus H_i$ and

$$f_{i+1}(v) = \begin{cases} f_i(v) - |M_i \cap H_i \cap \delta(v)| - 2|M_i \cap H_i \cap \delta_0(M_i)| \\ \quad \text{If } v \text{ is not a terminal vertex of } P_i. \\ f_i(v) - |M_i \cap H_i \cap \delta(v)| - 2|M_i \cap H_i \cap \delta_0(M_i)| - 1 \\ \quad \text{If } P_i \text{ is a nonempty nonclosed augmenting walk and } v \\ \quad \text{is one of the two terminal vertices of } P_i. \\ f_i(v) - |M_i \cap H_i \cap \delta(v)| - 2|M_i \cap H_i \cap \delta_0(M_i)| - 2 \\ \quad \text{If } P_i \text{ is a nonempty closed augmenting walk that starts and ends} \\ \quad \text{with } v. \end{cases}$$

Conceptually, this change restricts the f_i-matching M_i to the subgraph G_{i+1} while maintaining the property that each vertex still has the same deficiency, except for the terminal vertices of P_i, whose deficiencies are decremented by 1 (or 2 for closed walks) after augmenting along P_i. Finally, the algorithm adds the path P_i and cycles $\mathcal{C}_i$ to Ψ and $\mathcal{C}$, respectively, and terminates phase i.

A Detailed Illustration of *SEARCH-ONE*.
The call to SEARCH-ONE in phase i maintains a laminar set of blossoms Ω_i over vertices in G_i. Here Ω_i only contains the blossoms newly discovered in the search procedure and does not include the already contracted blossoms inherited from the input (vertices in V_b). In this section, we use the word *blossom* solely for the newly discovered blossoms in Ω_i and *blossom vertices* for blossoms inherited from the input, i.e., vertices in V_b. *Singletons* still refer to vertices in V_s. We use $B(v)$ to denote the inclusion-wise maximal blossom in Ω_i that contains v and $\beta(v)$ to denote the base of $B(v)$. If v is not contained in any nontrivial blossom in Ω_i, we define $B(v) = \{v\}$ and $\beta(v) = v$. Each blossom also might have a base edge $\eta(B)$. We denote the search structure on G_i with S_i, and use T_i to denote the search structure obtained from S_i by contracting all blossoms in Ω_i. Similar to [15, Sect. 8], the search structure S_i is a subgraph of G_i but not necessarily a tree, while we maintain that T_i must be a tree. We maintain T_i by storing a parent pointer for each vertex or blossom in Ω_i. Let $\tau(B)$ be the edge incident to vertex/blossom B joining it to its parent in T_i.

Blossoms are maintained using a data structure that supports the following operation: given a blossom B, a vertex v in blossom B, and a bit $s \in \{0, 1\}$, the data structure returns the alternating walk $P_s(v)$ from v to $\beta(B)$ whose existence is guaranteed in Lemma 1, in time linear in the length of the walk. This can be done using simple bookkeeping as in Gabow's implementation for Edmonds algorithm [9] and we leave the details to the readers. We also keep track of pointers such that given the blossom B, we can find $\beta(B)$ and the edge from B to its parent in T_i. Moreover, we use a union-find data structure to find $B(v)$ given v.

SEARCH-ONE explores the graph in a depth first fashion: The search begins at an unsaturated singleton or an unsaturated blossom vertex r in G_i. Similar to DFS, when the locus of the search is at u we have an alternating walk $P(u)$ from r to u in G_i. We call u the *active vertex* and $P(u)$ the *active walk*. For efficiency purposes, we do not maintain the active walk explicitly. Instead, we maintain a contracted active walk $\widehat{P}(u)$. The contracted active walk $\widehat{P}(u)$ is of the form $\langle B_0, e_0, B_1, e_1, \ldots, e_{k-1}, B_k\rangle$, where $r \in B_0$, $u \in B_k$ and each B_j are either singletons or blossoms (not necessarily maximal) in Ω_i. Each edge e_j connects B_{j-1} to B_j and edges e_j and e_{j+1} *alternate* at B_{j+1} for all $0 \le j < k-1$. The active walk $P(u)$ can be reconstructed from $\widehat{P}(u)$ in time $O(|P(u)|)$ using the blossom data structure mentioned above.

To maintain the property that the active walk is alternating, the algorithm also assigns tags for each vertex in S_i, which is either *inner* or *outer*. Tags are assigned according to Definition 6.

There is one more issue that arises in this paper compared to the similar DFS routine in [15, Sect. 8]. Upon discovering and augmenting along an alternating cycle C, the search structure S_i becomes *disconnected* from the root since C has been effectively removed from the graph. Therefore we remove all vertices and edges that descend from C in T_i from T_i and also remove their preimages from S_i. However, we still maintain some useful information about these vertices and edges. This includes the blossom structure, the former parent pointers of vertices and blossoms, as well as all the tags that vertices and edges carry and whether they are exhausted or not.

Initially, the contracted active walk consists of a single vertex r, and r is labelled outer. At each iteration, the algorithm *explores* a new edge (u, v) incident to the active vertex u that is *eligible for* u with respect to u's current inner/outer tag. Notice that this immediately defines an inner/outer tag for v. On exploring the edge (u, v), the algorithm does one of the following depending on the location of v with respect to the search structure, and the tag v is carrying.

1. *Augmentation* When v is an unsaturated singleton and (u, v) is unmatched, or v is an unsaturated blossom vertex, or when $v = r$ is the root of the search tree and the deficiency of v is at least 2, $P(u) \circ (u, v)$ forms an augmenting walk. We extend the active path with (u, v), terminate the search and set the active walk $P(v) = P(u) \circ (u, v)$ and $P_i = P(v)$. In this step, the edge (u, v) is considered *explored from* u.
2. *DFS Extension* If v is a singleton and v has never been exhausted before with the same tag, add (u, v) to the search structure S_i and make $B(v)$ a child of $B(u)$ in T_i. Set the active vertex to v and extend the contracted active walk $\widehat{P}(v) = P(u) \circ (u, v) \circ B(v)$ accordingly. In this step, edge (u, v) is considered *explored from* u.
3. *Cycle Cancellation* If both $B(u)$ and $B(v)$ are in T_i and $B(v)$ is an ancestor of $B(u)$ and (u, v) is not eligible for v, the tree path from $B(v)$ to $B(u)$, along with the edge (u, v), forms an alternating cycle C. We add C to $\mathcal{C}_i$. Retract the active walk back to v. After this step, all edges $e \in C$ will be categorized as explored from *both endpoints*.
   This step effectively disconnects all descendants of C from the root.[8] We remove this part from the search structure S_i and T_i. However, old parent pointers that were used to maintain T_i are still kept.
4. *Blossom Formation* If $B(u) \neq B(v)$ and $B(v)$ is not in $\widehat{P}(u)$, and edge (u, v) is also eligible for v, then we can potentially form a new blossom. We do so by extending the active walk from $B(u)$ to $B(v)$, then to the chain of ancestors of $B(v)$ encountered when following parent pointers, until one of the following stopping conditions is met. If we append $B(u)$ to the active walk then we stop, with a new blossom having been detected. The process may stop prematurely if (i) the next parent edge is already marked *explored* because it is in $\mathcal{C}_i$ or (ii) if the head of the walk is exhausted according to its tag.
   Notice that this step puts new tags onto vertices in these blossoms and also attaches them onto the search tree T_i. Let $\tilde{P}$ be the suffix of $\widehat{P}$ starting from $B(v)$. If $\tilde{P}$ ends at $B(u)$, we have identified a blossom B whose subblossoms consists of the set of blossoms in $\tilde{P}$. We contract the blossom by putting B into Ω_i and update the union-find data structure accordingly, then contract all vertices of $\tilde{P}$ in T_i.
   If $\tilde{P}$ ends prematurely before getting to $B(u)$, we cannot contract the blossom.

[8] It is still possible that later we discover some path from a descendant of C back to the root that circumvents C

This search has effectively appended $\tilde{P}$ to $\widehat{P}$ with DFS extensions and the search proceeds from the last element of $\widehat{P}$ as usual.

In both cases, edges are considered explored in the direction of the active walk when they first enter the active walk, and exhausted after they leave the active walk.

5. *DFS Retraction* If every edge (u, v) eligible for u has already been explored, retract from u to its predecessor on the (contracted) active walk. If $u = r$ is the only vertex in the search path, terminate the search with $P_i = \emptyset$. Otherwise, let w $(B(w))$ be the parent of u in the (contracted) active walk. The edge (w, u) is now considered *exhausted from* w. This means that every edge eligible for u is recursively exhausted and no augmenting walk can be found by following the active walk via the edge (w, u). (It may still be possible to find an augmenting walk via edge (w, u) when the search visits u again in a blossom formation step and explore (u, w) from u). Moreover, the vertex u's primary tag is now consider *exhausted*.
6. *Null Exploration* This step includes all scenarios where we explore the edge (u, v) to no effect. This includes: when $B(v)$ is a descendant of $B(u)$ and (u, v) is not eligible for v; when $B(v)$ is an ancestor of $B(u)$ and (u, v) is eligible for v; or when (u, v) is a cross edge in T_i. In these cases, we ignore the edge (u, v) while still categorizing it as *exhausted from* u.

As stated above, each edge (u, v) along with an endpoint of it, say u, has one of three statuses at any point in the algorithm:

1. *Explored from* u This means the search has visited the vertex u, extended the active walk from u to v using edge (u, v), in either Augmentation, DFS extension, Blossom Formation, or Cycle Cancellation step.
2. *Exhausted from* u This means the search has visited the vertex u, extended the search path to v via (u, v) and then backtracked to u in DFS Retraction or Null Exploration steps.
3. *Unexplored from* u If (u, v) is not considered explored or exhausted from u, it is then unexplored from u. This means the search has either yet to visit u; or the search has visited u but never extended the active walk using the edge (u, v) because it is ineligible for u; or it is eligible but the search has yet to explore (u, v).

Finally, the edge set H_i is the set of edges that are explored or exhausted from at least one of its endpoints during the search. Recall this is the set we remove from the graph G_i before termination of a phase. This completes our description of the `SEARCH-ONE` procedure.

Now we state the set of invariants satisfied by `SEARCH-ONE`.

Invariant 3 *1.* Structural Invariant of S_i: S_i *consists of a subset of edges in* G_i *that are visited during the search. Every vertex in* S_i *carries an inner tag or outer tag or both. When a vertex is outside the active walk, all its tags are exhausted. When it is inside the active walk, one tag is the primary tag. The other tag, if it exists, must be exhausted. Furthermore, If* v *is inner, there exists an alternating walk from* r *to* v *that ends with an unmatched edge if* $v \in V_s$ *or a non-*η *edge if* $v \in V_b$*. If* v *is outer, the alternating walk terminates with a matched edge if* $v \in V_s$ *or an* η *edge if* $v \in V_b$*. These walks avoid* $\mathcal{C}_i$*.*

2. Structural Invariant of T_i: *T_i is a contracted graph obtained from S_i by contracting all inclusionwise-maximal blossoms in Ω_i. T_i must be a tree. Some blossoms in Ω_i might not be represented in T_i.*
3. Depth-first property of S_i: *The union of the active walk and the set of alternating cycles $\mathcal{C}_i$ consists of precisely the edges that are explored but not exhausted from at least one endpoint. If (u, v) is an edge in H_i but not in the active walk or the alternating cycles, then (u, v) must be exhausted from u or v. Moreover, S_i is disjoint from $\mathcal{C}_i$.*
4. Maximality *If (u, v) is marked exhausted from u while (v, w) is an edge eligible for v, then the algorithm must have exhausted (v, w) from v. This holds for all edges regardless whether they are in S_i or not.*
5. Parent Pointers: *Fix any blossom B in Ω_i, possibly trivial and possibly not in T_i. If the parent pointer $\tau(B)$ is defined, consider the path generated by following parent pointers from B, terminating when one of the following conditions is met: (i) the path reaches r, (ii) the next edge in the path would be in $\mathcal{C}_i$, (iii) the last vertex in the path is exhausted w.r.t. the appropriate tag, or (iv) the last vertex in the path is in $\widehat{P}$. This path is well defined and is alternating.*

Lemma 13 Augmentation, DFS Extension, Blossom Formation, DFS Retraction, Null Exploration *and* Cycle Cancellation *all preserve Invariant 3.*

Proof The first invariant follows from how we grow the search structure S_i and active walk. When the active walk extends to a vertex v with the current tag outer, the active walk must be an alternating walk ending with a matched edge or an η edge. If the tag is inner, the active walk ends with a unmatched edge or an non-η edge. This ensures that there exists an alternating walk from the root to each vertex in S_i with a terminal edge that is consistent with its tag. Moreover, tags are labelled exhausted if and only if a vertex carrying the tag leaves the active walk by backtracking.

The second invariant follows from the fact that when we form a blossom in the Blossom Formation step, the constituent (subblossoms) in Ω_i always come from a connected ancestor-descendant path in T_i. Contracting a connected component in the tree will not create any cycle and thus T_i remains a tree.

For the third invariant, observe that an edge becomes explored from an endpoint when it joins the active walk in a DFS Extension, Blossom Formation, or Augmentation step. It becomes exhausted when it leaves the active walk at DFS Retraction, Null Exploration, and Cycle Cancellation step. Moreover, in the Blossom Formation step, since we are visiting vertices in descendant-to-ancestor direction, all edges in the active walk must remain explored and edges outside active walk are exhausted. Therefore, any edge in H_i that is not in the active walk or alternating cycles must be exhausted from at least one of its endpoints. Lastly, in blossom formation step and cycle cancellation step, we specifically maintain at any point, the active walk never uses any edge inside $\mathcal{C}_i$.

For the fourth invariant, first notice that (u, v) becomes exhausted via a DFS Retraction step or a Null Exploration step. In both cases the search must have retracted from v to some vertices and therefore has explored and exhausted every edge eligible for v, including (v, w). If w is an unsaturated singleton and (v, w) is unmatched, or w is an unsaturated blossom vertex, an Augmentation step would have occurred when the algorithm explores (v, w) and left the edge (v, w) explored and not exhausted.

For the fifth invariant, consider a blossom B and the path starting from B following the parent pointers. We call this path the *ancestral path* from B. By the inductive hypothesis, the ancestral path alternates until it ends by reaching r, or an exhausted vertex, or an alternating cycle edge, or the active walk.[9] Now consider how this path may change in a Cycle Cancellation, DFS Extension, DFS Retraction, or Blossom Formation step. In a Cycle Cancellation step, we might shorten the path by including one of its edges in an alternating cycle; this preserves the invariant. In a DFS Extension or Blossom Formation step, the active walk might extend into the ancestral path, making the ancestral path terminate earlier; this also preserves the invariant. A DFS retraction can remove the last vertex from $\widehat{P}$ that appears on the ancestral path. That vertex is by definition exhausted for its tag type, so the ancestral path terminates at the same point as before, but for a different reason. In all cases the invariant is preserved. Finally, notice that for any blossom inside T_i, the ancestral path from that blossom always alternates until it reaches the root r. □

Now we state the correctness of SEARCH-ONE:

Lemma 14 *When SEARCH-ONE terminates, if there is an augmenting path P' that intersects H_i at some edge e, then P' must intersect P_i or $\mathcal{C}_i$ at some edge.*

Proof Assume for contradiction that P' is edge-disjoint with P_i and $\mathcal{C}_i$. Let P' intersect H_i at some edge (u_0, u_1). Since (u_0, u_1) is not in P_i or $\mathcal{C}_i$, (u_0, u_1) must be exhausted in one of its directions, say from u_0. This makes (u_0, u_1) eligible for u_0. Now let $(u_0, u_1, \ldots, u_k)$ be the subpath of P' from u_0 to the terminal vertex u_k of P' in the (u_0, u_1) direction. We use induction to show that for all $0 \le i < k$, edges (u_i, u_{i+1}) must be eligible for u_i and exhausted from u_i:

The base case $i = 0$ holds by our assumption. Now suppose (u_i, u_{i+1}) is exhausted from u_i for some $i \ge 0$. Consider the edge (u_{i+1}, u_{i+2}). It is necessary that u_{i+1} was in the search structure S when (u_i, u_{i+1}) is marked exhausted from u_i. And u_{i+1} is either inner or outer or both at this moment. In particular, at this moment u_{i+1} must still own the tag that is consistent with the predecessor edge (u_i, u_{i+1}). Here, by consistent we mean the tag defined by Definition 6 by treating the edge (u_i, u_{i+1}) as the predecessor edge $\tau(u_{i+1})$. Combined with the fact that the edge (u_i, u_{i+1}) alternates with the edge (u_{i+1}, u_{i+2}), it is necessary that (u_{i+1}, u_{i+2}) is eligible for u_{i+1}. Hence by invariant 3, it must also be exhausted from u_{i+1}.

This means the edge (u_{k-1}, u_k) must be eligible for u_{k-1} and exhausted from u_{k-1}. Notice that the vertex u_k and edge (u_{k-1}, u_k) must satisfy the terminal vertex and edge requirement in Definition 4. But in this case, an augmenting walk would have been formed when the algorithm was exploring the edge (u_{k-1}, u_k) from u_{k-1}, which put the edge in P_i, which is a contradiction. □

This gives the following Lemma:

[9] Getting to the active walk does not automatically imply that you can then get to the root, since the ancestral path might not alternate at the vertex when it first reaches the active walk

Lemma 15 *SEARCH-ONE finds in time $O(m_i\alpha(m_i, n_i))$ an edge set H_i, a set of alternating cycles $\mathcal{C}_i$ and an augmenting walk P_i such that any augmenting walk P' disjoint from $\mathcal{C}_i$ that intersects H_i must also intersect P_i. Here m_i and n_i is the number of edges and vertices in H_i.*

Proof The correctness of SEARCH-ONE is argued in Lemma 14. For running time, notice that each edge we examined is always classified as explored or exhausted from at least one of its endpoints. Thus, the total number of edge examinations is $O(m_i)$. The only non-trivial data structure needed is one for maintaining the set of blossoms, in particular $\beta(\cdot)$, which can be solved in $O(m_i\alpha(m_i, n_i))$ with the standard union-find algorithm [24] . For reconstructing the active walk, we can use the bookkeeping labelling in [15, Sect. 8], which enable reconstruction of the augmenting walk in time linear in the length of the walk. □

Lemma 16 *We can find in $O(m\alpha(m, n))$ a set of augmenting walks Ψ and a set of alternating cycles $\mathcal{C}$ such that any augmenting walk P' must intersect Ψ or $\mathcal{C}$.*

Proof This algorithm can be seen as a recursive algorithm that first calls SEARCH-ONE on an input graph $G_1 = G$, finding an edge set H_1, a set of alternating cycles C_1 and an augmenting walk P_1. It removes H_1 from G_1 and the corresponding part in the f-matching to obtain G_2. Then it recurses on G_2. Let $\mathcal{C}'$ and Ψ' be the output of the recursive call. We output $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}'$ and $\Psi = \Psi' \cup \{P_1\}$.

By induction, any augmenting walk P' in G_2 must intersect Ψ' or $\mathcal{C}'$. Now suppose the augmenting walk P' contains an edge in H_1. By Lemma 15, P' must intersect P_1 or $\mathcal{C}_1$. Therefore P' must intersect Ψ or $\mathcal{C}$.

Finding a Maximal Set of Nested Blossoms

To find a maximal set of nested blossoms in the Blossom Formation step in Edmonds' Search, we can directly return the blossom set Ω' we discovered when we performs the Augmentation step as the maximal set of reachable blossom. We show that any reachable blossom after the augmentation must be included in this set. Here, the blossom as well as the alternating path from the base of the blossom to the root must avoid Ψ and $\mathcal{C}$.

Consider a blossom B that is reachable from one of the unsaturated vertices after the augmentation. Using a similar induction in Lemma 14, we can show that all edges in the path from the root to the blossom B, and edges in blossom B must be explored by the algorithm. Moreover, by the structure of the blossom, one of the edges inside the blossom that are incident to the base must be eligible for both endpoints at some point of the search and therefore must be explored in both direction. We show that this necessarily triggers a blossom step.

Lemma 17 *If (u, v) is an edge that is once explored from both u and v, then u and v must be in the same blossom in Ω'.*

Proof Notice that in depth-first search, when the algorithm explores (u, v) from both directions, $B(u)$ and $B(v)$ must be ancestor/descendant of one another in T_i. Now without loss of generality, we assume (u, v) is first explored from u. If v is a descendant of u, since the search has already backtracked from v, the only way that v enters the search again is by a blossom formation step from ancestor of u to a descendant of v, making them in the same blossom. If v is an ancestor of u, when (u, v) is explored from v, i.e., when the search backtracks from u to v, u must still be a descendant of v because any blossom step in this process will not change the ancestor-descendant relation between u and v. Then we have a blossom step triggered by (u, v) and put them in the same blossom. □

Also notice that Ω_i might includes blossoms that are not reachable because of the augmentation (say those that are disconnected from the root by an alternating cycle or an augmenting walk). However, these blossoms are also not reachable in the Dual Adjustment step and thus continue to have 0 dual values. They will be dissolved in the subsequent Blossom Dissolution step.

6 Algorithms for Unweighted *f*-Matching and *f*-Edge Cover

In this section we will give an $O(\sqrt{f(V)}m\alpha(m, n))$ algorithm for both maximum cardinality f-matching and minimum cardinality f-edge cover. This is a direct consequence of the $O(Wm\alpha(m, n)\epsilon^{-1})$ algorithm for the weighted problem. This algorithm matches the running time of [10] but does not rely on reduction to iterations of the Micali-Vazirani algorithm [20,26,27]. Moreover, it is much simpler to state and to analyze.

For illustration purposes we focus on maximum cardinality f-matching. The algorithm consists of two phases. The first phase, referred to as *batch augmentation*, finds an f-matching F that is close to optimal using an instance of the $O(Wm\alpha(m, n)\epsilon^{-1})$ algorithm. After F is close to optimum, we discard all dual variables y and z, dissolve all the blossoms in Ω and use our $O(m\alpha(m, n))$ augmenting walk algorithm to increase the cardinality of F until F becomes optimum.

This is stated formally in the following theorem:

Theorem 6 *A maximum cardinality f-matching can be computed in $O(\sqrt{f(V)}m\alpha(m, n))$ time.*

Proof We can view the maximum cardinality f-matching problem as a maximum weight problem with weight function $w(e) = 1$. Choose $\epsilon = 1/\sqrt{f(V)}$, by Theorem 3, we can compute a $(1 - \frac{1}{\sqrt{f(V)}})$-approximate maximum cardinality matching F in $O(\sqrt{f(V)}m)$ time. If F^* is the maximum cardinality f-matching, we have

$$|F| \geq \left(1 - \frac{1}{\sqrt{f(V)}}\right)|F^*| > |F^*| - \frac{1}{\sqrt{f(V)}} \cdot \frac{f(V)}{2} > |F^*| - \sqrt{f(V)}/2.$$

This means F is only $O(\sqrt{f(V)})$ augmentations away from optimal. Hence we can then discard the blossom structure Ω with duals y and z from the approximate f-matching and run the $O(m\alpha(m, n))$ augmenting walk algorithm of Lemma 16 in G with respect to F until F is optimal. By the discussion above, $O(\sqrt{f(V)})$ iterations suffice. The total running time of the algorithm is $O(\sqrt{f(V)}m\alpha(m, n))$.

We can also use Gabow's linear time subroutine [11] that finds one augmenting walk at a time. This can improve the running time to $O(\sqrt{f(V)}\alpha(m,n)m)$ $\square$

Theorem 7 *A minimum cardinality f-edge cover can be computed in $O(\sqrt{f(V)}m\alpha(m,n))$ time.*

Proof This is similar to Theorem 6. We first use the $O(Wm\epsilon^{-1})$ algorithm for f-edge cover in Corollary 1 to find an $(1+\sqrt{f(V)}^{-1})$-approximate minimum cardinality f-edge cover F by viewing the graph as a weighted graph with weight 1 everywhere. Choosing $\epsilon = 1/\sqrt{f(V)}$ will give us an f-edge cover F with $|F| \le |F^*| + \sqrt{f(V)}^{-1}|F^*|$. Notice that we always have $|F^*| \le f(V)$ because taking $f(v)$ arbitrary incident edges for each v and taking their union will always give a trivial f-edge cover with cardinality at most $f(V)$. Hence we have $|F| \le |F^*| + \sqrt{f(V)}$, which means at most $O(\sqrt{f(V)})$ reductions are needed to make F optimal. Therefore we can run the augmenting path algorithm from Lemma 16 to find reducing paths (P is a reducing path w.r.t. F if and only if it is an augmenting path w.r.t. $E \setminus F$) until no reducing path can be found. There are $\sqrt{f(V)}$ iterations in this phase. The total running time of the algorithm is $O(\sqrt{f(V)}m\alpha(m,n))$. $\square$

Acknowledgements Supported by NSF grants CCF-1217338, CNS-1318294, CCF-1514383, CCF-1637546, and CCF-1815316.

References

1. Choromanski, K., Jebara, T., Tang, K.: Adaptive anonymity via *b*-matching. In: Advances in Neural Information Processing Systems 27: 27th Annual Conference on Neural Information Processing System, NIPS 2013, pp. 3192–3200 (2013)
2. Chvátal, V.: A greedy heuristic for the set-covering problem. Math. Oper. Res. **4**, 233–235 (1979)
3. Drake, D.E., Hougardy, S.: A simple approximation algorithm for the weighted matching problem. Inf. Process. Lett. **85**, 211–213 (2003)
4. Duan, R., Pettie, S.: Linear-time approximation for maximum weight matching. J. ACM **61**, 1:1-1:23 (2014)
5. Duan, R., Pettie, S., Su, H.: Scaling algorithms for weighted matching in general graphs. ACM Trans. Algor. **14**, 8:1-8:35 (2018)
6. Edmonds, J.: Maximum matching and a polyhedron with 0,1-vertices. J. Res. Natl. Bureau Stand.: Sect. B Math. Math. Phys. **69B**, 125–130 (1965)
7. Edmonds, J.: Paths, trees and flowers. Can. J. Math. **17**, 449–467 (1965)
8. Edmonds, J., Johnson, E.L.: Matching: A well-solved class of integer linear programs. In: Combinatorial Optimization - Eureka, You Shrink!, Papers Dedicated to Jack Edmonds, 5th International Workshop, pp. 27–30 (2001)
9. Gabow, H.N.: An efficient implementation of Edmonds' algorithm for maximum matching on graphs. J. ACM **23**(2), 221–234 (1976)
10. Gabow, H.N.: An efficient reduction technique for degree-constrained subgraph and bidirected network flow problems. In: Proceedings of the 15th Annual ACM Symposium on Theory of Computing, STOC 1983, pp. 448–456 (1983)
11. Gabow, H.N.: Data structures for weighted matching and extensions to *b*-matching and *f*-factors. ACM Trans. Algor. **14**, 39:1-39:80 (2018)
12. Gabow, H.N., Sankowski, P.: Algebraic algorithms for *b*-matching, shortest undirected paths, and *f*-factors. In: 54th Annual IEEE Symposium on Foundations of Computer Science, FOCS, pp. 137–146 (2013)
13. Gabow, H.N., Tarjan, R.E.: A linear-time algorithm for a special case of disjoint set union. J. Comput. Syst. Sci. **30**, 209–221 (1985)

14. Gabow, H.N., Tarjan, R.E.: Faster scaling algorithms for network problems. SIAM J. Comput. **18**, 1013–1036 (1989)
15. Gabow, H.N., Tarjan, R.E.: Faster scaling algorithms for general graph-matching problems. J. ACM **38**, 815–853 (1991)
16. Håstad, J.: Clique is hard to approximate within $n^{1-\epsilon}$. Acta Math. **182**(1), 105–142 (1999)
17. Khan, A., Choromanski, K., Pothen, A., Ferdous, S.M., Halappanavar, M., Tumeo, A.: Adaptive anonymization of data using b-edge cover. In: Proceedings of the International Conference for High Performance Computing,Networking, Storage, and Analysis, SC, pp. 59:1–59:11 (2018)
18. Khan, A.M., Pothen, A.: A new 3/2-approximation algorithm for the b-edge cover problem. In: 2016 Proceedings of the Seventh SIAM Workshop on Combinatorial Scientific Computing, CSC, pp. 52–61 (2016)
19. Lawler, E.: Combinatorial Optimization: Networks and Matroids. Dover Books on Mathematics Series (2001)
20. Micali, S., Vazirani, V.V.: An $O(\sqrt{|V|}|E|)$ algorithm for finding maximum matching in general graphs. In: 21st Annual Symposium on Foundations of Computer Science, FOCS, pp. 17–27 (1980)
21. Pettie, S., Sanders, P.: A simpler linear time $2/3 - \epsilon$ approximation for maximum weight matching. Inf. Process. Lett. **91**, 271–276 (2004)
22. Pulleyblank, W.: Faces of matching polyhedra. PhD thesis, University of Waterloo, Ontario, Canada (1973)
23. Schrijver, A.: Combinatorial Optimization: Polyhedra and Efficiency. Springer, Algorithms and Combinatorics (2002)
24. Tarjan, R.E.: Efficiency of a good but not linear set union algorithm. J. ACM **22**, 215–225 (1975)
25. Tutte, W.T.: On the problem of decomposing a graph into n connected factors. J. Lond. Math. Soc. **s1–36**, 221–230 (1961)
26. Vazirani, V.V.: A theory of alternating paths and blossoms for proving correctness of the $O(\sqrt{|V|}|E|)$ general graph maximum matching algorithm. Combinatorica **14**(1), 71–109 (1994)
27. Vazirani, V.V.: An improved definition of blossoms and a simpler proof of the MV matching algorithm. CoRR, arXiv:1210.4594, (2012)
28. Zuckerman, D.: Linear degree extractors and the inapproximability of max clique and chromatic number. Theory Comput. **3**(1), 103–128 (2007)